University of Notre Dame
Department of Electrical Engineering

EE41440 Senior Design

---

uTune: An Automated Guitar Tuning Solution

Submitted To:
Dr. R. Michael Schafer
Professor
Electrical Engineering

Submitted By :
Christian Femrite
Joe Palasek
Pete Rymsza
Kevin Schneier
Ben Wdowik

Contents

1 Introduction

1.1 Motivation

Stringed instruments have long been utilized to create music by manipulating the properties of the strings to control pitch. These instruments range from lyres and guitars to violins and pianos, and they incorporate a number of different elements in order to control pitch and thus create music. Three key factors that affect pitch are string length, linear mass density, and tension. While linear mass density remains constant in a particular guitar string and the length is used to change pitch while playing, the tension in the string changes the pitch of the open string and thus changes the reference point for all fretted notes. Another higher-order effect occurs when one string is drastically tightened or loosened, as this results in a change in the overall tension at the bridge of the guitar, changing the tuning of the other strings. Another factor that significantly impacts tuning is temperature, and if there are significant temperature changes during, say, an outdoor performance, then proper adjustments would need to be made to ensure that a guitar would remain in tune.

Many different guitar tuning systems have been developed throughout the course of history. Standard tuning is known as EADGBE from low string to high string, and this is the common tuning used in many mainstream guitar songs of all genres. It originally came to be as a result of many experimental iterations of the guitar, and it turned out to be the tuning that seemed to require the least amount of finger movement between chord changes on the fretboard. However, this is not the best tuning system for other playing styles, such as fingerstyle, and many particular songs use unique tunings. General and song-specific tunings such as DADGBE (known as "drop D"), DADGAD for fingerstyle, D#G#C#F#A#D# or DGCFAD for a half or full step down, CGDGBD for Constellations by Jack Johnson, C#A#D#G#A#D# for Barcelona by Ed Sheeran, EABGBD# for Yellow by Coldplay, DADF#AD for The Cave by Mumford and Sons, EBEF#BE for In Your Atmosphere by John Mayer, and many others are prominent in the music world and required in a guitarist's repertoire today.

In a performance environment, it is paramount to be able to transition between these various tuning schemes smoothly and effortlessly. The current solution is often to simply maintain several guitars on/off stage during a performance and quickly swap them in between songs. Unfortunately, this has the disadvantages of (1) additional cost of buying multiple guitars, (2) difficulty transporting extra equipment, and (3) potential for the tuning of a guitar to change as it sits unused.

1.2 Background and Features

A variety of use cases present themselves for a device such as this one. The first and most obvious case is the aforementioned rapid-switch performance scenario where a quick re-tune or tuning switch is essential to the momentum of a performance. The other primary use case is for musical beginners who have not developed their sense of pitch and intonation. Although most people can tell when a particular pitch is higher than another, it takes time to refine the ear to tell if a pitch is flat or sharp, especially as the string pitch approaches the desired note. Often, it can take years to refine the ability to a point of perfection of relative pitch, and not many people are gifted with the ability of perfect pitch (the ability to identify pitch without a reference). A device that centers on a tuning pitch automatically would provide the music student the experience of listening to a tuning procedure to reinforce the auditory sensation of progression toward intonation. Additionally, to properly tune a guitar for anything other than a solo performance requires a reference pitch. This is avoided by hard-coding the desired note frequencies so that the guitar returns to the same absolute pitch every time, rather than tuning relative to the lowest string.

Alongside the development of a beginner's ear training, this raises questions about how to measure the numerical tolerances of a pitch, and what levels of error are detectable by the human ear. The concept of a "critical bandwidth" describes the difference in frequency at which a listener begins to distinguish two simultaneously-played pitches as distinct rather than clashing. The "equivalent rectangular bandwidth" (ERB) is a measure of this phenomenon, and is a function of the center frequency of the two pitches. Interestingly, what is found is that the human auditory system is better capable of distinguishing notes at higher frequencies than notes at lower frequencies by this measure (in terms of the number of half-step separations). Since higher semitone intervals entail naturally wider frequency bands, this result makes intuitive sense. As a consequence, it means that if a tuning device has a constant resolution across all

frequencies, it must be acceptable on both ends of the audible spectrum. A constant frequency error tolerance across all frequencies would theoretically result in larger perceived errors at lower frequencies (because this error is a larger proportion of the base frequency), but as a result of the phenomenon of the critical bandwidth, the human ear is not as capable of distinguishing these errors. This result lends itself to the implementation of a Fast Fourier Transform with constant spacing resolution which is the central element of the uTune algorithm.

As an extension of these considerations, there is an entire developing field of study in the area of psychoacoustics, but one of the more fundamental concepts of tuning is the mathematical reasoning behind the spacing of notes in the Western musical system. There are two primarily referenced tuning systems: equal or just temperament. Nearly always, the equal temperament is used due to its ability to provide consonant sound in a wide variety of keys and musical styles. Equal temperament is defined by the fact that (1) octaves of notes are integer multiples of one another and (2) the notes are all perfectly geometrically spaced. By contrast, the just tuning system requires each note within a single octave to be tuned to a very specific multiple of the base note. The requirement of a selection of this base note limits instruments tuned to this system to essentially playing in this key, else strange harmonic misalignments will be perceived in a different key.

With a system such as this proposed automatic guitar tuner, there is naturally a wide range of acceptable possible features. Of course, the primary one is that it must turn the pegs on the guitar head properly to tune the guitar, but the automation level falls on a spectrum. Should the guitar pluck its own strings? Should the strings be plucked individually (like a standard tuning procedure), or simultaneously (for quicker tuning)? Should the device be able to play its own music like a self-playing piano? How should the user interact with the device? Which motors will be the most size-efficient for the product? All of these were considerations made in the design process. An uncountable number of different tuning apps and handheld tuners are already out on the market, but the differentiating factor of uTune is its ability to execute the tuning process on behalf of the user. Ultimately, uTune is capable of tuning one string's peg at a time (in any order) to custom and pre-set tuning schemes of the user's selection. This selection of frequencies is intentionally limited to the practical range in which a string would be played (i.e. the strings may only be tuned down, as any song that would require an "up-tuning" would simply require a capo).

1.3 Vision

From the outset, there was a vision of the uTune device. The first conceptualization was that it would be a lightweight, low-power, small structure mounted on the back of the guitar head, potentially replacing the tuning pegs themselves with servo motors to keep the device as contained as possible. The device would have a "tuning mode" and a touch screen on the back of the guitar head that would allow the user to select their tuning scheme. With a single strum, the microphone on the guitar head would pick up the signal and turn the pegs into a perfectly tuned position. As this was a lofty goal of a device, there were naturally some difficulties in creating this perfect ideal of what an automatic guitar tuner should be, but the design did come close to meeting many of these factors. First of all, though the microphone was not located on the guitar head but rather in the body of the guitar, this is an issue that could have been addressed with more convenient installation instructions, remote Bluetooth connection to the microphone (since Bluetooth was already enabled for the iPhone app), or by using an adjustable gain operational amplifier. Secondly, the touch screen operated nearly exactly as desired, offering the user feedback on tuning progress while tuning, as did the iPhone app in conjunction with the touch screen (i.e. both could be operated simultaneously without worry of conflicting settings). Next, the size of the device proved to be something difficult to minimize. The size of the servo motors available within budget proved to be bulkier than desired, although not unwieldy enough to prohibit the 3D construction of a final prototype. The physical structure of the device was made a second priority to the electrical functionality. However, the final structure was not as heavy as anticipated due to the lightweight nature of the 3D printed device apparatus.

At the conclusion of the project, a variety of future improvements for the uTune device itself were considered and elaborated upon. This discussion will take place in a later section, but there are also myriad other considerations for how similar technologies could be used in other applications. For example, if a similar product existed that could tune pianos in a reasonable amount of time without the assistance of a professional piano tuner once or twice every year (for over one hundred dollars per visit!), it would save

people a great deal of money. There are also several orchestral stringed instruments that could benefit from an automatic tuning technology. Instruments such as the violin, viola, and cello are naturally more difficult to tune because of the lack of fret divisions on the bridge such as there are on a guitar. Not only could an orchestral uTune device be used to tune to the correct pitches, but it could also foreseeably be used to correct pitch mid-performance and correct as mistakes are made mid-symphony, or as tuning drifts throughout a performance.

2 System Requirements

2.1 Overview

In this section, a general set of requirements are outlined for each major component of the uTune device.

2.2 Motors

Torque: Each motor must provide sufficient torque to turn a guitar peg, with string attached, to three notes above its standard tuning pitch (e.g. must be able to turn E string up to a G).

Speed: Each motor must be able to complete a full rotation in under 10s.

Weight: The guitar player must be able to individually support the guitar with tuner attached. Each motor must weigh no more than 75g.

Accuracy: Each motor must be able to turn to within 0.5 degrees in order to meet pitch accuracy requirement.

Current: Each motor must draw no greater than 1 amp (stall current).

Attachment: Each attachment must connect securely to the motor and peg and require no more than $\frac{1}{4}$ turn of the peg when attaching.

Cost: Each motor and connector combination must cost less than $15.00.

2.3 Batteries

Power: Device must be able to run entirely on battery power. Batteries need to have sufficient ability to power 6 motors in tandem, LCD screen, microphones, wireless chip, and microcontroller.

Charge: Batteries must be rechargeable or easily replaceable when burnt out.

Weight: Batteries must collectively weigh less than 100g.

Cost: Rechargeable batteries must cost less than $5.00 each. Non-rechargeable (i.e. replaceable) batteries must cost less than $1.00 each.

Time: Rechargeable batteries must be able to provide no fewer than 100 full tunes before charging. Replaceable batteries must be able to provide no fewer than 1000 full tunes. The device must be able to remain powered on for no fewer than 6 continuous hours on new or fully charged batteries.

2.4 Microcontroller

Abilities: The microcontroller must be able to:

-read analog microphone signals, convert to digital data, perform frequency analysis (i.e. FFT), and run algorithm to find local peaks and determine current string frequency

-interface with mobile app and LCD screen to determine user-selected string frequencies

-decide based upon user selection and current string frequency how to control the motors

-send a signal to an amplified power source to turn the motors such that they tune the string according to the user's input

Size: The microcontroller and accompanying circuit components must fit on a PCB smaller than the head of the guitar in all dimensions.

Weight: The microcontroller and board assembly must weigh no more than 70g.

Cost: The microcontroller must cost no more than $20.00.

2.5 Physical Build

Weight: Device must weigh little enough that the guitar player is able to comfortably support the guitar and tuner setup while standing. Entire apparatus must weigh less than 500g.

Assembly: Device must be relatively simple to put-on and take-off. Either procedure must be possible for an experienced user to complete in under 1 minute.

Robustness: Assembly must be constructed from materials that will allow the device to survive normal operation for at least two years of frequent usage.

Cost: Assembly for device (i.e. not including components such as motors, PCB, etc.) must cost less than $30.00.

2.6 LCD

Size: The LCD screen must be large enough to display 6 characters for the tuning selection and several lines to indicate menu options and settings, but must be no wider than the head of the guitar.

Weight: The LCD screen must weigh no more than 50g.

Visibility: The LCD screen must have resolution greater than 8 pixels per character.

Buttons: The LCD screen must have three connected buttons, to control up and down selection and an "enter" or "next" button to change selections. These must be between 0.5 and 1 cm in diameter, and must weigh no more than 5g.

Cost: The LCD screen must cost no more than $10.00. The buttons must cost no more than $0.50 each.

2.7 Mobile App

Devices: The app must be able to connect to a minimum of one device at a time.

Range: The tuner must be able to connect to the mobile device up to a distance of at least 10 ft.

Bluetooth Chip: The bluetooth chip must be low-energy (BLE), smaller than 2 square centimeters, and easy to program and interface with the microprocessor.

2.8 Frequency Detection System

Size: The frequency detection system (e.g. microphones) must be no larger than 1 cm in diameter, such that they can fit in the corners of the guitar head or another location that is optimal for detecting pitches.

Weight: The frequency detection assembly must weigh no more than 40g.

Placement: The microphone assembly must be built into the device such that the user does not need to place them at all, other than simply clipping the device onto the guitar head.

Quantity: No more than 6 frequency detection devices must be utilized.

Cost: The frequency detection assembly must cost less than $20.00.

2.9 Safety

Exposed electricity: There must be no exposed metal or wires, such that there exists no possibility of shock or electrocution during normal, safe operation.

Heat: There must be no exposed components which become very hot, such that the user may manipulate the device and put it on the guitar or take it off with no possibility of burning their hand, and so that there is no risk of fire.

Shut-off: The device must include a shut-off button or switch that is easily accessible, easy to operate, and will immediately shut-off the device, stopping all operations immediately. This is to prevent a scenario in which the device malfunctions and continuously operates a motor such that there exists a possibility of a string snapping, which could injure the user.

3 Project Description

uTune is an automated guitar tuner designed to complete the process of turning a guitarist's tuning pegs without outside assistance. The involvement of the user is limited to the mechanical strumming of the strings as is usually done during a manual tuning sequence, but the labor and iterative pitch correction is left to the functionality of the uTune device. This allows for greater convenience and flexibility in both modifying and correcting the tuning of the guitar.

The uTune device is comprised of a primary housing which hosts a set of three printed circuit boards, each dedicated to a different aspect of the device functionality. The main PCB is mounted to the back of the physical apparatus and features a dsPIC microcontroller, a Bluetooth module, and electrical connections to the touch screen, power board, and microphone board in addition to programming/debugging connections. The power board features USB charging via a power management chip as well as both 5-volt DC power for the motors and 3.3-volt DC power for the other modules. The microphone board is mounted inside the guitar body under the strings and connected via a long wire to the main PCB, and also includes an operational amplifier in order to increase the signal to a level better interpreted by the ADC on the dsPIC.

3.1 System Theory of Operation

3.1.1 Initial uTune Configuration

When uTune is first powered on, the reset button (labeled on the main PCB) must be pressed in order to reset the touch screen display. Next, the arrows may be pressed by the user to reach the desired notes to which each string should be tuned (the boxes containing the desired pitches, from left to right, are listed in order of increasing pitch). Once this set of pitches is determined and is displayed, pressing the "Tune" button on the main screen will enter the device into tuning mode. Alternatively, the user may accomplish the same setup via the iPhone app which includes a similar 6-note display and a button to enter the device into tuning mode within the app. In tuning mode, the user must continuously pluck each string individually until the corresponding letter on the touch screen turns green, indicating that the string is in tune. During this process, if the current pitch is sharper than the desired pitch for that particular string, a red dot will appear above the corresponding note box on the touch screen, and if the current pitch is flatter than the desired pitch for that particular string, a red dot will appear below the corresponding note box on the touch screen. The strings may be tuned in any order, and it is not required to tune every string while in tuning mode. After tuning is completed, simply press "Stop" on the touch screen to exit the device from tuning mode.

3.1.2 Description of the Signal Path

A core functionality of uTune is the audio path, which entails converting sound produced by the guitar into meaningful electrical information which can be used to execute the turning of the pegs of the guitar to properly tune it according to the user's input. This involves a closed feedback loop which implementas a proportional control algorithm that continuously executes when uTune is in tuning mode. When in tuning mode, the system waits in standby for a microphone intput. The 12mV level of the microphone input is amplified to the 3.3V input level of the ADC on the dsPIC, which is then sampled at a rate of approximately 1.33kHz in order to capture sound data through the second harmonic frequency of the high E string, which is the highest pitch that the guitar would ever need to be tuned. These 10-bit samples fill a buffer of length 1024 in the dsPIC onboard memory (giving a frequency resolution of approximately 1.3Hz), where a fast-Fourier transform is taken followed by a peak identification algorithm. These identified peaks are utilized to determine the currently-observed fundamental frequencies of the strings, and this data may be leveraged to implement a control algorithm for the motors such that they will turn the pegs an angle (in the correct clockwise or counterclockwise direction) proportional to the frequency correction needed to be made in order to change the operating frequency of the string to the desired frequency.

3.2 System Block Diagram

The following diagrams consist of a number of block diagrams, to illuminate the functionality of a number of different subsystems. These will be explained in more detail later on in this section. Figure 1 shows the most basic user inputs and outputs, Figure 2 shows the overall connections of the subsystems, and Figures 3-6 show the operation of each subsystem.

Figure 1: uTune "Black Box" Inputs and Outputs

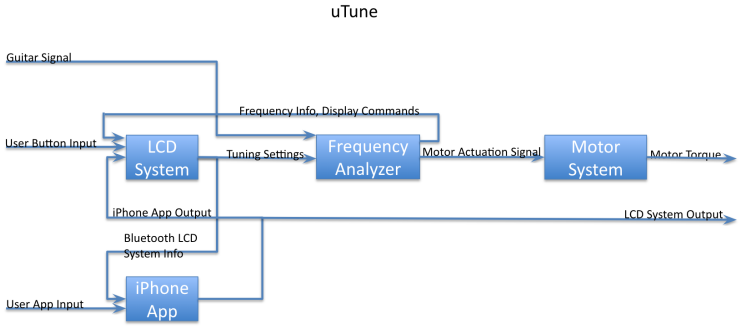

Figure 2: uTune Overall Block Diagram

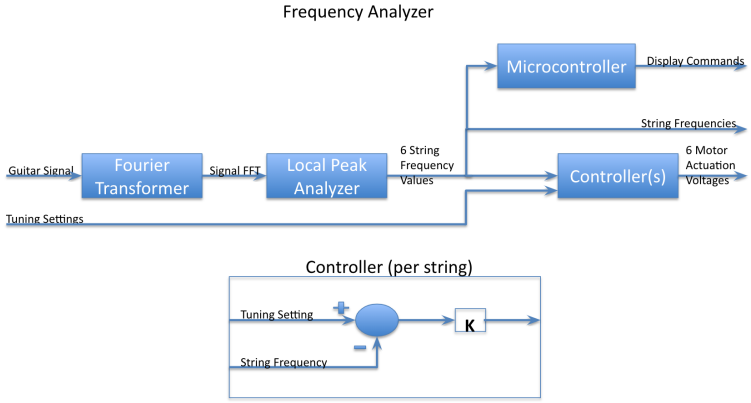

Figure 3: uTune Frequency Analysis and Control System Block Diagram

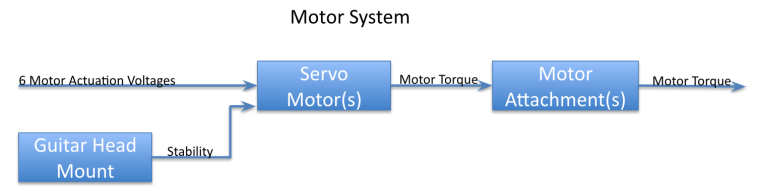

Figure 4: uTune Motor System Block Diagram

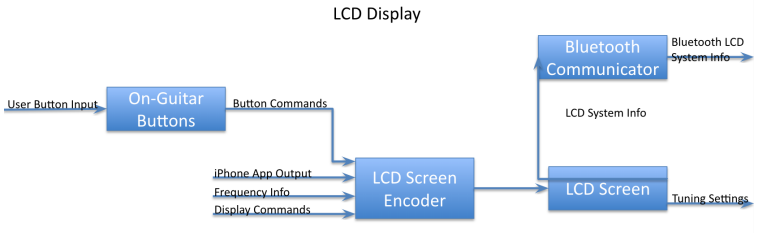

Figure 5: uTune LCD Display Block Diagram

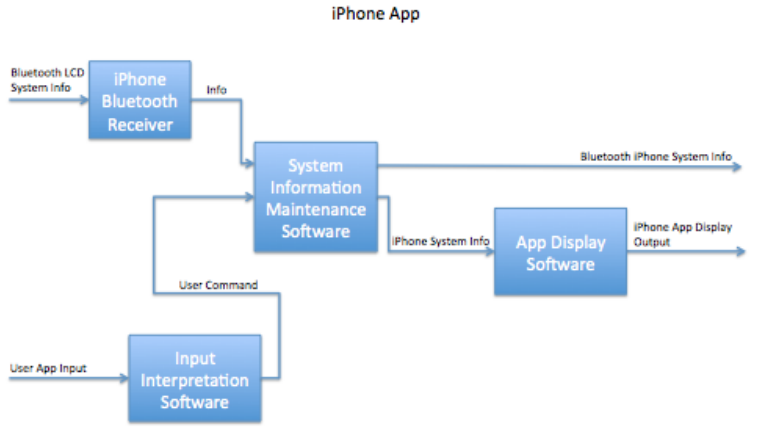

Figure 6: uTune iPhone App Block Diagram

3.3 Power Subsystem Design and Operation

The power subsystem was designed with several high-level objectives. These include:

(1) Provide adequate power distribution to the peg-turning motors during simultaneous operation while maintaining sufficient power for background operation (including processing and touch screen display).
(2) Allow an option for portable USB charging instead of requiring a power source or a wall outlet.
(3) Include sufficient battery capacity to meet requirements and avoid frequent recharging.
(4) Provide a safety mechanism to avoid overheating should the charge regulation system fail.

In order to achieve these goals, the best components available were initially selected. The most detail-oriented approach to power management seemed to be the option of using a "gas gauge" chip which measures current flow to and from the battery in order to assess and display current battery life. Of the available options, the best seemed to be the Texas Instruments BQ28Z610DRZR Li+ Gas Gauge chip. It offers autonomous battery charging control using a dedicated I2C interface, high-side protection for communication

during fault cases, programmable protection levels for current, voltage, and temperature, and emphasizes its own low power consumption. Additionally, two RPM-2.0 DC/DC converters were initially selected largely on the basis of their claimed efficiencies of 98 percent. Ideally, power conversion from the 3.7V Li+ battery cell would result in minimal losses. Unfortunately, after further research and after a preliminary board design, it was determined that the packaging options and footprints offered for all of these chips would either not be feasible for either hand-soldering or stencil-soldering, or require dimensional tolerances too fine for PCB board house manufacturing. As a result, it was not possible to design a PCB within budget given these constraints, and the ball-grid-array (BGA) packaging of the DC/DC converters especially proved to be an unacceptable risk for later in the design process in terms of debugging interfacing.

After a further search for more physically practical components, the core IC for charging management that was selected is the Texas Instruments BQ2057CSN Li+ power management IC. Aside from its optimal feature combination such as one percent voltage regulation tolerance, more serviceable packaging, and automatic low-power mode, one of the convenient features of this chip was that the same packaging and footprint were available for both two-cell and one-cell Li+ battery configurations. This meant that even if ultimately a design change was made later, it would be possible to use a two-cell Li+ battery instead of the one-cell option. The decision to use a one-cell battery will be discussed further later. Though this chip lacked the detailed "gas gauge" features that the first chip had embedded, it was suitable to guide the charging sequence through the three charging phases (as shown in Figure 7). More importantly, however, this IC includes a charge indicator which uses a simple LED circuit to indicate that it is charging properly while it is plugged in. This greatly aided in the debugging process for this subsystem. Additionally, a thermistor was included in the design as a method of disabling charging in the event of overheating. With this particular IC, the temperature boundaries are user-selected using formulas in the BQ2057CSN datasheet. For the purposes of this project, that temperature range was set to be between 0 and 60 degrees Celsius. The thermistor is glued (or otherwise secured) to the battery such that its resistance changes with temperature, resulting in pulling the signal on the corresponding pin either high or low depending on the direction of the tempreature fault.

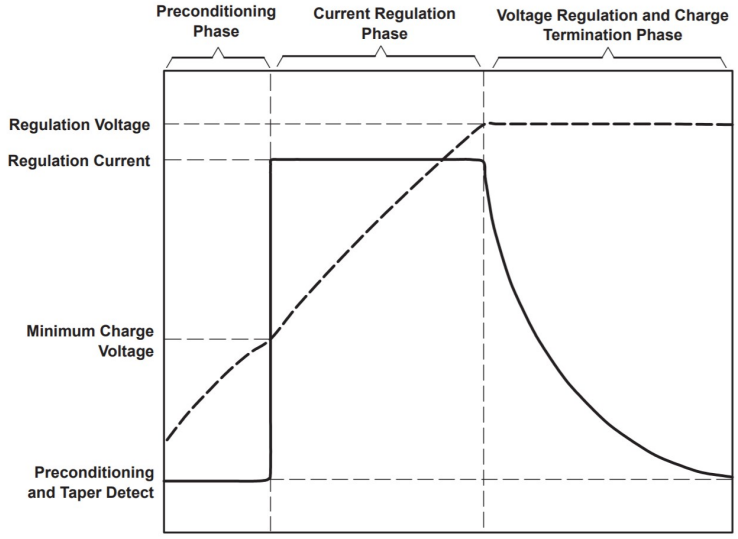

Figure 7: Typical Charge Profile

The BQ2057CSN power management IC operates by controlling the current delivered to the battery as it charges at a voltage set by the collector voltage of a FZT788B BJT power transistor. Internally, the power management chip is aware of the current passing into the emitter of this BJT because it measures a voltage difference between VCC and the voltage on a pin after a small amount of voltage is dropped across a precision 0.2ohm resistor. From Ohm's Law, internally it can extrapolate the current that is being delivered from the collector, and adjust the base current using the CC pin using a feedback loop. When this charging sequence is ongoing, the status LED is lit. Other than the thermistor which disables this process during a temperature fault, the majority of the other components on the board are primarily decoupling capacitors.

A summary of the pinouts of the BQ2057CSN Power Management IC can be seen in Table 1:

Table 1: BQ2057CSN Power Management IC Pins

Pin Name (BQ2057CSN)	**Pin Description**
BAT	Voltage sense input
CC	Charge control output
COMP	Charge rate input compensation (AutoComp)
SNS	Current sense input
STAT	Charge status output
TS	Temperature sense input
VCC	Supply voltage
VSS	Ground

After finding a suitable power management IC and knowing that the objective was to allow for battery charging via USB, an issue arose. USB3.0 can deliver 5V at 900mA. This provides a fundamental charging limitation. However, Li+ batteries only come in multiples of 3.7V supplies due to their chemistry. This implied either choosing a one-cell battery and boosting its output to the motors at 5V, or choosing a two-cell solution and regulating the voltage level given to the motors while boosting the input voltage from the USB charger. The latter seemed initially more complicated due to the fact that it required an additional voltage level change to 7.4V for two-cell storage, which also seemed to pose an issue for charging speed if this initial boost reduced the maximum USB charging current. The one-cell solution ultimately proved to function perfectly well, but the initial concern was primarily in finding a proper boost converter that would supply enough current to the motors.

As the power management IC solved the problem of charging a Li+ battery via USB, the other side of the system power needed to be solved as well, i.e. power distribution. This requires ample power at 3.3V as well as substantial power capacity at 5V to power the motors as the draw their peak current in order to avoid power shortages throughout the rest of the device. To accomplish this, two different ICs were selected. The first was the XC9141B50CMR-G boost converter, capable of converting a 3.7V battery input into a 5V output at 0.8A with minimal voltage dropout (experimentally 0.3V drop when drawing more than 500mA). This was originally selected with the intention of running six motors simultaneously that could each draw 100mA at their peak power draw. Additionally, its power conversion efficiencies approach 94 percent if the input voltage is high enough (3.7V input more than sufficed in our case). This chip required minimal external components, primarily for power decoupling and oscillation minimization. Initially, this chip caused some difficulties as the first iteration of the power board had mis-routed traces to this chip, shorting power and ground, but ultimately this problem was solved and it provided ample functionality. The second selected IC for power regulation was the TLV71333PDBVR Linear Regulator. This linear regulator is capable of an output of up 150mA at a fixed 3.3V, which is enough to power the rest of the subsystems. Ultimately, a linear regulator proved sufficient for the purposes of this project, especially since a small voltage conversion from 3.7V to 3.3V would not dissipate a lot of power inherently. Also, this chip occupied minimal board space, only requiring additional decoupling capacitors. As a whole, these voltage conversion chips and the power management chip occupied a reasonably-sized board that met the goal of mountability on the back of the uTune physical apparatus. The board layout for the power subsystem can be seen in Figure 29 in Appendix A.

3.4 Touch Screen Display Subsystem Design and Operation

This device utilizes an Adafruit ILI9341 2.4" TFT LCD (thin-film transistor liquid-crystal display), which comes with a resistive touchscreen mounted to the surface and built-in controller with RAM buffering to simplify the work required by the dsPIC. This product was selected due to the abundance of online resources, including GitHub repositories of header files with functions to draw shapes, upload images, and more (note: these resources are designed for Arduino and had to be rewritten in C for use in the MPLAB IDE).

Table 2: Touchscreen to Microcontroller Pin Connections

Pin Description	**Pin Name (Device)**	**Microcontroller**
Ground	GND	Ground
Input Voltage	VIN	3-5V
SPI Clock	CLK	SDK
SPI Master In Slave Out	MISO	SDI
SPI Master Out Slave In	MOSI	SDO
SPI Chip Select	CS	Digital pin
SPI Data/Command Selector	DC	Digital pin
Touchscreen Y+	Y+	Analog pin
Touchscreen X+	X+	Digital pin*
Touchscreen Y-	Y-	Digital pin
Touchscreen X-	X-	Analog pin

*Digital pin X+ is directly connected to the SPI Data/Command Selector digital min, thus resulting in only ten connections.

The LCD display with touch capability required ten connections to operate (see Table 2). The display itself runs via standard SPI, with connections for Master-In/Slave-Out and Master-Out/Slave-In, chip select, and clock, as well as power and ground. The touchscreen requires an additional four pins, two digital outputs and two analog inputs, which are used to determine the x and y coordinates of a touch, as well as the amount of pressure applied. The microcontroller continually switches the digital pins between permutations combinations of high and low voltage, while sampling the corresponding analog pins to get the X, Y, and pressure data. When X+ is set to high and X- is set to low, Y+ is read via the ADC on the dsPIC to obtain the X value. When Y+ is set to high and Y- is set to low, X- is read via the ADC on the dsPIC to obtain the Y value. Lastly, when Y- is set to high and X+ is set to low, both Y+ and X- are read via the ADC on the dsPIC to obtain the pressure applied.

The user interface (UI) for the touchscreen is designed to allow for most of the same functionality as the iPhone app. This includes the ability to select different tuning schemes and start/stop the tuning process, but does not include the ability to store or save those tuning schemes. As shown in Figure 8, the touchscreen UI consists of 6 boxes (one for each string), arrows above and below the box (when selected) to allow the user to adjust the intended note up or down (within pre-determined limits), and a "tune" button to initialize the process. When this button is pressed, uTune enters tuning mode and the user cannot select or alter the notes until the "stop" button is pressed. The "flowchart" for the UI is extremely simple, since it operates in only two modes: it is either in (a) tuning mode, which restricts touch features and provides tuning feedback, or (b) not tuning mode, which allows the user to alter the tuning scheme and initialize tuning mode.

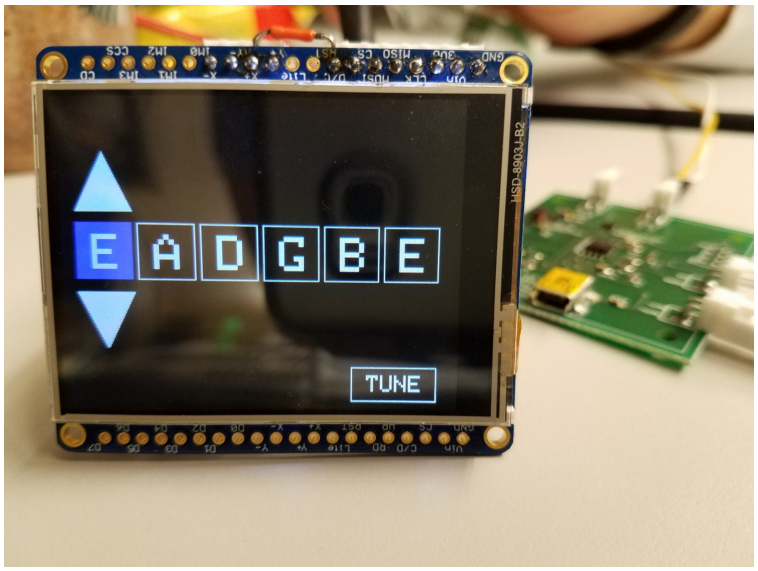

Figure 8: Touchscreen User Interface (not in tuning mode)

The touchscreen UI also provides additional information that is not included in the iPhone app, which makes it useful even if the user has chosen to select their tuning scheme and control tuning operation through the iPhone app: when in tuning mode, the LCD will display the progress of each note, providing real-time feedback to the user. For example, Figure 9 depicts the lowest string on the guitar being tuned to a D, with a red dot above the note to indicate that the string is currently sharp - this means that the motor is also currently turning such that the string is adjusting to a lower frequency. On the other hand, Figure 10 depicts the lowest string being tuned to an E, with a red dot below the note to indicate that the string is currently flat - this means that the motor is currently turning to adjust the string to a higher frequency. When a note is tuned correctly, the red dots disappear and the letter turns green. Figure 11 depicts the touchscreen once all 6 strings are in tune. Note that in all three of these cases, the touchscreen is clearly indicating that uTune is in tuning mode, by highlighting the "tune" button (in the lower righthand corner of the screen) blue. Additionally, the user may press the "stop" button at any point in the tuning sequence to exit tuning mode.

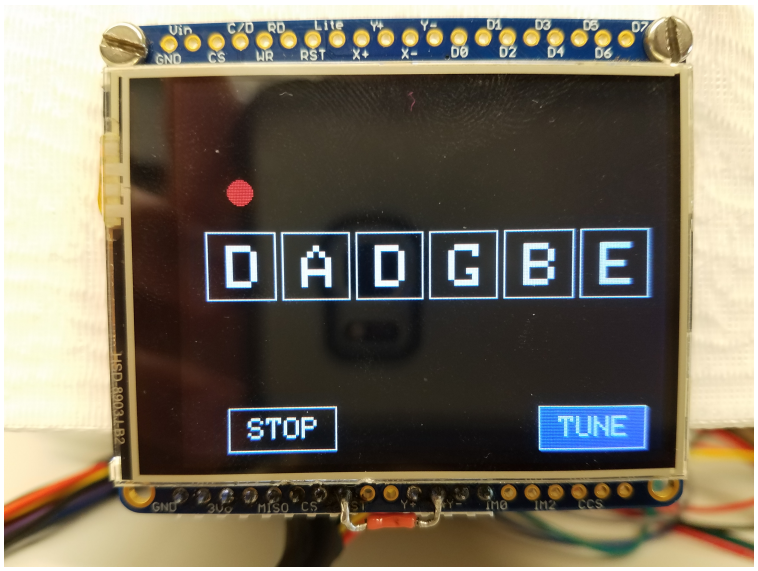

Figure 9: Touchscreen User Interface (tuning mode - sharp)

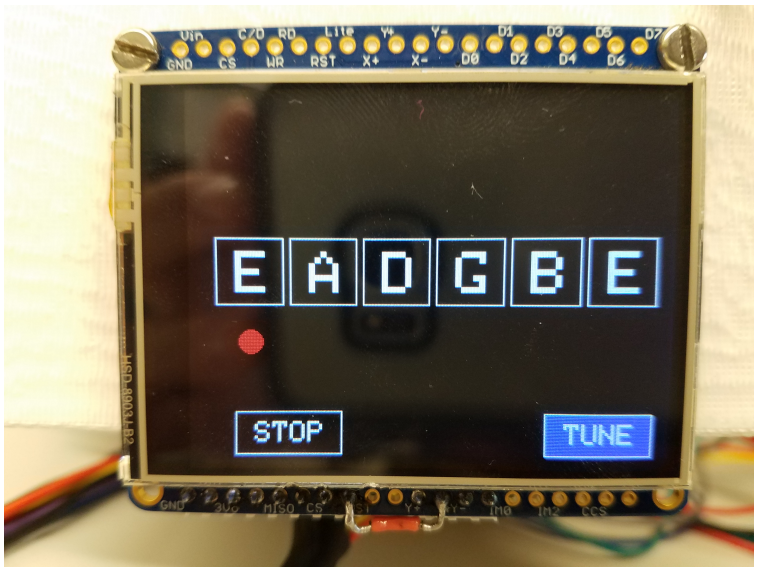

Figure 10: Touchscreen User Interface (tuning mode - flat)

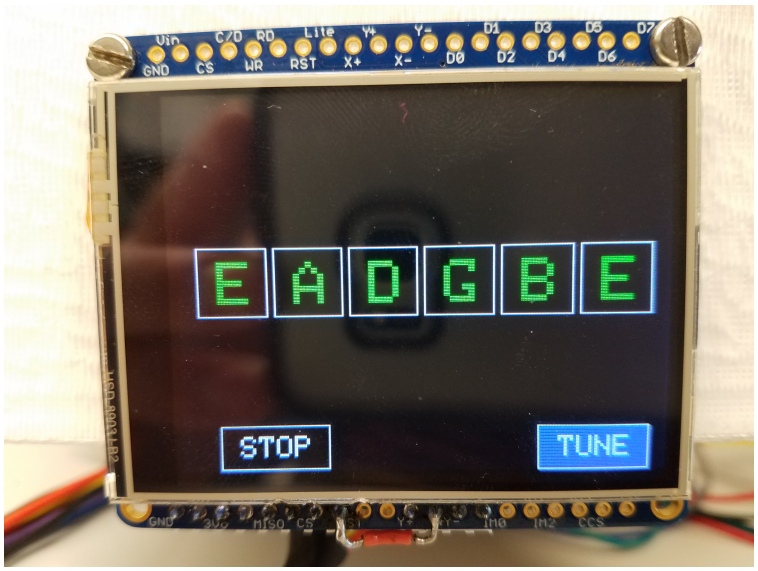

Figure 11: Touchscreen User Interface (tuning mode - fully tuned)

3.5 Audio Path Subsystem Design and Operation

The full audio path consisted of everything that took in the audio signal of the strummed guitar string as the input and output a number representing that string's current fundamental frequency. This consisted of two main components: the microphone component and the signal processing component.

3.5.1 Microphone

This component of the audio path subsystem is the component that took the audio signal of the strummed guitar string and brought the proper corresponding signal to the microcontroller's ADC input. In the tuning process, there must be some component that can assess the state of a string so that it can determine how the tuning procedure is progressing. This does not necessitate a microphone, as there are both physical and electrical pickup options which could have accomplished the same goal. However, the problem with these solutions is that they are not affordable (usually over one hundred dollars), and even if they would have been able to fit into the project budget, they would prevent a device such as uTune from ever being sold with consumer-friendly pricing. As an aside, many of these devices would also have eliminated the challenge of implementing an FFT to do the work of isolating the string frequencies, but that would have also eliminated one of the central elements of this project.

Consequently, a simple microphone eventually prevailed as the choice for the audio input sensor. There were many considerations in the process of choosing a specific microphone, but among the most important were operating voltage, frequency range, acoustic overload point, and sensitivity. Ultimately, the InvenSense ICS-40619 Analog MEMS Microphone was chosen because of its performance in these areas, and its data sheet can be found in Appendix B.3. This microphone's operating voltage is between 2.2V and 3.63V, which makes it well suited to connect to the power subsystem's 3.3V output. The frequency range of this microphone is 50 Hz to 20 kHz; since the lowest expected string frequency on the Low E would be a C2, or about 65 Hz, and 20 kHz is the top end of the audible frequency spectrum, this frequency range was very suitable for uTune's pusposes. The acoustic overload point of the microphone is 132 dB SPL, which is around the human audio threshold of pain. Since it is obvious that the guitar's loudness does not come close to the threshold of pain, this acoustic overload point is very suitable as well. Finally, based on the sensitivity of the microphone, the output analog microphone signal is around 10-20 mV for a standard guitar string pluck input. Since the microcontroller ADC takes an input of 0V-5V, the microphone output needed to be scaled properly.

To scale the microphone output, a differential amplifier was used because the microphone has a + and - output. Since the microphone output was about two orders of magnitude below the middle of expected ADC

input level range, a gain of 100 was chosen for the differential amplifier. A simple schematic of a differential amplifier can be found in Figure 12 below:

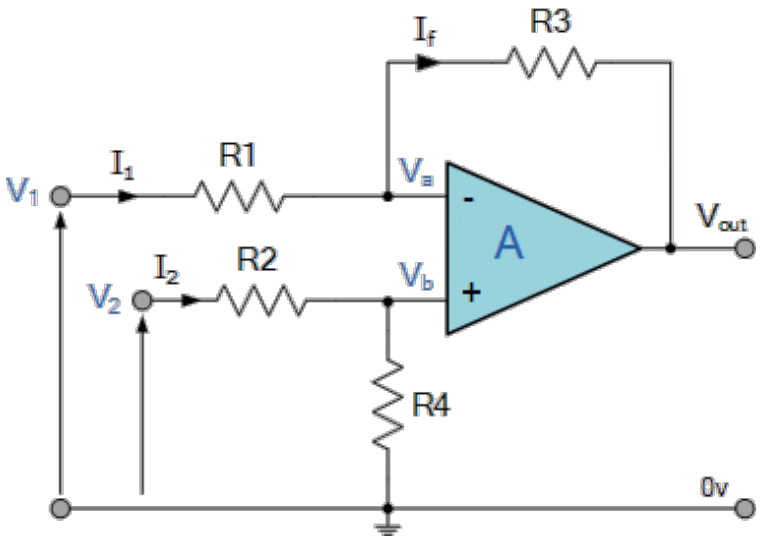

Figure 12: Differential Amplifier Schematic

The Texas Instruments TLV274 Op Amp was chosen, a standard op amp operable at 3.3V. To pick the proper resistor values for the differential amplifier circuit to yield a gain of 100, the equation of gain can be derived. First a voltage divider rule is used to find the expression for V_b in terms of V_2, R2, and R4:

$$V_b = V_2 \frac{R4}{R2 + R4} \tag{1}$$

Then, two more equations can be produced using Ohm's Law:

$$I_1 = \frac{V_1 - V_a}{R1} \tag{2}$$

$$I_f = \frac{V_a - V_{out}}{R3} \tag{3}$$

Since no current enters the op amp + or - inputs,

$$I_1 = I_f \tag{4}$$

so

$$\frac{V_1 - V_a}{R1} = \frac{V_a - V_{out}}{R3} \tag{5}$$

which rearranges to

$$V_a(R1 + R3) = R3V_1 + R1V_{out} \tag{6}$$

Since this is an op amp,

$$V_a = V_b \tag{7}$$

so

$$V_2 \frac{R4(R1 + R3)}{R2 + R4} = R3V_1 + R1V_{out} \tag{8}$$

which rearranges to

$$V_{out} = V_2 \frac{R4(R1+R3)}{R1(R2+R4)} - V_1 \frac{R3}{R1} \tag{9}$$

When $R1 = R2$ and $R3 = R4$,

$$V_{out} = V_2 \frac{R3(R1+R3)}{R1(R1+R3)} - V_1 \frac{R3}{R1} \tag{10}$$

so

$$\frac{V_{out}}{(V_2 - V_1)} = \frac{R3}{R1} \tag{11}$$

where the gain is equal to $\frac{R3}{R1}$. Thus, to have a gain of 100, the resistor values for $R1 = R2$ and $R3 = R4$ had to have a ratio of 100. The values of $R1 = R2 = 1k\Omega$ and $R3 = R4 = 100k\Omega$ were chosen for the differential amplifier circuit. The + output of the microphone was used as V_2 and the - output of the microphone was used for the V_1 output. The schematic for this differential amplifier circuit along with its subsequent PCB design can be found in Appendix A.3.

The measured voltage signal out of the differential amplifier circuit for a standard guitar string pluck was approximately a 2.2V, almost in the exact middle of the microcontroller ADC 0V-5V input range. This differential amplifier output was successfully read in by the microcontroller ADC, converted to a digital signal, and was ready to be processed.

3.5.2 Signal Processing

To process the audio signal, an FFT implementation for the dsPIC33FJ64GS606 was used. The FFT was run on a buffer of size 1024 that was the result of a 1.333kHz sampling rate, giving a resolution between frequency values of about 1.30Hz. This FFT of a string pluck contained peaks corresponding to either the fundamental frequency of the string or integer multiples of that fundamental, an example of which is shown below in Figure 13.

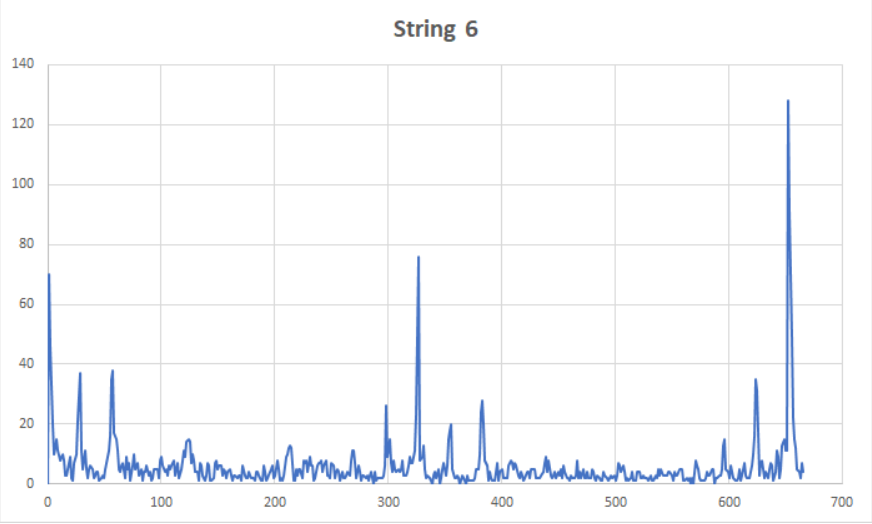

Figure 13: Example FFT of a single string

For each string, the same peak detection algorithm was used: first, the array of FFT values was sifted through to find the maximum points (corresponding to each peak shown in the graph). Each string has different characteristics that were able to be exploited for the algorithm; for instance, for the low E string, the highest peak corresponds to the frequency at twice the value of its fundamental frequency, while for the

low A string the highest peak corresponds to the frequency of its fundamental. Using these characteristics, which were found after taking dozens of data samples of each string and plotting the FFT for each, the location of the strongest peak in the FFT was tracked and checked to see if any peaks existed at frequencies that could be reached by dividing the most prominent peak's frequency value by an integer. If these were found, the lower peak was checked to be within a valid range for the type of multiplicity which it was (for instance, if a peak was found at 30Hz that fit as a multiple into 300Hz, it was known to throw that peak out since it is not within a valid range for a string). After this process was completed, the greatest common factor of the most prominent peak and the other peaks in the FFT is found, which corresponds to the fundamental frequency of the string.

3.6 Housing Apparatus Subsystem Design and Operation

Although this is an electrical engineering project, it was necessary to create a physical structure to hold all of the boards, motors, and other components in order to allow the device to physically tune the guitar. Since no preexisting products exist, and due to the weight constraints imposed by the design requirements, uTune was created by designing and 3D-printing the entire structure. This allowed for maximum customization ability and minimum weight, due to the plastic polymer used for printing. The full structure is shown in Figure 14, and will be discussed in greater detail below.

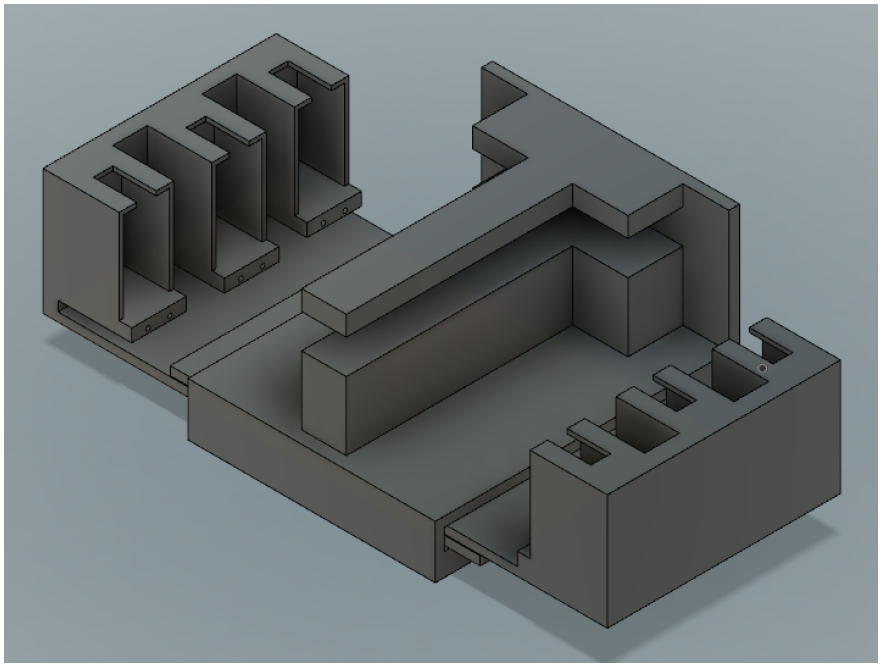

Figure 14: Final Full Design of the Housing Apparatus

The build consists of four main components: (a) an attachment to connect the motors to the guitar pegs, (b) housing for each individual motor, (c) a method of attaching uTune to the head of the guitar, (d) an overall structure to connect these components, allow the device to be easily removed/installed, and features to attach the touchscreen and PCBs.

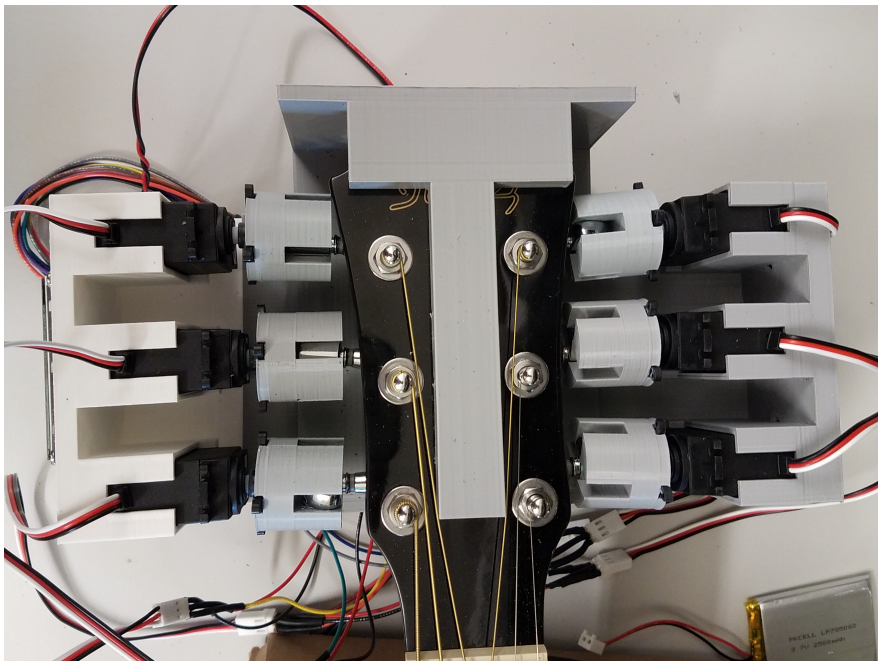

Figure 15: Full uTune Device Attached to Guitar Head

(a) In order for the motors to physically turn the pegs of the guitar, it was necessary to create a clip to physically connect the two. This was accomplished by designing the clip to neatly fit the 4-blade propellers that come with the servo motors. The propellers were superglued to the clips on the final device. Additionally, the clips were designed to minimize the amount of rotation required to fit the peg: by taking on the shape of a plus-sign, the maximum rotation required to align the peg with one of the slots is 45°. The inside edge of the clip was also curved in order to more snugly fit each clip, allowing the force to be applied closer to the end of the peg, reducing the amount of force required for a constant torque. The final peg design is shown in the following figures, where Figure 17 shows the side that fits the motor propeller, and Figure 16 shows the side that clips onto the guitar peg.

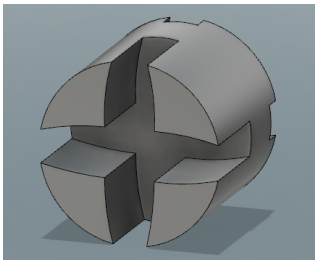

Figure 16: Front Side of the 3D-Printed Peg Clip

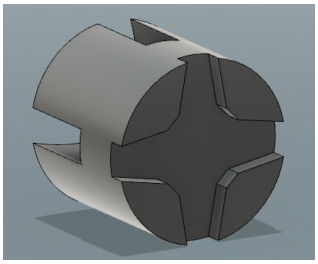

Figure 17: Back Side of the 3D-Printed Peg Clip

(b) To ensure the motor torque is applied to the pegs and not to the motor body, motor housings were created to firmly secure all six in place and prevent any movement. The motors slide snugly into the housing (which has a notch on the top for the wires), and then secure via two screws at the bottom. Due to the close fit and screws, the motors are secured to the main housing. Figure 18 depicts the 3D rendering, which clearly illustrates the size, wire notch, and screw holes. Additionally, Figure 19 shows the real product with the motors secured, and also depicts the utilization of the clips (see section a) to attach the motors to the guitar pegs.

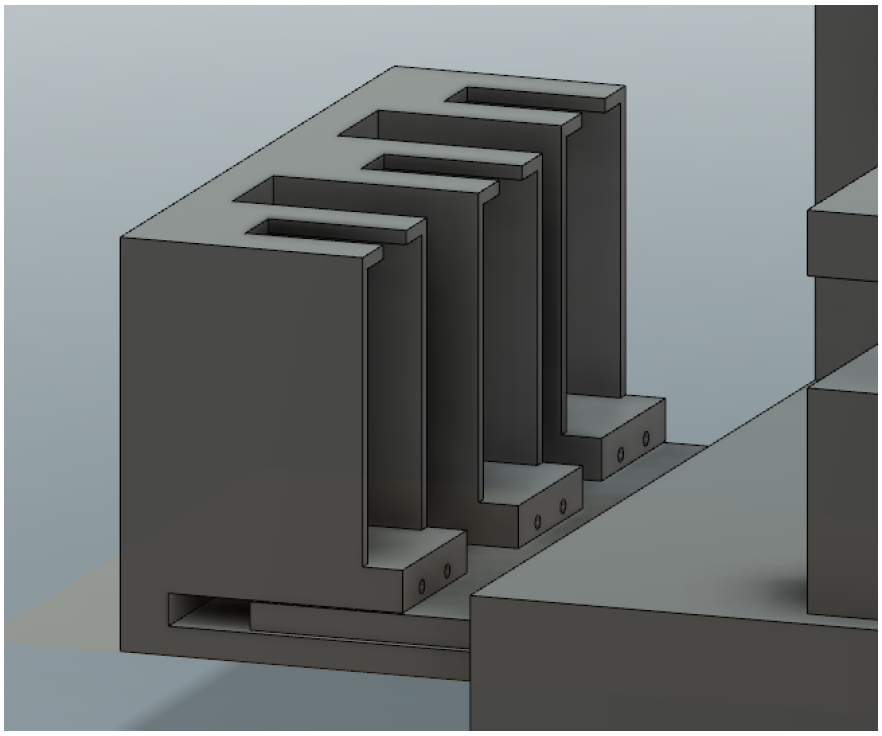

Figure 18: 3D Rendering of the Motor Housing

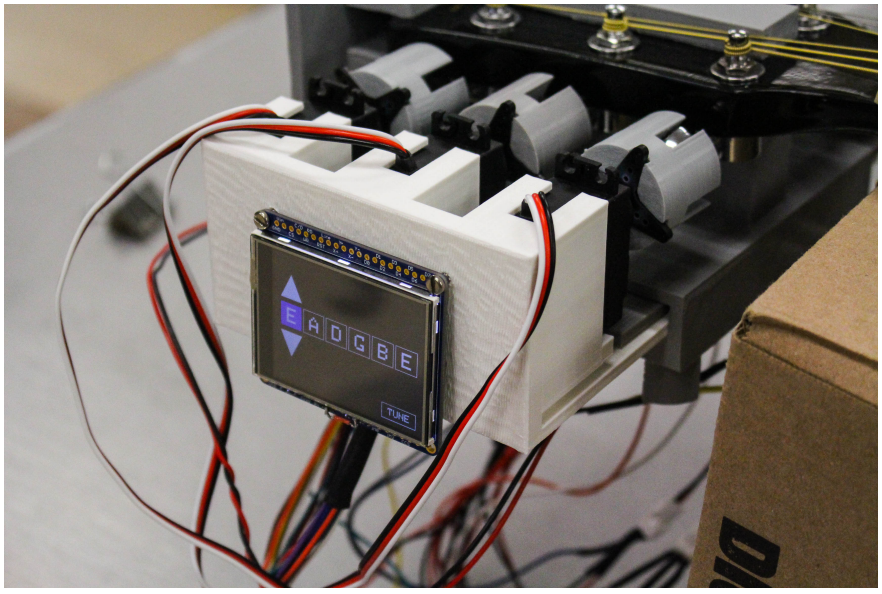

Figure 19: Real uTune Device, Zoomed into Motors and Peg Clips

(c) uTune is attached to the head of the guitar via a clip that slides directly over the top part of the head, down the middle in between where the strings attach to the pegs. This clip is clearly shown in the middle of Figure 14. Once this clip is secured to the guitar, the two side pieces slide together until the peg clips line up directly with the pegs. This sliding is illustrated in Figure 14, and more zoomed-in in Figure 18 (below the motor housing).

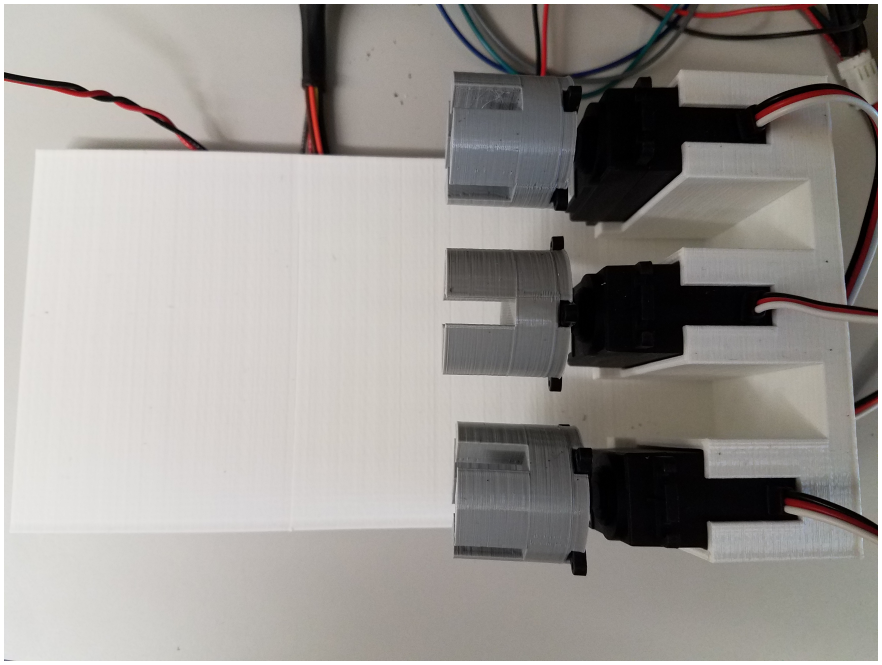

Figure 20: Close-up of Motor Housing and Peg Clips

(d) The full structure has been discussed previously, and as mentioned before, is depicted in Figure 14. Additionally, this model incorporates 10 additional screw-holes, which are utilized to attach various components onto the housing apparatus. On the left-hand side of the device, the touchscreen is attached via the two screw-holes shown in Figure 21. On the bottom of the device, the main PCB and power PCB are attached to the 2 sets of 4 holes, as shown in Figure 22.

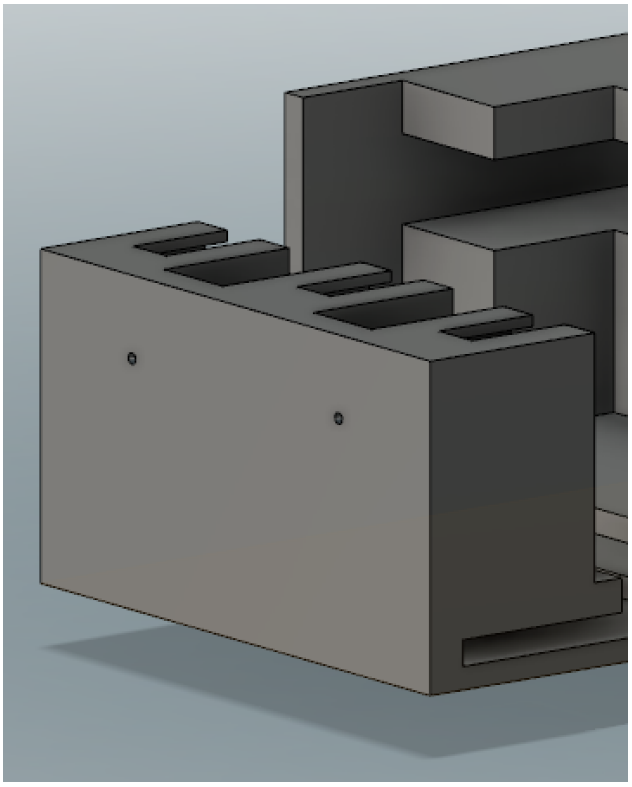

Figure 21: Depiction of the Two Screw-Holes for Securing the Touchscreen

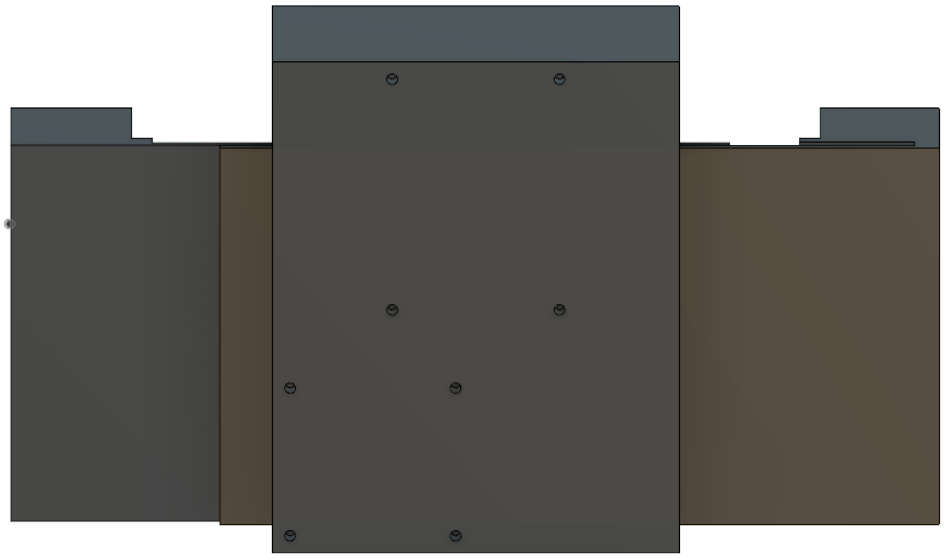

Figure 22: Depiction of the Screw-Holes for Securing the PCBs to the Bottom of the Housing Apparatus

3.7 iPhone App Subsystem Design and Operation

3.7.1 App Development

An additional aspect of this project was to create an iPhone App with the objective of smoothly allowing the user to create, select, and save tuning schemes from their pocket. This required the development of an iPhone app, which required learning an entirely new programming language and environment: Swift programming in the Xcode IDE. The core functionality objective of the app was originally simply to implement tuning schemes remotely from a handheld device, as it is widely known that app development can be quite cumbersome. Ultimately, however, the app was engineered to have higher functionalities such as tuning presets that persist when the app is closed and restarted as well as scroll bar boundaries to prevent users from selecting a tuning scheme that could potentially snap a guitar string.

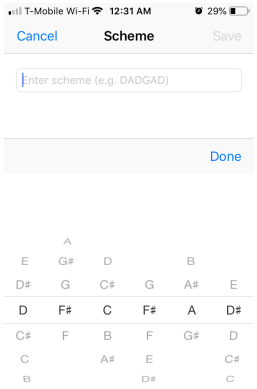

Figure 23: Scheme definition page in app

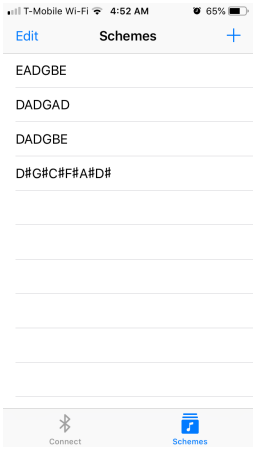

Figure 24: Add, edit, or delete schemes page

The app was created based on both tab and navigational views in Xcode; the user enters the app and can either select from a list of advertising uTune devices, or go straight to the library of preset tuning schemes to add, edit, or delete. Robustness was the app's main goal; if the user is not connected to a device, the button to initiate the tuning sequence is automatically hidden. Additionally, when the user selects a uTune device and connects, the app programatically moves to show tuning presets, avoiding the possible problem of a user attempting to reconnect after a connection has already been established.

The tuning preset page added another challenge, since the user should not be able to input invalid tuning schemes (within the allowable range). Therefore, a slide selector was implemented rather than a keyboard, adding visual ease and robustness for the user's scheme definition process. With this implementation, shown in Figure 23, the user cannot input invalid characters that will not be recognized by either the app's data structure or the dsPIC's decoding algorithm.

The development of the app also necessitated a specific data structure for each scheme, to allow for persistence when the app closed and to allow for accurate data transmission. For each scheme that was saved, a `name` and a `frequency` field is populated, with a string and double precision floating point value respectively. When the user presses the 'Tune' button when viewing a selected scheme, the app translates the `frequency` field of the scheme data structure into a string of 6 numbers, sending the data as a string over the BM70 Transparent UART connection to the dsPIC, where the string is decoded with space delimiters and read into a global variable.

3.7.2 Bluetooth Connectivity

Given the creation of an interactive iPhone app, some kind of communication protocol was required since the iPhone app would have to send data wirelessly to the dsPIC microcontroller. This connectivity would dictate the seamless functionality of the "tuning mode" of the device, so that the user would not be confused by non-communicating touch screen and app interfaces as well.

After some research, the best communication protocol for the project was determined to be Bluetooth Low Energy (BLE), as opposed to plain Bluetooth. This is a variant of the Bluetooth 4.0 protocol established in 2011 with an emphasis on minimizing power consumption. It accomplishes this goal primarily by remaining in a low-power standby mode when data is not being transmitted because as data rates have increased due to the implementation of transmission in higher frequency bands, less transmission time is needed to transmit small packets of data. The reason for this decision was primarily that when the task at hand is simplified, the only data transmission that needs to occur is sending six floating point numbers from the app to the device to indicate the six string frequencies. This is a remarkably small amount of data, and as a result, it makes sense to save power by allowing the Bluetooth chip to remain in a low-power standby state as much as possible. The BLE chip that was selected was the Microchip BM70, which provided a Transparent UART service that allowed the chip to function as a raw data pipe between the iPhone app and the dsPIC. This allowed for greater development speed and the avoidance of having to learn GATT protocol in BLE.

The user connects to a uTune device through the app by searching for the correct advertising Universally Unique Identifier (UUID); a user-defined, new UUID was chosen for uTune such that only uTune devices show up in the list of connectable devices on the app. To set this up, the BM70 was programmed to advertise a `0xFFF1` UUID and the app to specifically search for a `0xFFF1` UUID.

As discussed above in Section 3.7.1, when the user hits the 'Tune' button while viewing a scheme and connected to a device on the app, the app sends a string of 6 floating points numbers through UART from the BM70 to the dsPIC, where a global array containing the `double` values of the 6 target frequencies is stored. The dsPIC's main execution loop constantly looks at the values of this array with each iteration; if there are values stored in the global frequencies array, that means the user has either selected tuning mode from the touchscreen or from the app, and the PWM should be enabled. If the user wishes to exit tuning mode, however, he/she may push the 'Done' button on the app (as well as the 'Done' button on the touchscreen). When 'Done' is pushed on the app, a string of zeros formatted correctly for dsPIC interpretation is sent across UART from the app through the BM70, which is then read into the global frequency array on the dsPIC. Once this UART transmission is decoded, the next iteration of the main function on the dsPIC will see that the global variable array is filled with zeros for frequencies and the device will disable its PWM outputs and exit tuning mode.

3.8 Motor Subsystem Design and Operation

The motors chosen were continuous servos, which operate from a 5V power connection, ground connection, and digital PWM input. For the chosen motors (and for most continuous servos), the PWM input must be a 50Hz square wave with a very specific duty cycle.

For non-continuous servos, usually the duty cycle of the PWM input determines the position to which the motor will turn, with a degree mapping to a specific duty cycle. For a continuous servo, however, the duty cycle corresponds to the speed at which the motor operates. In a 50Hz square wave, the period is 20ms. For the chosen servos, to make the motor turn counter-clockwise the 'on' time of the duty cycle had to be between 1.2 and 1.4 milliseconds. Conversely, to turn the motor clockwise, the 'on' time of the duty cycle had to be between 1.6 and 1.8 milliseconds. Finally, to pause the motor, the duty cycle is set to either 0 or as close to a 1.5 millisecond 'on' time as possible.

To design the PWM's, the high-speed PWM module on the dsPIC was used. The module runs off the auxiliary clock, which scales down the main FRC with PLL clock from 40MHz to a 50Hz range. Based on input from the audio path, if the measured frequency necessitated some motor motion with regards to getting to the target frequency, the corresponding PDCx or SDCx register could be set, which set the 'on' time of the corresponding PWM and turned the correct motor the correct direction.

For uTune's design, the motor speed was held constant but fine tuning was achieved through proportionally decreasing the time the motor was active as the measured frequency approached the target. So, if the measured frequency was very far from the target, the motors would initially turn for a long time (up to a second) before audio sampling began a second time, at which point the time the motors would be turned on to approach the target frequency would be shortened considerably to avoid overshoot.

3.9 Interfaces

As has been discussed in previous sections, the dsPIC33FJ64GS606 Digital Signal Controller serves as the epicenter of the device, by taking inputs from a number of external sources, running computations and algorithms, and outputting signals to a number of external devices as well. This microcontroller is powered with a 3.3V supply from the power board, which converts a 3.7V Li+ battery into 3.3V and 5V outputs and sends power to the microcontroller via a jumper cable. The power board outputs 5V to the six motors with sufficient current capability.

The dsPIC communicates with the BM70 Bluetooth module via UART, the traces for which are included on the main PCB. This initializes and runs uTune's BLE capabilities such that the user is able to connect to the device via their iPhone. Once the Bluetooth connection is established and the user has selected a tuning scheme on the app, that data is sent to the BM70 and then input into the microcontroller via the UART connection.

To operate the touchscreen, a number of different connections are required. The touchscreen is programmed by and directly interfaced with the dsPIC via a SPI connection. The dsPIC sends display data to the LCD and receives data from the four output pins on the resistive touchscreen. The dsPIC then runs a number of algorithms to convert the touch data into display coordinates, verify the intended effect of the touch (i.e. is it within a number of recognized boxes), and output appropriate changes to the display. Additionally, the touchscreen can be utilized to enter "tuning mode," which can also be done via the app.

Once in tuning mode, the user strums a single string to initiate the tuning sequence. The audio data is received by the microphone, which passes the data through a differential amplifier to boost the signal and attempt to reduce some background noise. From there, the amplified signal is sent to the dsPIC via a long MOLEX-connected wire. The dsPIC then runs the FFT algorithms to find the frequency peaks and determine the fundamental frequency, and thus the string that was plucked. The fundamental frequency is then compared with the intended/selected frequency, and the dsPIC determines for what length of time and in which direction the servo motor should turn. It then outputs a command via a PWM pin to the appropriate servo, which, powered by the 5V board supply, turns the peg of the guitar in the correct direction. At this time, the dsPIC will also update the touchscreen display to indicate to the user whether the current string is sharp, flat, or in tune. When all six strings are fully in tune, indicated by all six letters turning green on the touchscreen, the user should exit tuning mode and play with their in-tune device.

4 System Integration Testing

With any consumer electronic product, integration testing can be a massive task. The approach that the uTune team took to integrate all of the subsystems involved approximately four concurrent integration processes that met at the end of the project. Then, of course, the final task was assembling each of these sub-assemblies of the project at once to integrate every subsystem into the full system. These processes were:

1. Successfully convert the microphone input signal to an identified frequency; that is, prove that the system can interpret what note is currently played.
2. Establish rigorous connectivity between the iPhone app and Bluetooth module, eventually coordinating system states with the touch screen as well.
3. Perfect the power system such that it will provide reliable power to the main PCB, and then the other external PCBs while charging or operating from battery power.
4. Implement the motor PWMs with a control algorithm to execute the tuning loop.

4.1 Integration of Audio Path

The system integration process began with the microphone and understanding its signal profile. This required understanding the type of digital data that could be extracted from the microphone. As a first step to achieve this goal, a simple mounting board for the ICS-40619 microphone was manufactured, and this breakout board was breadboarded with an Arduino Uno to read a constant serial stream of 10-bit digital input data in the Arduino IDE console. Unfortunately, there was almost zero dynamic variation of the digital value read by the Arduino. Upon further investigation, however, it was determined that the reason for this was that the bit resolution of a 10-bit ADC running at 5V does not allow for much dynamic variation when signal perturbations are around a few millivolts. Despite having both positive and inverted signals available from the microphone centered around 2.2V, this would not be enough to define frequency peaks from an FFT. Upon consulting the datasheet, this outcome made logical sense given that the microphone was being operated in "high-power mode" at a 3.3V power voltage, and the digital values corresponded to the serial port display on the Arduino. Therefore, in order to increase the signal level, a Texas Instruments TLT272 differential amplifier with precision resistors was used. With one-percent resistor tolerances and by selecting the proper matching resistance values, it was shown that the common-mode rejection ratio (CMRR) would remain high enough for the purposes of this project. Therefore, only the difference of the signal from the small signal output of the positive and negative strings would be amplified and sent to the dsPIC microcontroller.

With an appropriate audio signal reaching the ADC, the sampling rate was adjusted by adapting the system clock using the Timer 1 interrupt such that the highest frequency interpreted digitally would be the second harmonic of the high E string (the highest note to which any string of the guitar would ever be tuned). The FFT is a well-established algorithm, so once a sample was taken, it was a matter of implementing it as a "black box" with an array of the sample values inputted and an array of the frequency spectrum outputted. However, this ran into a memory issue, with the array's maximum length being set to 1024, as this was the maximum available storage on the dsPIC microcontroller that was used for this project. This limited the frequency resolution to approximately 1.3Hz when a sampling frequency of 1.333kHz was used. This proved to be adequate to tune a guitar within proper ear-measured tolerance, but in future projects, external memory would be advisable. One technique explored to address this issue was zero-padding the sample before taking the FFT. However, this simply introduced more memory issues and did not increase the resolution of the data, but merely smoothed the FFT graph to make it more readable (something not needed by the processor). Zero-padding is not used for periodic signals like the audio signal of the string pluck.

Once the FFT spectrum array was obtained and when graphed confirmed the presence of the original audio signal as expected (compared to what was observed when the microphone was connected to an oscilloscope), the peak identification algorithm was deployed. This algorithm analyzes the array to find which frequency bins contained large values compared to a threshold, and then uses a series of common integer ratios to compare to the ratios of the frequency components, which eventually identifies the greatest common divisor, which is the fundamental frequency of the string being plucked.

At this point, it was possible to obtain a single floating point value for the fundamental frequency of the string being plucked from a microphone signal input. This sequence was then implemented in conjunction with "tuning mode" in the UI subassembly outlined below.

4.2 iPhone, Bluetooth, and Touch Screen

The first step in integrating the iPhone, Bluetooth, and touchscreen was to begin work in the Xcode IDE. With the simulator tool built in to Xcode, code could be added to the Swift compilation and tested on a dummy app in real time. Additionally, code could be uploaded to an iPhone to test how things appeared on different screen sizes.

To start integration of the Bluetooth module, the BM-70-CDB was purchased from Microchip, which is a development board that contains the BM70 chip that would be used in uTune's final board. The BM-70-CDB came with pre-packaged firmware that allowed it to work with the Microchip Bluetooth SmartData app. This app allowed the user to type in a string of text on the iPhone and, after opening a terminal in PuTTY corresponding to the COM port of the BM-70-CDB, see the text show up in the terminal emulator after hitting 'Send'. Additionally, the BM-70-CDB caught any keystrokes from the keyboard, displaying each character on the app. This demonstrated the device's discoverability and connectability, as well as its demonstrated purpose; the board could demonstrate transmission of simple information with the touch of a button from an iPhone app with the BM70.

Additionally, since the BM-70-CDB had header pins leading to the U1RX and U1TX pins on the UART module, it could be verified that the UART signals were being received and transmmitted correctly from the iPhone using a Salae Logic Analyzer.

The next step was introducing Bluetooth connectivity on the iPhone app. The first step was demonstrating connectability to the development board, and writing the correct Swift code to send data over the Bluetooth Tx characteristic. At that point, since the BM70 comes packaged with a Transparent UART Characteristic, it acts as a simple data pipe from the iPhone to the attached MCU. Once writing a simple string from our iPhone app to the PuTTY terminal emulator was demonstrated, it was known that our data was being received on the Rx pin of the UART connected to the BM70.

In the app code, each frequency corresponding to the different strings was defined to either 1 or 2 decimal places; if the frequency was below 100Hz, it was defined to 2 decimal places, and was defined to 1 elsewhere. This was so that, when the BM70 sends the frequency encoding from the iPhone to the MCU as a string, the MCU knows exactly how many transmissions to look for between each space delimiter (5 transmissions; 4 digits and 1 decimal point character). A UART interrupt was set up on the dsPIC that decodes all six frequencies when one interrupt flag is raised on the UART pins, allowing the user to continually check the values in a global variable array of selected frequencies to determine whether the device is in tuning mode (nonzero entries) or settings mode (zero entries).

To test that this process was working correctly, the dsPIC was run in debug mode with a breakpoint set up after the decoding function in the UART ISR. In order to reach this breakpoint, the dsPIC would have to correctly decode the string incoming from the BM70 upon sending, and store the values in the global frequencies array. It was shown that, each time 'Tune' or 'Done' was hit on the iPhone app and sent the data, the breakpoint was reached, verifying that the correct baud rate of 115200 was set and that the interrupt functions were decoding correctly.

To integrate touch screen capability with our full system, the touch screen was connected to the Arduino with a pre-made code that draws lines when the user touches the background. The success of this code demonstrated both the success of the display functionality of the touchscreen as well as the touch functionality, since the user could draw lines with touch. Next, a simple UI was designed that incorporated the six string selections with arrows and a tune button, again coded for Arduino. Using Arduino provided an easier baseline for getting code to run on each systems, so the UI could be easily tuned and tweaked based on what was desired from the screen. Once the Arduino code was satisfactory to be moved to the dsPIC, the screen was shown to respond on the dsPIC to rudimentary SPI commands, focusing first on the display capability. Adafruit libraries were consulted, changing some functions to be compatible with the SPI module on the dsPIC, eventually getting to a point where screen color could be dictated and it was demonstrated that SPI commands were being interpreted correctly by the screen.

Next, the command set was built to reconstruct the Arduino UI on the dsPIC, translating the C++

functions from Arduino to C and SPI for the dsPIC until a replica of the UI built in C++ for Arduino existed.

Next, the goal of providing touch functionality needed to be achieved as described above in Section 3.4. This process was very difficult; the touch capability on the screen is enabled by two analog inputs and digital outputs that switch functionality within our code, so care was needed in setting the configuration of each of the pins. The voltage on the output analog pins of the touchscreen was tested with a multimeter while pressing on different points of the screen, determining a voltage mapping that corresponded to points on an x-y plane describing the screen's area. From this, and from similar tests on the threshold of pressure necessary to generate a high enough voltage to be considered a "touch," functions were created to read x and y values for touch location as well as a z value for touch pressure.

Finally, to ensure touch capability had real use on our screen's UI, each coordinate from the touch output was mapped to its proper location on the UI, determining which display buttons were pressed from each x, y, and z reading from the analog outputs.

The touchscreen integrated with the master uTune code through the managing of a master array that sorts through which notes are selected and maps to frequencies to be stored in the global frequencies array when the 'Tune' button is pressed on the screen. Since the screen was being controlled by the dsPIC, the storing of frequency variables did not have to be interrupt-driven in our code; if a touch within the region defined by the 'Tune' button was read, the current values in the master array given by the notes on the screen were simply written into the global frequencies array, one-by-one. In doing this, the main execution loop could also enter and exit tuning mode just by checking that there were nonzero values within the global frequencies array.

4.3 Power

Once the components were selected and a board designed for the power subsystem, the two primary objectives were to achieve battery charging functionality, then to prove that the selected converters would provide sufficient power from the charged battery at peak draw, and then finally test the operation of the motors with the power supply. At first, when the power board was ordered, the simple approach of populating the board, inspecting each solder connection to eliminate any undesired bridges, then plugging in and testing was unsuccessful. The 5V converter was overheating, so on a new copy of the board, it was omitted, and it was discovered that the battery charging feature was functional, as the status LED remained lit for the duration of the USB charging cycle, then eventually went out when the battery was fully charged. The full charge was verified with the ttEnergy LiPo charging device. Additionally, 3.3V power was measured with a multimeter both while in charging mode and while only battery power was functional. This 3.3V output performed according to the component datasheet, dropping about 0.2V as the current demand exceeded 100mA.

With full functionality of the 3.3V system power, the issue remained the 5V system power. With the 5V converter chip absent from the power board, the traces on the board were analyzed, and it was confirmed that the 5V chip's VCC pin had been connected to GND by a trace instead of to the enable pin. Therefore, with the current board design, the chip would not be able to be installed properly. Before a modified board design was ordered, the traces around the XC9141B50CMR-G 5V boost converter were cut and rewired such that the chip could be tested under the intended operating conditions. After doing so, the suspected issue was isolated and confirmed, and the proper modifications made to order a new power PCB. Once this modified board arrived, the 5V converter was tested to confirm that it could output up to its rated 800mA peak power draw to operate all of the motors simultaneously. It was capable of doing so with minimal voltage drop (0.4V), which would only ever occur for a fraction of a second, so the power board was fully operational, and with the proper Molex connectors, proved to successfully power both the main PCB and the microphone PCB.

4.4 Motors and PWM

The goal for the motors was to have enough torque to easily be able to turn a guitar peg. To test this, a simple button-driven forward/backward motor setup on Arduino was first constructed, with a 3D printed a motor head attachment to fit a peg. When the motor was held up to the guitar, it was found that given

a PWM with a duty cycle sufficiently far from 1.5ms on (that is, with enough speed to provide sufficient torque), the motor could easily tune each peg, including the toughest case of tuning the high E string *upward*.

After verifying motor strength, a logic analyzer was used to find the PWM duty cycle requirements for the continuous servos by observing the digital output of the Arduino that was operating the PWM. From this, it was verified that a 50Hz PWM was needed with specific duty cycles as described in Section 3.8 above.

With this knowledge, our auxiliary PWM clock could be tuned on the dsPIC to set a simple constant output program with the PWM module at a target of 50Hz, in order to check that a high-fidelity PWM for our continuous servo could be generated. Once the motors were turning with sufficient torque (which was a simple constant speed), simple proportional logic was implemented to turn each motor on for a set amount of time based on the difference between the measured frequency from the audio path and the target frequency as specified by the user for a specific string.

4.5 Full System Integration

At this point, the number of smaller subsystems were ready to be integrated within the confines of the final physical apparatus. After the final 3D print of the apparatus was ready, the touch screen was installed on the side part such that it would be visible to a guitarist looking down on the guitar. Next, the motors were installed in their sockets using a strong adhesive. Finally, the power board and main PCB were screwed onto the backside of the uTune device. Custom Molex connectors were made for each of the sets of connections including six power/ground/PWM connections for the motors, a 10-pin and a 4-pin connector for the touch screen, a long 4-pin connector for the microphone board to extend to the hollow body of the guitar, and power distribution cables from the power board.

Aside from the physical integration, additional software was needed to ensure a smooth transition of states. While the audio path could identify a fundamental frequency, the touch screen and app could send user-specified desired note frequencies, and the motor PWMs were controllable by the microcontroller, these needed to be executed in an organized fashion. This required the implementation of "tuning mode" which continuously runs the feedback cycle of listening, identifying the fundamental, and turning the motors for an amount of time proportional to the difference between the observed and desired frequency in the proper direction. Likewise, the ability to exit tuning mode implemented a way for the user to disable the motors while playing a song and leaving uTune attached.

4.6 System Requirements Evaluation

In this section, each of the system requirements outlined in Section 2 will be evaluated according to the performance of the final uTune prototype.

4.6.1 Motors

Torque: Each motor is capable of providing sufficient torque to turn a guitar peg, with string attached, to three notes above its standard tuning pitch (e.g. must be able to turn E string up to a G).

Speed: Each motor can complete a full rotation in approximately two seconds - less than the required ten seconds.

Weight: The guitarist is capable of carrying the guitar with the uTune device clipped. Each motor weighs 38g, which is less than the 75g requirement.

Accuracy: The motors are capable of turning in small increments, but not necessarily as small as 0.5 degrees. This requirement was not explicitly met; however, the motors are capable of turning in small enough increments to achieve the 1.33Hz resolution allowed by the FFT.

Current: Each motor draws a peak current of approximately 150mA, much less than the 1A requirement.

Attachment: Each attachment connects securely to the motor and peg and requires no at most a $\frac{1}{4}$

turn of the peg when attaching the uTune apparatus to the guitar.

Cost: Each motor and connector combination must cost approximately $6.00 when considering the $4.55 cost (currently minimum available quote) of the motor and the small price of 3D printed material, which would certainly be lower if it was mass manufactured instead.

4.6.2 Batteries

Power: uTune can run completely from battery power. The battery can to power 6 motors in tandem, the LCD screen, microphones, wireless chip, and microcontroller.

Charge: The battery is easily rechargeable and can be replaced easily if its capacity is diminished over its lifetime or if there is a malfunction.

Weight: The specified battery weighs 22g, and similar-capacity batteries weigh similar amounts.

Cost: The specified battery costs $9.95 plus shipping, so it is more expensive than the required $5.00.

Time: At least 100 full re-tune cycles can be completed on a single charge. The rechargeable battery will last for significantly longer than 1000 tunes over its lifetime. The device can remain powered on for a period of time longer than a performance, but not necessarily 6 hours.

4.6.3 Microcontroller

Abilities: The microcontroller is capable of:
-reading analog microphone signals, converting them to digital data, and performing frequency analysis (i.e. the FFT), then running an algorithm to find local peaks and current string frequency
-interfacing with mobile app and LCD screen to determine user-selected string frequencies
-deciding based upon user selection and current string frequency how to control the motors
-sending a PWM signal to turn a specific motor such that it turns the guitar peg according to the user's input

Size: The microcontroller and accompanying circuit components are not smaller than the guitar head, but they can be feasibly mounted on the uTune apparatus.

Weight: The microcontroller and board assembly weigh under 70g.

Cost: The microcontroller used costs well under $20.00.

4.6.4 Physical Build

Weight: The guitarist can comfortably support the guitar and use the tuner setup while playing if he/she uses a guitar strap. However, uTune weighs more than 500g.

Assembly: The uTune device can be easily put on or removed in under 1 minute.

Robustness: The assembly is made of robust materials that usually are durable enough to last at least two years of frequent use.

Cost: The assembly for the device (not including the boards and components) can cost less than $30.00 if it is sourced from an affordable 3D printing service, or an in-house printer.

4.6.5 LCD

Size: The LCD screen is large enough to display 6 characters for the tuning selection, the accompanying arrows,
and the tuning mode buttons. It is smaller than the guitar head.

Weight: The LCD screen weighs 30.5g, which is less than the 50g requirement.

Visibility: The LCD screen has much higher resolution than 8 pixels per character.

Buttons: Buttons were not necessary since a screen with touch functionality was used instead.

Cost: The LCD screen must costs approximately $27.50 (plus shipping) which is over budget.

4.6.6 Mobile App

Devices: The app can successfully maintain a connection with the uTune device.

Range: The tuner can connect to the mobile device at distances beyond 10 feet.

Bluetooth Chip: The bluetooth chip is low-energy (BLE), smaller than 2 square centimeters, and easy to program and interface with the microprocessor.

4.6.7 Frequency Detection System

Size: The frequency detection system (the microphone PCB) is larger than 1cm x 1cm, and it must be placed in the body of the guitar rather than a more convenient location.

Weight: The microphone PCB only weighs a few grams.

Placement: The microphone must be placed separately from the rest of the assembly, so this requirement was not met.

Quantity: Only one microphone was necessary for this project, which is less than the required maximum of six.

Cost: The microphone board unit cost (depending on quantity ordered) could easily be under $20.

4.6.8 Safety

Exposed electricity: Currently, there are many exposed wires, so the user should have caution during operation not to accidentally short any connections or mess up any wires.

Heat: The component most at risk of overheating and fire is the battery, and this component is regulated with a thermistor-enabled fault system.

Shut-off: The device has built-in abort sequences if strange data is coming from the microphone (i.e. multiple strings being strummed at once), but it does not prevent accidental tuning above the specified range if the user does not start the guitar in the proper tuning.

5 User's Manual

This section will outline the proper installation, use, and troubleshooting for the uTune prototype. These should serve as a first resource for a new user's setup, operation, and potential troubleshooting.

5.1 Installation

Before Installation

Make sure that the six strings are tuned within their reasonably expected range, as listed in Table 3 below:

Table 3: Allowable Note Range For Each String

String Name	**String Note Range**
Low E	C2-E2
A	F2-A2
D	A#2-D3
G	D#3-G3
B	G#3-B3
High E	C4-E4

uTune Components:

The uTune structure is comprised of the following separable physical pieces: center structure (Figure 25), left structure (Figure 26), and right structure (Figure 27).

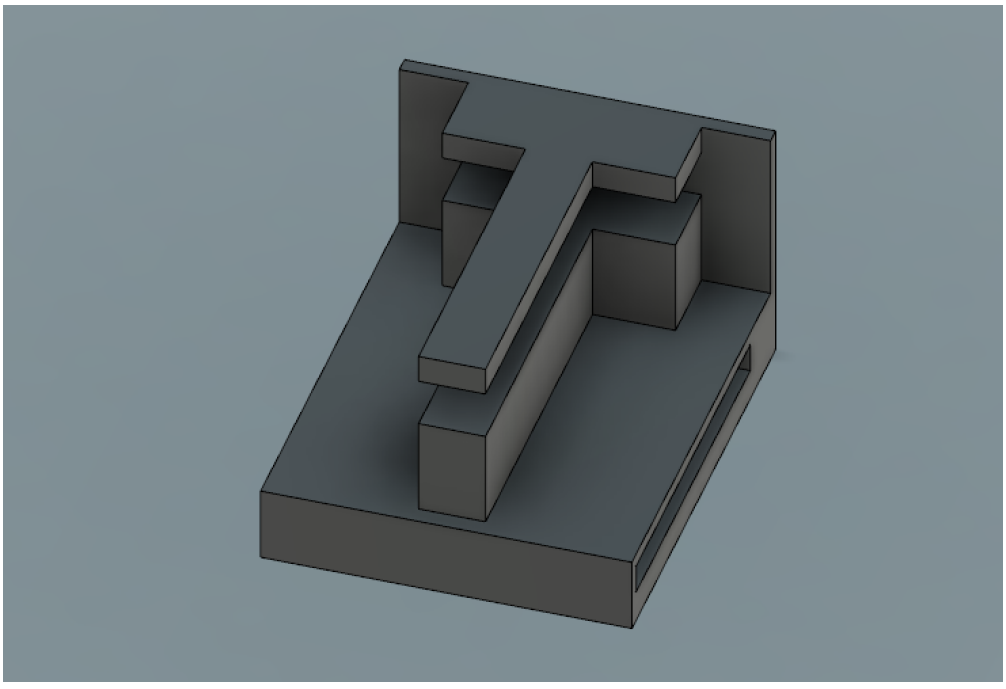

Figure 25: Center piece of uTune device

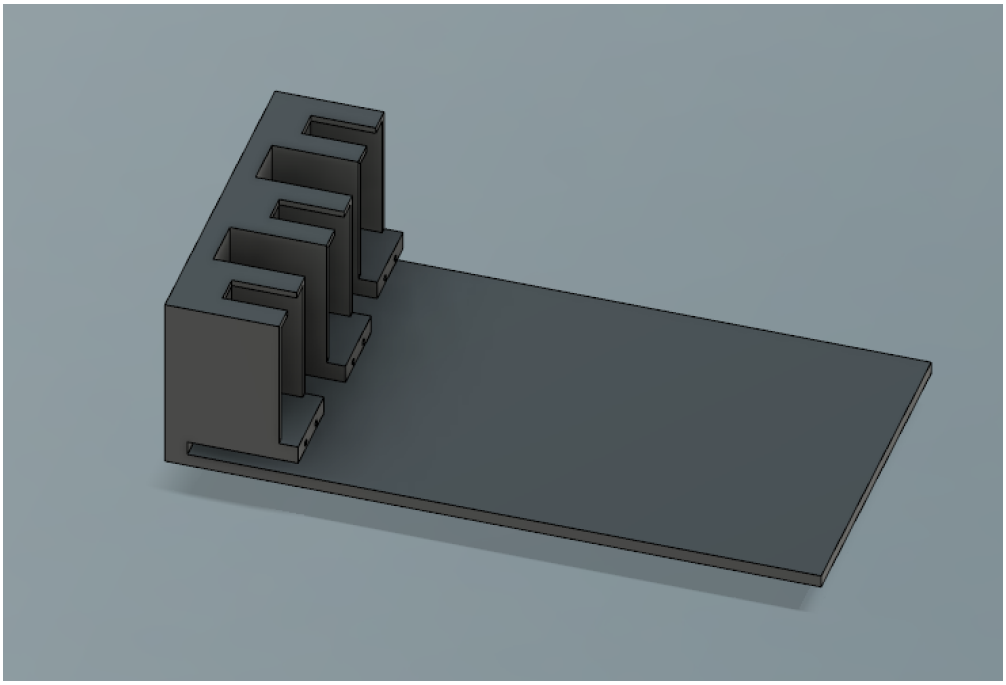

Figure 26: Left piece of uTune device

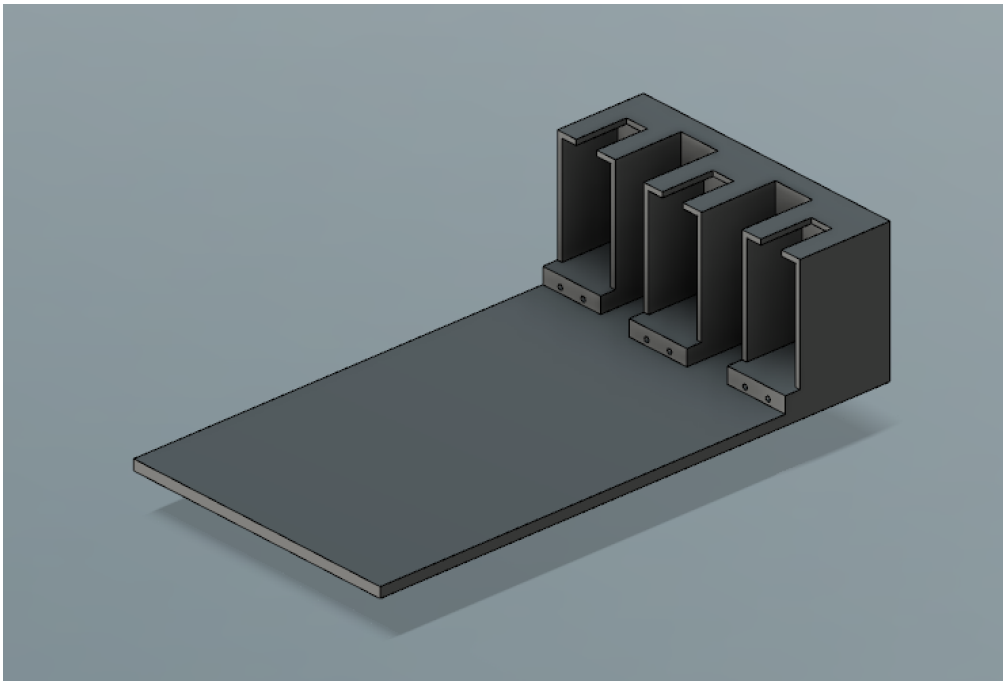

Figure 27: Right piece of uTune device

To install device:
1. Slide left and right structures (see Figure 26 and Figure 27) away from each other.
2. Slide center structure onto head of guitar (see Figure 25).
3. Turn motors on left and right structures to line up with guitar tuning pegs.
4. Slide left and right structures toward each other.

5. Power on device by plugging in battery.
6. Click dsPIC reset button on board underneath center structure.

5.2 Touch Screen Use

There are two uTune modes: Settings mode and Tuning mode.
uTune starts in Settings mode.

Setting tuning schemes:
1. Touch box on touch screen corresponding to string.
   [From left to right, the string boxes are for: Low E, A, D, G, B, High E]
2. Touch up/down arrows to select desired note.
3. Repeat (1)-(2) for other strings.

Tuning guitar:
1. In Settings mode, press 'uTune' button on bottom right corner of screen to exit Settings mode and enter Tuning mode.
   [WARNING: DO NOT PLAY MORE THAN ONE STRING AT A TIME IN TUNING MODE. This will cause the device to be confused and go haywire, which could cause harm to strings.]
2. Pick a string. Pluck the string repeatedly with thumb downward for best results. If out of tune, a red dot will appear by its box on the touch screen. If above the box, the string is sharp compared to its desired note. If below the box, the string is flat compared to its desired note. Once note letter in box turns green, string is in tune.
3. Repeat (2) for all other strings needing tuning.
4. Press the 'Done' button to exit Tuning mode and enter Settings mode.

5.3 App Use

Setting up app connection:
1. Open app by clicking App icon.
2. Make sure iPhone Bluetooth is on.
3. At the bottom of the App, click the 'Connect' tab on the left.
4. Click uTune device to connect over Bluetooth.

Creating tuning schemes:
1. At the bottom of the App, click 'Schemes' tab on the right.
2. Click '+' in the top right corner.
3. Scroll the six wheels at the bottom of the screen to set the tuning of each of the strings.
   [From left to right, the string wheels are for: Low E, A, D, G, B, High E]
4. When finished, click 'Done' and 'Save' to save the tuning scheme, or click 'Cancel' to cancel this tuning scheme.

Deleting tuning schemes:
1. At the bottom of the App, click 'Schemes' tab on the right.
2. Click 'Edit' in the top left corner.
3. Click red circle next to an existing tuning scheme to delete it.

Tuning guitar:
1. Connect to a device. The app will programatically switch to the 'Schemes' tab view.
2. Click on tuning scheme of choice from list of saved tuning schemes.
3. Click 'uTune' in the middle to exit Settings mode and enter Tuning mode.
   [WARNING: DO NOT PLAY MORE THAN ONE STRING AT A TIME IN TUNING MODE. This will cause the device to become unpredictable, which could cause harm to strings.]
4. Pick a string. Pluck the string repeatedly with thumb downward for best results. If out of tune, a red

dot will appear by its box on the touch screen. If above the box, the string is sharp compared to its desired note. If below the box, the string is flat compared to its desired note. Once note letter in box turns green, string is in tune.

5. Repeat (4) for all other strings needing tuning.
6. Press the 'Done' button to exit Tuning mode and enter Settings mode.

5.4 Troubleshooting

If the device appears to be frozen:

1. Try resetting the dsPIC by pressing the reset button included on the PCB.

If the App is having difficulty connecting to the device or the device is not properly responding the instructions sent by the App:

Try resetting the BM70 module:

1. Remove the RST_N jumper.
2. Briefly connect the RST_N jumper to the pair of pins directly below its original position.
3. Return the jumper back to its original position, and the LED should begin to flash.

If the motors are failing to respond to a guitar strum:

1. Check that the device is in "Tune" mode.
2. If this is correct and the device is not working, try resetting the dsPIC by using the reset button on the PCB.
3. If the device still is not working, try to find a quieter setting; significant background noise can cause the device to malfunction.
4. If strings are tuned out of range, remove device as shown in Subsection 5.5, bring strings within their allowable frequency range as shown in Table 3, and reinstall uTune as shown in Subsection 5.1.

5.5 Device Removal

To remove device:

1. Power off device by unplugging in battery.
3. Slide left and right structures away from each other.
4. Slide center structure off of head of guitar.
5. Slide left and right structures back toward each other.

6 To-Market Design Changes

During the process of designing and creating uTune, there were limits in budget, time, and experience that resulted in a product with decent room for improvement prior to its commercial release. The following sections detail changes that would be made to improve the uTune design for efficiency, style, and user-friendliness.

6.1 Self-Plucking

In the use of the current uTune model, the user is required to pluck the string of the guitar that as it is being tuned. While this is certainly not too burdensome, as the user still does not have to worry about turning the tuning pegs or monitoring its tuning status, this is still an effort that the user must exert. In order to improve the user experience, a future uTune design would include a small component by the sound hole of the guitar that would do this plucking in the stead of the user, providing a fully hands-free tuning experience. This plucking device would require much more time and nuanced mechanical understanding to develop, which is why it is not present in the current uTune model. The plucking device would likely be located inside of the guitar body and pluck the strings through the sound hole so as to not obstruct the user's normal strumming by its mounted location.

6.2 Allowing Full Strum for Tuning

The current uTune model uses an iterative process to tune the guitar strings–one at a time. While this method works to properly tune the guitar, it would be even faster and require less precision for uTune to add the capability of tuning the guitar with a strum of all six strings at once. For the user, this would require less attention and precision to strum fully rather than strum different strings at a time. If the mechanical plucker from Subsection 6.1 were used, this ability to tune with a full strum would reduce a great amount of mechanical complexity that would be required to pluck individual strings. The process to tune the guitar with a full strum, assuming that the audio path is microphone-based, would involve taking an FFT, slightly de-tuning one string while all are still ringing out, taking a new FFT, and subtracting the older from the newer FFT so as to remove all of the static harmonics and identify the harmonics that belong to that moved string. The same process as uTune's current model would calculate the fundamental string frequency based on these harmonics. Then, the same frequency difference proportional control loop would turn the motor to tune the string toward its set point. After each turn and new sample, a difference between the previous and newest FFT would be the frequency data to be processed, rather than simply the newest FFT as with the current uTune model.

With a bigger budget, instead of using this wiggle technique, an expensive hexaphonic pickup could be used in the place of a microphone. A hexaphonic pickup measures and sends six different signals, one for each string. Thus, it is given which string each measured harmonic set belongs to. This would remove the need to 'wiggle' a string and take a difference between FFTs.

Also in the current uTune model, there is a restriction of each string's tunable frequency range which has two results. First, there may be frequencies close but outside of this range to which the user may want to use the guitar (these cases are very rare, but there could be a few). Secondly, if a string accidentally gets tuned outside of this range, or if the guitar simply starts with a string in this range, uTune is unable to know which string it actually is when it is plucked. In the current uTune algorithm, the device uses the string frequency to identify the string position, which naturally restricts the frequency range of each string position. With the full strum method, the string position is given and the heard string frequency can be anything, always attributed to the known string position. This solves the problem of a guitar with strings outside of their currently allowed ranges.

6.3 Smaller Servo Motors

The current uTune servos successfully turn the tuning pegs with plenty of torque. While budget and time were a concern in the designing of uTune, the second iteration of uTune would be able to look into smaller servo motors for tuning peg control with less maximum torque than the current motors, which would still

be able to turn the tuning pegs if high enough. This would lower the bulk and power consumption of uTune which are inherently limiting factors of a device of this nature.

6.4 Improved Housing

In the current uTune model, the physical housing for the electrical hardware would be improved in multiple ways. Though this is a mechanical engineer's job, it is necessary for a professional, appealing, and user-friendly product. The new device housing would fully contain all electrical boards and wires in the most space-efficient manner.

6.5 Improved User Interface

The current user interface does the job of allowing the user to pick tuning schemes and displaying the status of strings as they are tuned. Due to time concerns, this was the extent of the touchscreen user interface's capabilities. Before releasing uTune commercially, it would be beneficial to add more features on the touch screen, such as the ability to store and load tuning schemes as can be done in the iPhone app. This would eliminate the need to own an iPhone in order to have this capability. Also, it could be helpful to add box in the touchscreen that displays the current string notes, current string frequencies, and set point frequencies to increase the available knowledge about the guitar's current tuning state. There could also be added a capability for defining the note A4, which generally defines the frequencies of all notes.

Finally, an on/off switch should be added to uTune such that the user does not have to touch the battery and all electrical hardware may stay inside of the device housing.

6.6 Tension-Based Tuning

Rather than using the sound wave of a string to tune it, the tension of each string could be used to identify the current tuning of the string. Since each string, given its linear mass density and length, has a specific tension to which it would need to be tuned for each note, the matching of this tension on all strings to match the tension required for each of the six note set points would allow constant tuning maintenance without needing to strum any strings or be using the guitar at all. The device could be set up to have a wake mode and a sleep mode. Sitting on the guitar at all times, it would rest in sleep mode. At medium intervals, uTune would then enter wake mode and check the current tension of the six strings. A frequency would be attributed to each of the strings based on the measured tension, these values would be compared to the frequencies of the six string note set points, and the motors would be turned in the current proportional difference control loop to restore the tension of each string to the proper level that would yield the desired note on each.

7 Conclusion

Tuning is an essential aspect of a musical performance, but it should not be a burden for a performer while they are in the middle of making music. A performer, especially an inexperienced one, can spend a significant amount of time focused on tuning. Additionally, the human ear is not as high resolution as modern microphones due to critical bandwidth perception limitations. That makes an automated guitar tuner both more accurate and more efficient, and allows less experienced guitar players to focus on learning to play. Due to all of these reasons, an automated guitar tuner is a highly desirable and useful item.

uTune set out to fulfill these market needs by satisfying a list of criteria: it is lightweight, removable from the head of the guitar, relatively easy to install, can tune all six strings, requires minimal work from the user, and incorporates two different UI interfaces (built-in touchscreen and iPhone app) for different environments (i.e. hidden touchscreen is useful during an onstage performance, to prevent having to pull out iPhone in front of the audience). With the objectives and requirements laid out in Section 2, the uTune team set out to create the perfect device.

The process of constructing uTune was involved, and required a great deal of creative thinking, especially with the physical device arrangement, memory management, and frequency harmonic peak identification. Additionally, debugging the power board, touch screen, and app connectivity proved to be substantial design challenges. Overcoming these challenges was essential to the success of the project.

uTune features a microphone and differential amplifier to record sound from a plucked string, and refined Fast Fourier Transform and frequency evaluation algorithms to determine the fundamental frequency of the string. This computation is done quickly and efficiently on the dsPIC microcontroller, which then compares the recorded frequency to a stored "intended" frequency and sends a PWM signal to the corresponding motor, which turns the guitar peg in the proper direction in order to adjust the frequency of the string. These "intended" frequencies are selected by the user in one of two ways: via an iPhone app which communicates with the dsPIC via a Bluetooth module and UART protocol, or a thin-film transistor LCD touchscreen connected directly to the dsPIC via SPI. The LCD also displays tuning progress (sharp, flat, or in-tune). Together, these subsystems allow the user to very easily select a tuning scheme, and by strumming one string at a time, uTune takes care of all of the behind-the-scenes work to physically turn each peg until the entire guitar is in tune.

With a working prototype, evaluating the various elements of the project, there were also a number of areas for future development, especially if this particular product were to be taken to market. Some of these proposed additional features have the purpose of minimizing work done by the user (such as self-plucking strings or simultaneous full-strum tuning), while others are designed to improve practical use (such as the use of smaller motors and an overall smaller housing, or an improved user interface). Still, there is room for greater exploration and redesign, including exploring other tuning methodologies such as tension-based tuning.

Ultimately, a working prototype was created and uTune accomplished the major objectives it was intended to fulfill. Through the interconnected subsystems described above and in the rest of this report, uTune resolves several key issues faced by musicians: going out of tune over time due to temperature and other stressors, requiring quick changes to different tuning schemes during performances or practice, and the ability to simplify the process for less experienced players, thus making guitar performance more accessible to all who are interested. uTune has immense market potential, and will someday enhance the experience of performers, musicians, and audience members alike.

A Hardware Schematics

A.1 Power Subsystem

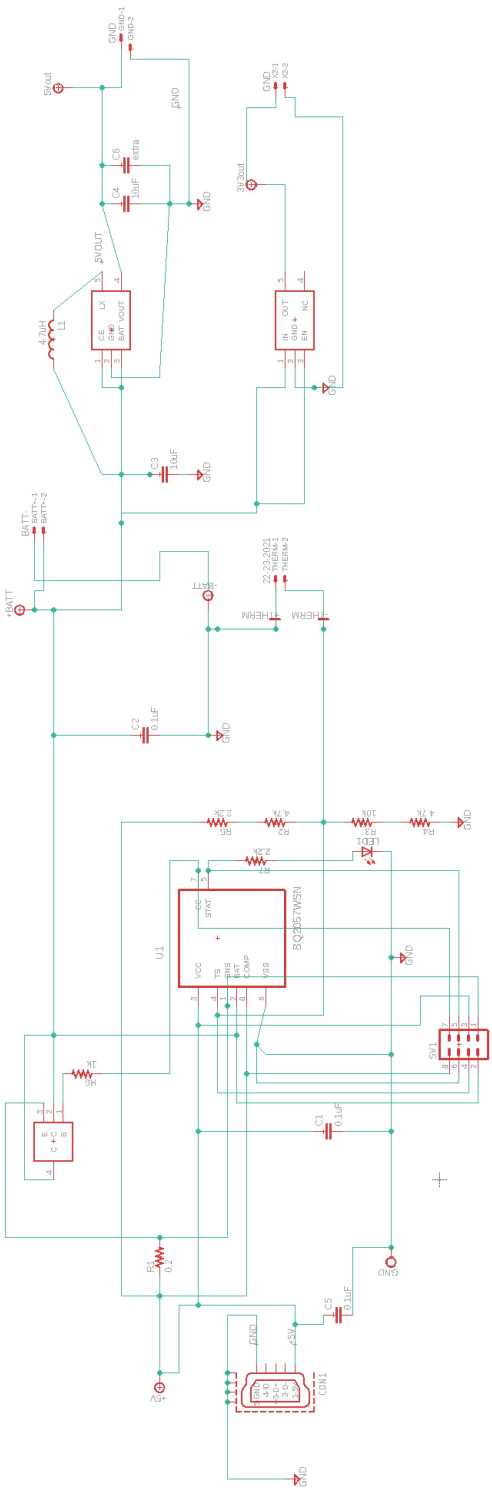

Figure 28: Power Subsystem Schematic

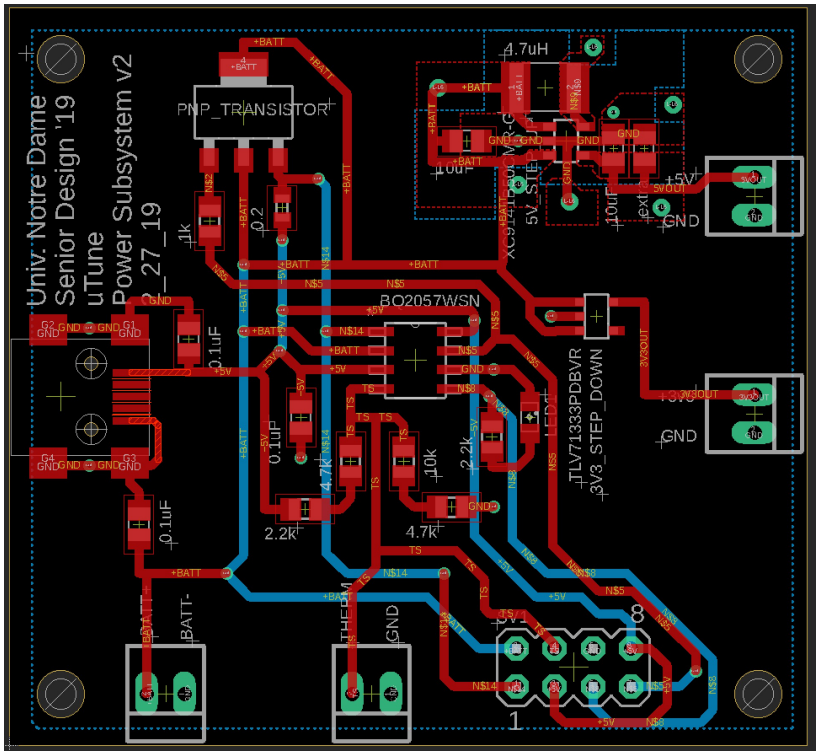

Figure 29: Power Subsystem PCB

A.2 dsPIC Subsystem

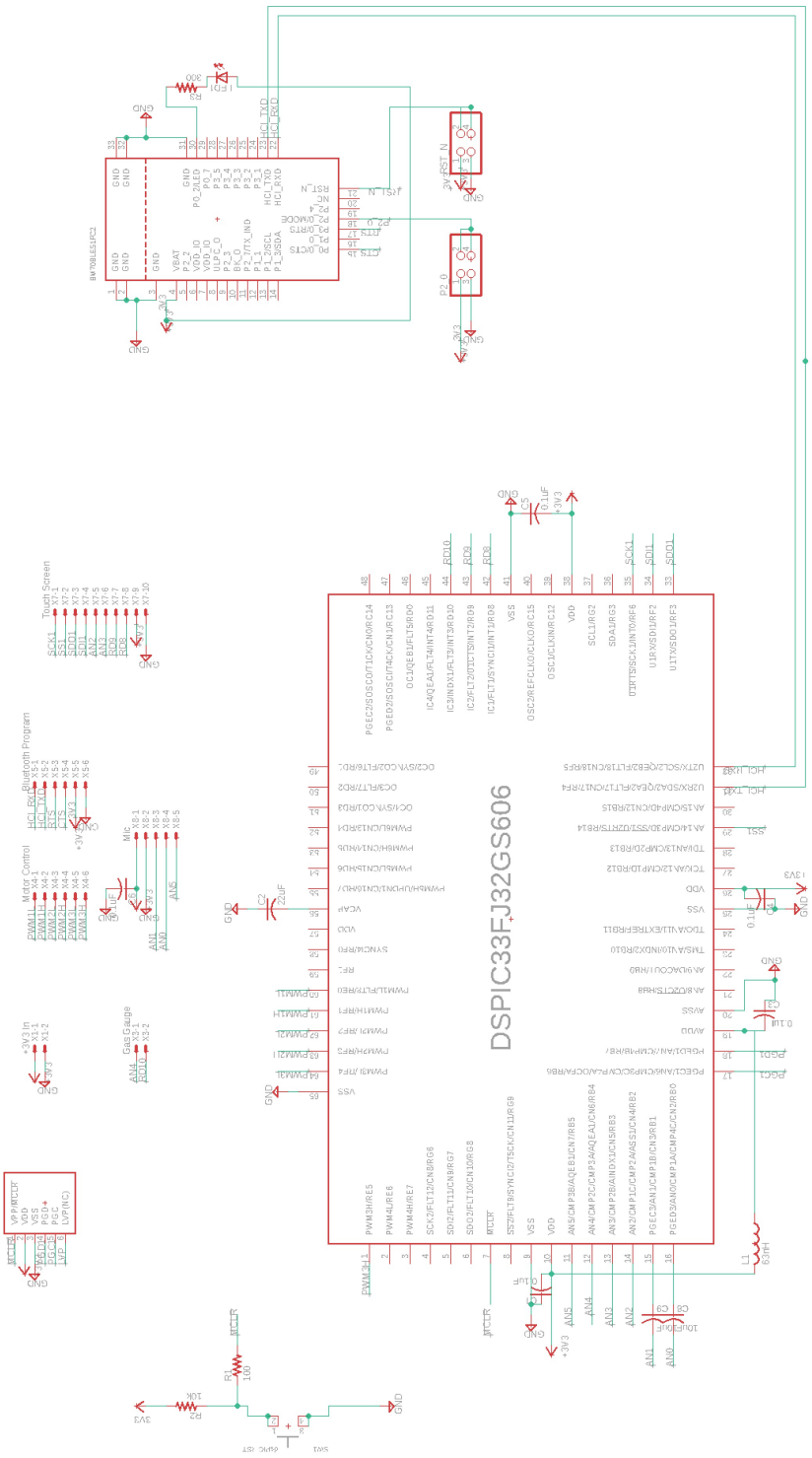

Figure 30: dsPIC Subsystem Schematic

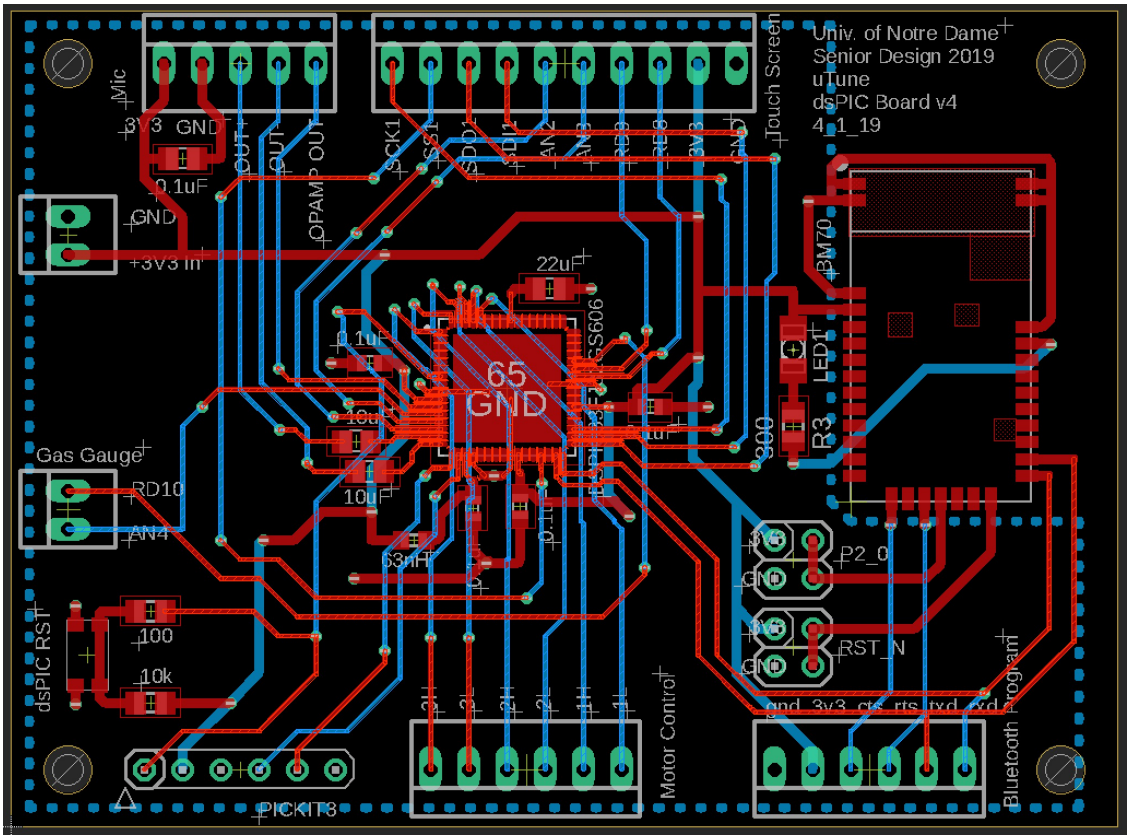

Figure 31: dsPIC Subsystem PCB

A.3 Microphone Subsystem

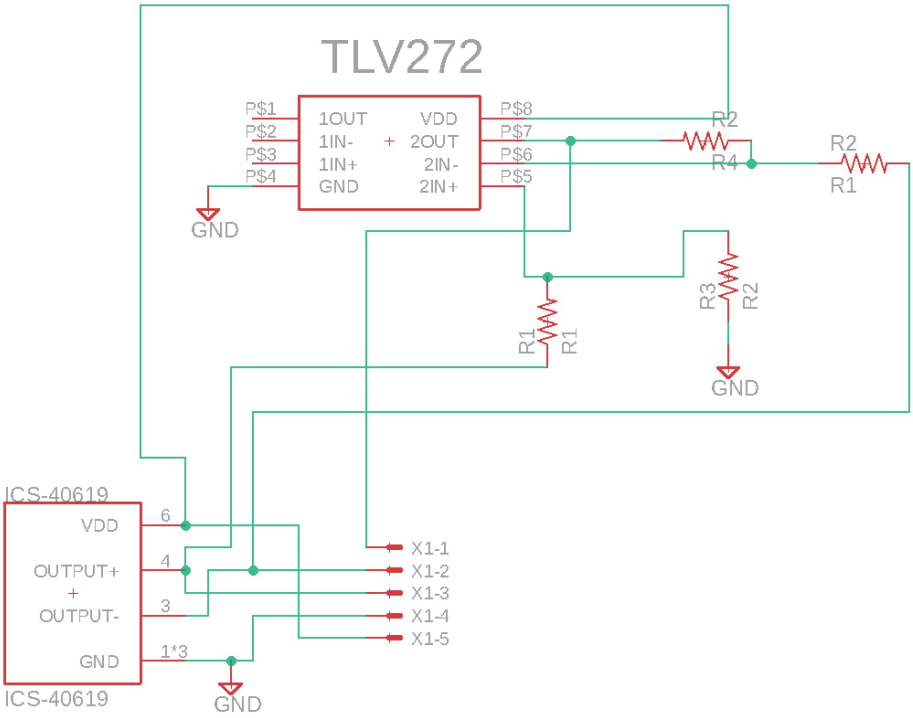

Figure 32: Microphone Subsystem Schematic

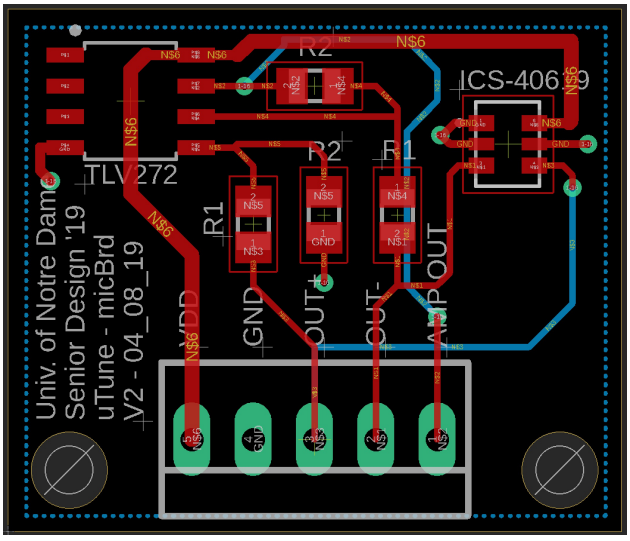

Figure 33: Microphone Subsystem PCB

A.4 3D Print Schematics

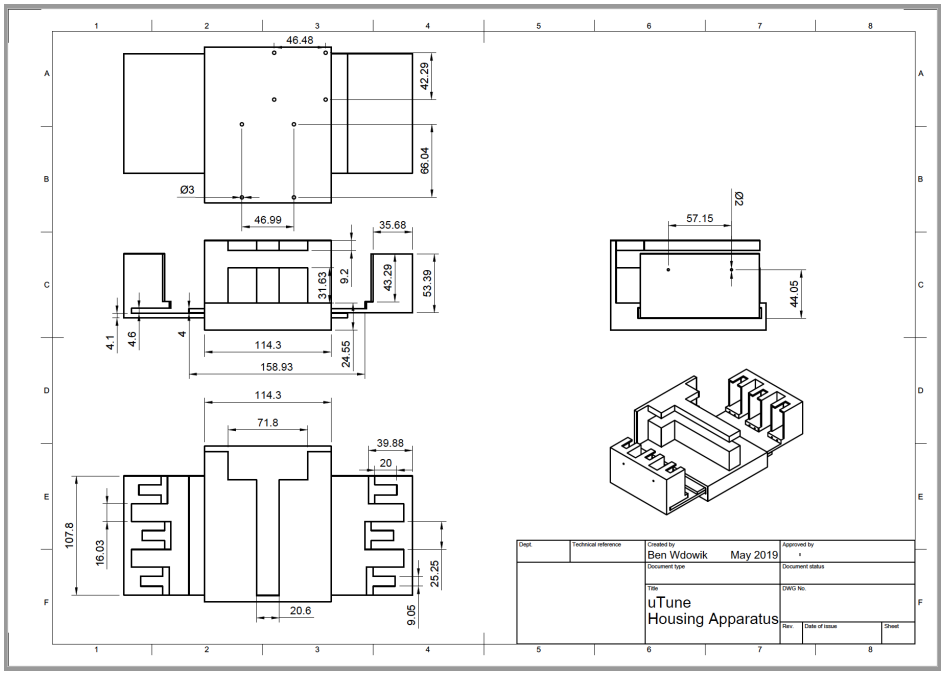

Figure 34: 3D-Print Housing Apparatus Schematic

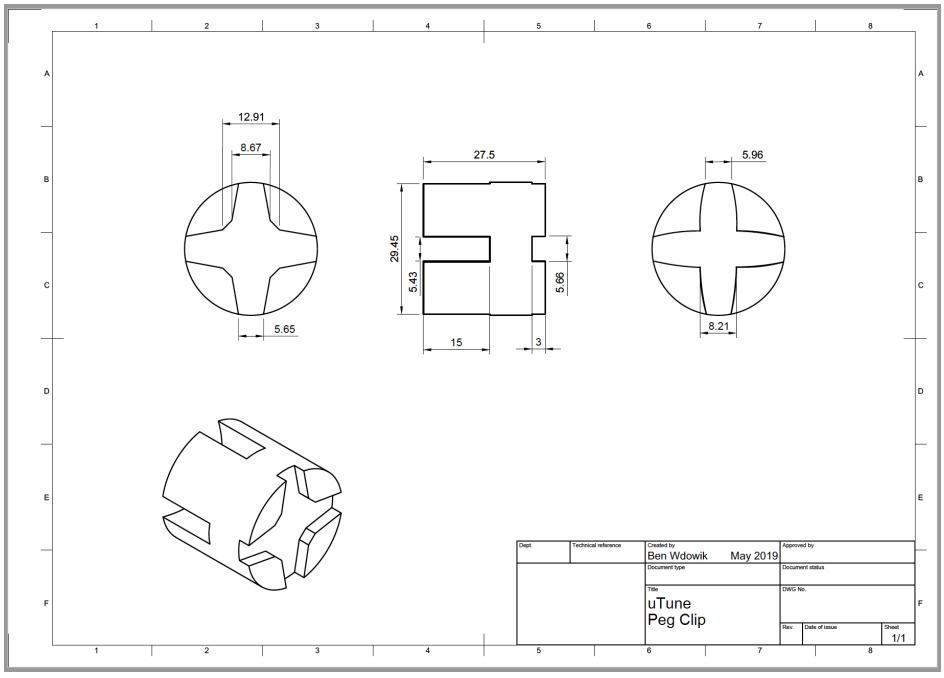

Figure 35: 3D-Print Peg Clip Schematic

B Data Sheets

B.1 dsPIC33FJ64GS606

http://ww1.microchip.com/downloads/en/DeviceDoc/70000591f.pdf

B.2 Power Subsystem

Charge Management IC: https://www.ti.com/lit/ds/symlink/bq2057t.pdf
5V DC/DC Converter: https://www.torexsemi.com/file/xc9141/XC9141-XC9142.pdf
3.3V Regulator: http://www.ti.com/lit/ds/symlink/tlv713p.pdf

B.3 Microphone Subsystem

Microphone: https://www.invensense.com/wp-content/uploads/2016/02/ICS-40619-Datasheet.pdf
Operational Amplifier: http://www.ti.com/lit/ds/symlink/tlv272.pdf
Differential Amplifier Schematic: https://www.electronics-tutorials.ws/opamp/opamp_5.html

B.4 BM70 Bluetooth Module

http://ww1.microchip.com/downloads/en/DeviceDoc/BM70-71-Bluetooth-Low-Energy-BLE-Module-Data-Sheet-DS60001372H.pdf

B.5 Touchscreen

https://cdn-learn.adafruit.com/downloads/pdf/adafruit-2-4-color-tft-touchscreen-breakout.pdf

C Software

A red arrow symbolizes that the previous line of code was too large for a single line and thus represents continuation of the previous line.

C.1 MPLAB Code

C.1.1 main.c

```
/*
 * File: main.c
 * Author: Kevin Schneier
 *
 */


/* Preprocessor ******************************************************************/


#define FCY 39613750
#define BAUDRATE 115200
#define ADC_BUFFER_SIZE 1024
#define ADC_RATE 1333.33333
//#define ADC_RATE 1323.767753
// By the math, the ADC_RATE should be 1323.767753; but this isn't what it seems like?
#define SORT 12

/* Motor Defines */
#define s1Up() {SDC1 = 163;}
#define s1Down() {SDC1 = 183;}
#define s2Up() {PDC1 = 163;}
#define s2Down() {PDC1 = 183;}
#define s3Up() {SDC2 = 163;}
#define s3Down() {SDC2 = 183;}
#define s4Up() {PDC2 = 163;}
#define s4Down() {PDC2 = 183;}
#define s5Up() {SDC3 = 163;}
#define s5Down() {SDC3 = 183;}
#define s6Up() {PDC3 = 163;}
#define s6Down() {PDC3 = 183;}

// For calibration between touchscreen and screen coordinates
#define TS_MINX 200
#define TS_MINY 120
#define TS_MAXX 900
#define TS_MAXY 900

#define MINPRESSURE 10
#define MAXPRESSURE 1000

// UI-specific definitions
#define noteBoxSize 46
#define spacing 50
```

```
#include "xc.h"
#include "fft.h"
#include <stdlib.h>
#include <math.h>
#include <p33FJ64GS606.h>
#include "config.h"

/* Touchscreen */
#include "touchscreen.h"
#include "touchCapability.h"

/* Get extracting function */
#include "fundamental.h"




/* Function Prototypes ****************************************************/

void init_PLL(void);
void init_timer(void);
void init_ADC(void);
void init_PWM(void);
void init_UART(void);
void send_UART(char c);
unsigned char receive_UART(void);
double decode_UART(void);
void init_SPI(void);
void stopMotors(void);
int motorControl(double measured, double target);
int masterControl(uint16_t *fft, int numPoints, double *freqs);
void init_timer2(void);
int * getPoint(void);
void startTuning(void);
void stopTuning(void);
void tuningDisplay(int tuningStringNum, int mode);
void freqPicker(void);
void resetDisplay(void);
void updateDisplayText(int noteRst);
void selectNote(void);
void rangeMapping(int input_x, int input_y);
void updateScreenMemoryArray(void);


/* Global Variables *******************************************************/

double frequencies[6] = {0};
//double frequencies[6] = {82.41, 110.0, 146.8, 196.0, 246.9, 329.6};
double fundamental = 0;
double last_fundamental = 0;
double peaks[SORT] = {0};
```

```
int freq_index = 0;
int ADC_index = 0;
int FFT_index = 0;
int divisor = 8;
int mode = 0;
double threshold = ADC_RATE/ADC_BUFFER_SIZE;


int ADC_values[ADC_BUFFER_SIZE] = {0}; // Initialize real array
int ADC_imaginary[ADC_BUFFER_SIZE] = {0}; // Initialize imaginary components to 0
uint16_t amplitudes[ADC_BUFFER_SIZE/2] = {0}; // Initialize amplitude array


/* Touchscreen */
unsigned short _width = ILI9340_TFTWIDTH;
unsigned short _height = ILI9340_TFTHEIGHT;
unsigned short cursor_y = 0;
unsigned short cursor_x = 0;
unsigned short textsize = 0;
unsigned short textcolor = 0;
unsigned short textbgcolor = 0;
unsigned short wrap = 0;
unsigned short rotation = 0;

int ADC_AN3_val=0;
int ADC_AN2_val=0;
int coords[] = {-99,-99,-99};
int touchMode = 0;
int counter=0;
int touchReady = 0;
int i;

// Initialize note data
int selectedString = 0; //note selected on display, can be 0-5
int lastSelectedString = 0;

int noteSpacing[6];
int setNotesArray[6]={4, 0, 3, 6, 1, 4}; //A-0, B-1, C-2, D-3, E-4, F-5, G-6
char fullNotesArray[7] = "ABCDEFG";

int isSharp[6] = {0, 0, 0, 0, 0, 0};

//Initialize conversion data
int touchLoc[2];


// Initialize tuning data
int isTuning = 0;


/* Interrupt Service Routines ****************************************************/

void __attribute__((__interrupt__, no_auto_psv)) _T1Interrupt(void) {
```

```
    if (ADC_index++ >= ADC_BUFFER_SIZE/divisor) {
        IEC0bits.T1IE = 0; // Disable Timer 1 interrupt
        ADC_index = 0;
        IEC7bits.ADCP2IE = 1; // Enable ADC interrupt, we're ready to sample again
        stopMotors();
    }


    IFS0bits.T1IF = 0;
    IFS7bits.ADCP2IF = 0; // Clear ADC Pair 2 Interrupt Flag

}

void __attribute__((__interrupt__, no_auto_psv)) _ADCP2Interrupt(void) {

    ADC_values[ADC_index++] = ADCBUF5;
    ADSTATbits.P2RDY = 0; // Clear status bit
    if (ADC_index >= ADC_BUFFER_SIZE){
        IEC7bits.ADCP2IE = 0; // Disable ADC interrupt when buffer is full
        FFT(ADC_values, ADC_imaginary);
        for (FFT_index = 1; FFT_index < ADC_BUFFER_SIZE/2; FFT_index++){
            amplitudes[FFT_index] = FFTAmplitude(ADC_values[FFT_index], ADC_imaginary[
                ↪ FFT_index]);
        }
        divisor = masterControl(amplitudes, ADC_BUFFER_SIZE/2, frequencies);
        ADC_index = 0;
        IEC0bits.T1IE = 1; // Enable timer 1 interrupt
    }

    IFS0bits.T1IF = 0; // Clear Timer 1 Interrupt Flag
    IFS7bits.ADCP2IF = 0; // Clear ADC Pair 2 Interrupt Flag
}

void __attribute__((__interrupt__, no_auto_psv)) _U2RXInterrupt(void) {
    freq_index = freq_index % 6;
    frequencies[freq_index++] = decode_UART();
    if (freq_index == 6 && frequencies[5] > 1){
        updateScreenMemoryArray();
    }
    IFS1bits.U2RXIF = 0;
}

void __attribute__((__interrupt__, no_auto_psv)) _ADCP1Interrupt(void) {

        /* ISR here */

        /* Whenever you read a value (i.e. myVal = ADCBUF2 or myVar = ADCBUF3),
        you have to do the following immediately after:
        ADSTATbits.P1RDY = 0; */

    ADC_AN3_val = ADCBUF3;
    ADC_AN2_val = ADCBUF2;

    touchReady = 1;
```

```
    ADSTATbits.P1RDY = 0;

        /* At the end of the ISR: */
        IFS0bits.T2IF = 0;  // Clear Timer 2 interrupt flag
        IFS6bits.ADCP1IF = 0;  // Clear ADC Pair 1 interrupt flag

}



/* Main Execution ***************************************************************/

int main(void) {

    /* Setup PLL */
    init_PLL();
    init_SPI();

    /* Initialize components */
    init_UART();
    init_PWM();
    init_timer();
    init_ADC();

    /* Initialize ADC*/
    init_timer2();

    /* Initialize Screen */
    tft_begin();

    /* Do stuff */
    resetDisplay();
    selectNote();


    /* Main loop */
    while(1){
        if (frequencies[5] < 1 && frequencies[0] > 1){
            continue;
        }
        if (frequencies[5] < 1){
            if (isTuning == 1){
                stopTuning();
            }
            PTCONbits.PTEN = 0;
        }
        else {
            if (isTuning == 0){
                startTuning();
            }
            PTCONbits.PTEN = 1;
        }
```

```
    getPoint();
    rangeMapping(coords[0],coords[1]);

    if (touchLoc[0]<0 || touchLoc[0]>320 || touchLoc[1]<0 || touchLoc[1]>240)
        coords[2]=0;
    if (coords[2]>MINPRESSURE && coords[2]<MAXPRESSURE){
        if (isTuning == 0) {
                        //check if any display functions are being pressed (e.g.
                            ↪ note boxes, triangles, etc.)
                        //boxes
            if ((touchLoc[1] > 91) && (touchLoc[1] < (91+noteBoxSize))) {
                for (i=0;i<6;i++){
                    if ((touchLoc[0] > 12 + i*spacing) && (touchLoc[0] < 12+i*spacing+
                        ↪ noteBoxSize)){
                        lastSelectedString = selectedString;
                        selectedString = i;
                    }
                }
                selectNote();
            }

//triangles
if ((touchLoc[0] > 12+selectedString*spacing) && (touchLoc[0] < 12+selectedString*
    ↪ spacing+noteBoxSize)) {
  //up arrow
  if ((touchLoc[1] > 81-noteBoxSize) && (touchLoc[1] < 81)) {
    //tft_fillCircle(touchLoc[0], touchLoc[1], 3, ILI9340_RED);
    switch (selectedString) {

        case 0: // low E string

                        //add sharps
          //if not currently on B or E, and not sharp
            if ((setNotesArray[selectedString]==4) && (isSharp[selectedString]==0)){
                setNotesArray[selectedString]=1;
            }
            else if ((setNotesArray[selectedString] != 1) && (setNotesArray[
                ↪ selectedString] != 4) && (isSharp[selectedString] == 0)) {
              isSharp[selectedString] = 1;
            }
            //increase note selection by 1
            else if (setNotesArray[selectedString] < 6) {
              setNotesArray[selectedString] ++;
              isSharp[selectedString] = 0;
            } else { //loop back to beginning of note array
              setNotesArray[selectedString] = setNotesArray[selectedString] - 6;
              isSharp[selectedString] = 0;
            }


            break;

        case 1: // A string
```

```
                                    //add sharps
      //if not currently on B or E, and not sharp
    if ((setNotesArray[selectedString]==0) && (isSharp[selectedString]==0)){
        setNotesArray[selectedString]=5;
    }
    else if ((setNotesArray[selectedString] != 1) && (setNotesArray[
        ↪ selectedString] != 4) && (isSharp[selectedString] == 0)) {
      isSharp[selectedString] = 1;
    }
    //increase note selection by 1
    else if (setNotesArray[selectedString] < 6) {
      setNotesArray[selectedString] ++;
      isSharp[selectedString] = 0;
    } else { //loop back to beginning of note array
      setNotesArray[selectedString] = setNotesArray[selectedString] - 6;
      isSharp[selectedString] = 0;
    }
    break;

case 2: // D string
                                    //add sharps
      //if not currently on B or E, and not sharp
    if ((setNotesArray[selectedString]==3) && (isSharp[selectedString]==0)){
        setNotesArray[selectedString]=0;
        isSharp[selectedString]=1;
    }
    else if ((setNotesArray[selectedString] != 1) && (setNotesArray[
        ↪ selectedString] != 4) && (isSharp[selectedString] == 0)) {
      isSharp[selectedString] = 1;
    }
    //increase note selection by 1
    else if (setNotesArray[selectedString] < 6) {
      setNotesArray[selectedString] ++;
      isSharp[selectedString] = 0;
    } else { //loop back to beginning of note array
      setNotesArray[selectedString] = setNotesArray[selectedString] - 6;
      isSharp[selectedString] = 0;
    }
    break;

case 3: // G string
                                                    //add sharps
      //if not currently on B or E, and not sharp
    if ((setNotesArray[selectedString]==6) && (isSharp[selectedString]==0)){
        setNotesArray[selectedString]=3;
        isSharp[selectedString]=1;
    }
    else if ((setNotesArray[selectedString] != 1) && (setNotesArray[
        ↪ selectedString] != 4) && (isSharp[selectedString] == 0)) {
      isSharp[selectedString] = 1;
    }
    //increase note selection by 1
    else if (setNotesArray[selectedString] < 6) {
      setNotesArray[selectedString] ++;
```

```
            isSharp[selectedString] = 0;
          } else { //loop back to beginning of note array
            setNotesArray[selectedString] = setNotesArray[selectedString] - 6;
            isSharp[selectedString] = 0;
          }
          break;

      case 4: // B string
                                                    //add sharps
          //if not currently on B or E, and not sharp
          if ((setNotesArray[selectedString]==1) && (isSharp[selectedString]==0)){
              setNotesArray[selectedString]=6;
              isSharp[selectedString]=1;
          }
          else if ((setNotesArray[selectedString] != 1) && (setNotesArray[
              ↪ selectedString] != 4) && (isSharp[selectedString] == 0)) {
            isSharp[selectedString] = 1;
          }
          //increase note selection by 1
          else if (setNotesArray[selectedString] < 6) {
            setNotesArray[selectedString] ++;
            isSharp[selectedString] = 0;
          } else { //loop back to beginning of note array
            setNotesArray[selectedString] = setNotesArray[selectedString] - 6;
            isSharp[selectedString] = 0;
          }
          break;

      case 5: // high E string
                                                    //add sharps
          //if not currently on B or E, and not sharp
          if ((setNotesArray[selectedString]==4) && (isSharp[selectedString]==0)){
              setNotesArray[selectedString]=2;
          }
          else if ((setNotesArray[selectedString] != 1) && (setNotesArray[
              ↪ selectedString] != 4) && (isSharp[selectedString] == 0)) {
            isSharp[selectedString] = 1;
          }
          //increase note selection by 1
          else if (setNotesArray[selectedString] < 6) {
            setNotesArray[selectedString] ++;
            isSharp[selectedString] = 0;
          } else { //loop back to beginning of note array
            setNotesArray[selectedString] = setNotesArray[selectedString] - 6;
            isSharp[selectedString] = 0;
          }
          break;
  }



  updateDisplayText(0);

}
```

```
// down arrow ****FIX THIS****
else if ((touchLoc[1] > 91+noteBoxSize+10) && (touchLoc[1] < 91+(2*noteBoxSize)+10))
    ↪ {
  //tft_fillCircle(touchLoc[0], touchLoc[1], 3, ILI9340_GREEN);

  switch (selectedString) {

      case 0: // low E string
          if ((setNotesArray[selectedString]==1) && (isSharp[selectedString]==0)){
              setNotesArray[selectedString]=4;
          }
          else if (isSharp[selectedString] == 1){
              isSharp[selectedString] = 0;
          }
          else {

              if (setNotesArray[selectedString] > 0) {
                setNotesArray[selectedString] --;
              } else { //loop back to beginning of note array
                setNotesArray[selectedString] = setNotesArray[selectedString] + 6;
              }

              if ((setNotesArray[selectedString] != 1) && (setNotesArray[
                  ↪ selectedString] != 4)) {
                isSharp[selectedString] = 1;
              }
          }

          break;

      case 1: // A string
          if ((setNotesArray[selectedString]==5) && (isSharp[selectedString]==0)){
              setNotesArray[selectedString]=0;
          }
          else if (isSharp[selectedString] == 1){
              isSharp[selectedString] = 0;
          }
          else {

              if (setNotesArray[selectedString] > 0) {
                setNotesArray[selectedString] --;
              } else { //loop back to beginning of note array
                setNotesArray[selectedString] = setNotesArray[selectedString] + 6;
              }

              if ((setNotesArray[selectedString] != 1) && (setNotesArray[
                  ↪ selectedString] != 4)) {
                isSharp[selectedString] = 1;
              }
          }
          break;

      case 2: // D string
          if ((setNotesArray[selectedString]==0) && (isSharp[selectedString]==1)){
```

```
            setNotesArray[selectedString]=3;
            isSharp[selectedString]=0;
        }
        else if (isSharp[selectedString] == 1){
            isSharp[selectedString] = 0;
        }
        else {

            if (setNotesArray[selectedString] > 0) {
              setNotesArray[selectedString] --;
            } else { //loop back to beginning of note array
              setNotesArray[selectedString] = setNotesArray[selectedString] + 6;
            }

            if ((setNotesArray[selectedString] != 1) && (setNotesArray[
                ↪ selectedString] != 4)) {
              isSharp[selectedString] = 1;
            }
        }
        break;

    case 3: // G string
        if ((setNotesArray[selectedString]==3) && (isSharp[selectedString]==1)){
            setNotesArray[selectedString]=6;
            isSharp[selectedString]=0;
        }
        else if (isSharp[selectedString] == 1){
            isSharp[selectedString] = 0;
        }
        else {

            if (setNotesArray[selectedString] > 0) {
              setNotesArray[selectedString] --;
            } else { //loop back to beginning of note array
              setNotesArray[selectedString] = setNotesArray[selectedString] + 6;
            }

            if ((setNotesArray[selectedString] != 1) && (setNotesArray[
                ↪ selectedString] != 4)) {
              isSharp[selectedString] = 1;
            }
        }
        break;

    case 4: // B string
        if ((setNotesArray[selectedString]==6) && (isSharp[selectedString]==1)){
            setNotesArray[selectedString]=1;
            isSharp[selectedString]=0;
        }
        else if (isSharp[selectedString] == 1){
            isSharp[selectedString] = 0;
        }
        else {
```

```
                if (setNotesArray[selectedString] > 0) {
                  setNotesArray[selectedString] --;
                } else { //loop back to beginning of note array
                  setNotesArray[selectedString] = setNotesArray[selectedString] + 6;
                }

                if ((setNotesArray[selectedString] != 1) && (setNotesArray[
                    ↪ selectedString] != 4)) {
                  isSharp[selectedString] = 1;
                }
            }
            break;

        case 5: // high E string
            if ((setNotesArray[selectedString]==2) && (isSharp[selectedString]==0)){
                setNotesArray[selectedString]=4;
            }
            else if (isSharp[selectedString] == 1){
                isSharp[selectedString] = 0;
            }
            else {

                if (setNotesArray[selectedString] > 0) {
                  setNotesArray[selectedString] --;
                } else { //loop back to beginning of note array
                  setNotesArray[selectedString] = setNotesArray[selectedString] + 6;
                }

                if ((setNotesArray[selectedString] != 1) && (setNotesArray[
                    ↪ selectedString] != 4)) {
                  isSharp[selectedString] = 1;
                }
            }
            break;
      }



    }
        updateDisplayText(0);

  }

  //tune button
  if ((touchLoc[1] > 210) && (touchLoc[1] < 240) && (touchLoc[0] > 235) && (touchLoc[0]
      ↪  < 308)){
      freqPicker();
      startTuning();
  }
}
//stop button
      if ((touchLoc[1] > 210) && (touchLoc[1] < 240) && (touchLoc[0] > 25) && (touchLoc
          ↪ [0] < 98)){
          frequencies[0] = 0;
```

```
            frequencies[1] = 0;
            frequencies[2] = 0;
            frequencies[3] = 0;
            frequencies[4] = 0;
            frequencies[5] = 0;
            stopTuning();
        }



      }

    }
    return 0;

}



/* Function Definitions ****************************************************/

void init_PLL(void) {

    /* Configure PLL prescaler, PLL postscaler, PLL divisor */
    PLLFBD = 41; /* M = PLLFBD + 2 */
    CLKDIVbits.PLLPOST = 0; /* N1 = 2 */
    CLKDIVbits.PLLPRE = 0; /* N2 = 2 */

    /* Configure Auxiliary Clock */
    ACLKCONbits.FRCSEL = 1; /* FRC provides input for Auxiliary PLL (x16) */
    ACLKCONbits.SELACLK = 1; /* Auxiliary Oscillator provides clock source for PWM & ADC
        ↪ */
    ACLKCONbits.APSTSCLR = 0; /* Divide Auxiliary clock by 256 */
    ACLKCONbits.ENAPLL = 1; /* Enable Auxiliary PLL */

    /* Wait for clock */
    while(ACLKCONbits.APLLCK != 1); /* Wait for Auxiliary PLL to Lock */

}

void init_timer(void) {

    /* Timer 1 Configuration */
    T1CONbits.TON = 0; // Turn off Timer 1
    T1CONbits.TCS = 0;
    T1CONbits.TGATE = 0;
    T1CONbits.TCKPS = 0b00; // Clock prescaler 1:1

    /* Timer 1 Value Setting*/
    TMR1 = 0; // Reset Timer 1 Value
    PR1 = 39900/4*3; // Timer 1 Period Match

    /* Timer 1 Interrupt Handling */
    IPC0bits.T1IP = 0x01; // Set Timer 1 Interrupt Priority
```

```
    IFS0bits.T1IF = 0; // Clear Timer 1 Interrupt Flag
    IEC0bits.T1IE = 0; // Disable Timer 1 Interrupt

    /* Timer 1 Activation */
    T1CONbits.TON = 1; // Turn on Timer 1

}

void init_ADC(void) {

    /* Set ADC Configuration Bits */
    ADCONbits.ADON = 0; // Turn ADC Module off
    ADCONbits.FORM = 0; // Output in integer format
    ADCONbits.SLOWCLK = 0; // ADC is clocked by primary PLL
    ADCONbits.EIE = 0; // Interrupt is generated after first conversion is completed
    ADCONbits.ORDER = 0; // Convert odd channel first
    ADCONbits.SEQSAMP = 0; // Select simultaneous sampling
    ADCONbits.ADCS = 5; // ADC clock = FADC/1


    /* Set Trigger Source */
    ADCPC1bits.TRGSRC2 = 0b01100; // ADC channels AN4 and AN5 have conversion triggered
        ↪ by Timer 1 period

    /* Set Analog Pins */
    ADPCFGbits.PCFG4 = 0; // Select AN4 as analog pin
    ADPCFGbits.PCFG5 = 0; // Select AN5 as analog pin

    /* Handle Interrupt initialization */
    IPC28bits.ADCP2IP = 0x01; // Set ADC Pair 2 Interrupt Priority (Level 2)
    IFS7bits.ADCP2IF = 0; // Clear ADC Pair 2 Interrupt Flag
    IEC7bits.ADCP2IE = 1; // Enable ADC Pair 2 Interrupt



    /* OTHER ADC SETUP */
    ADCPC0bits.TRGSRC1 = 0b11111; // ADC channels AN2 and AN3 have conversion triggered
        ↪ by Timer 2 period

    /* Set Analog Pins */
    ADPCFGbits.PCFG2 = 0; // Select AN2 as analog pin
    ADPCFGbits.PCFG3 = 0; // Select AN3 as analog pin

    /* Handle Interrupt initialization */
    IPC27bits.ADCP1IP = 0x02; // Set ADC Pair 1 Interrupt Priority
    IFS6bits.ADCP1IF= 0; // Clear ADC Pair 1 Interrupt Flag
    IEC6bits.ADCP1IE = 1; // Enable ADC Pair 1 Interrupt

    /* Turn on ADC Module */
    ADCONbits.ADON = 1;

}

void init_PWM(void) {
```

```
    PTCONbits.PTEN = 0; // Disable PWM module

    /* Configure Clock Prescaler and Period */
    PTCON2 = 5; // Scale down clock by 2^(this number)
    PTPER = 2305; // PWM period value (must match with our clocks)
    /* Old was PTCON2=5, PTPER=2305 */
    /* Old was 163, 182 */

    /* PWM1 Configuration */
    IOCON1bits.PENH = 1; // PWM1H is controlled by PWM module (formerly zero to be GPIO)
    IOCON1bits.PENL = 1; // PWM1L is controlled by PWM module
    IOCON1bits.PMOD = 0b11; // Select independent output PWM mode
    PWMCON1bits.ITB = 0; // No independent timebase mode
    PWMCON1bits.MDCS = 0; // PDCx and SDCx provide duty cycle information
    PDC1 = 0; // Initial duty cycle for PWMxH
    SDC1 = 0; // Initial duty cycle for PWMxL

    /* PWM2 Configuration */
    IOCON2bits.PENH = 1; // PWM2H is controlled by PWM module (formerly zero to be GPIO)
    IOCON2bits.PENL = 1; // PWM2L is controlled by PWM module
    IOCON2bits.PMOD = 0b11; // Select independent output PWM mode (important)
    PWMCON2bits.ITB = 0; // No independent timebase mode
    PWMCON2bits.MDCS = 0; // PDCx and SDCx provide duty cycle information
    PDC2 = 0; // Initial duty cycle for PWMxH
    SDC2 = 0; // Initial duty cycle for PWMxL

    /* PWM3 Configuration */
    IOCON3bits.PENH = 1; // PWM2H is controlled by PWM module (formerly zero to be GPIO)
    IOCON3bits.PENL = 1; // PWM2L is controlled by PWM module
    IOCON3bits.PMOD = 0b11; // Select independent output PWM mode (important)
    PWMCON3bits.ITB = 0; // No independent timebase mode
    PWMCON3bits.MDCS = 0; // PDCx and SDCx provide duty cycle information
    PDC3 = 0; // Initial duty cycle for PWMxH
    SDC3 = 0; // Initial duty cycle for PWMxL

    /* Enable PWM Module */
    PTCONbits.PTEN = 1; // Enable PWM module

}

void init_UART(void){

    /* Set Configuration Bits */
    U2MODEbits.USIDL = 0; // Continue in idle mode
    U2MODEbits.IREN = 0; // IrDA encoder and decoder are disabled
    U2MODEbits.RTSMD = 0; // U2RTS in Flow Control mode
    U2MODEbits.UEN = 0b00; // U2TX and U2RX pins are enabled and used; no flow control
        ↪ pins
    U2MODEbits.WAKE = 0; // Wake-up is disabled
    U2MODEbits.LPBACK = 0; // Loopback mode is disabled
    U2MODEbits.ABAUD = 0; // Autobaud is off
    U2MODEbits.URXINV = 0; // U2RX idle state is '1'
    U2MODEbits.BRGH = 1; // High speed mode (BRG generates 4 clocks per bit period)
```

```
    U2MODEbits.PDSEL = 0b00; // 8-bit data, no parity
    U2MODEbits.STSEL = 0; // 1 stop bit
    U2STAbits.URXISEL = 0b00; // Interrupt flag bit is set when a character is received
    U2MODEbits.UARTEN = 1; // Enable UART RX/TX
    U2STAbits.UTXEN = 1; // Enable the transmitter

    /* Configure Baud Rate */
    int brg = (FCY/BAUDRATE/4)-1; // Usual equation for high-speed mode (BRGH=1)
    brg = 85; // round up, since it's closer
    U2BRG = brg; // Store the value in the register

    /* Set up UART Interrupts*/
    IPC7bits.U2RXIP = 0b01; // Set U2RX interrupt priority
    IFS1bits.U2RXIF = 0; // Clear the U2RX interrupt flag
    IEC1bits.U2RXIE = 1; // Enable U2RX interrupt

}

void send_UART(char c){
    // wait for transmit buffer to be empty
    while (U2STAbits.UTXBF);
    // write character to transmit register
    U2TXREG = c;
}

unsigned char receive_UART(void){
    // Formerly uint8_t
    while (1){
        if (U2STAbits.FERR == 1){
            continue;
        }

        if (U2STAbits.OERR == 1){
            U2STAbits.OERR = 0;
            continue;
        }

        if (U2STAbits.URXDA == 1){
            return U2RXREG;
        }
    }
}

double decode_UART(void) {

    /* Initialize placeholders */
    char results[5];
    double result = 0;

    /* Initialize counting variable */
    int count = 0;

    /* Set while loop to end at SPACE */
    while (count < 5){
```

```
        results[count++] = receive_UART();
    }

    receive_UART(); // handle the space

    /* Determine Result */
    result = atof(results);

    return result;

}

void init_SPI(void) {

    SPI1CON1bits.DISSCK = 0; // Internal serial clock is enabled
    SPI1CON1bits.DISSDO = 0; // SDO1 pin is controlled by the SPI module
    SPI1CON1bits.MODE16 = 0; // 8-bit communication
    SPI1CON1bits.MSTEN = 1; // Master mode enabled
    SPI1CON1bits.SMP = 0; // Input data is sampled at the middle of data output time
    SPI1CON1bits.SSEN = 0; // SS1 is controlled by the PORT function
    SPI1CON1bits.CKE = 1; // Serial output data changes on transition
                                        // from idle clock state to active clock state
    SPI1CON1bits.CKP = 0; // Idle state for clock is low level; active high
    SPI1CON1bits.PPRE = 0b11; // Primary prescaler 64:1
    SPI1CON1bits.SPRE = 0b111; // Secondary prescaler 8:1
    SPI1STATbits.SPIEN = 1; // Enable SPI module

}

void stopMotors(void) {
    SDC1 = 0;
    PDC1 = 0;
    SDC2 = 0;
    PDC2 = 0;
    SDC3 = 0;
    PDC3 = 0;
}

int motorControl(double measured, double target){

    double difference;
    int div;
    int string;
    int flat = 0;

    if (measured < 50 || measured > 340 || target < 50){
        stopMotors();
        return 8;
    }

    /* Determine String */
    if (target < 87 && target > 61){
        string = 1;
    }
```

```
    else if (target < 116 && target > 87){
        string = 2;
    }
    else if (target < 155 && target > 116){
        string = 3;
    }
    else if (target < 207 && target > 155){
        string = 4;
    }
    else if (target < 261 && target > 207){
        string = 5;
    }
    else if (target < 330 && target > 261) {
        string = 6;
    }
    else {
        return 8;
    }

    if (measured <= target-threshold){
// if (string != 1)
            tuningDisplay(string-1, 0);
        flat = 1;
    }
    else if (measured > target+threshold){
// if (string != 1)
            tuningDisplay(string-1, 2);
        flat = 0;
    }
    else {
        tuningDisplay(string-1, 1);
        stopMotors();
        return 8;
    }

    difference = abs(measured - target);
    if (difference > 14){
        div = 1;
    }
    else if (difference > 9){
        div = 2;
    }
    else if (difference > 4){
        div = 4;
    }
    else {
        div = 8;
    }

    switch (string) {

        case 1:
            if (flat == 1) {
                s1Up();
```

```
            }
            else {
                s1Down();
            }
            break;

        case 2:
            if (flat == 1) {
                s2Up();
// s1Up();
            }
            else {
                s2Down();
// s1Down();
            }
            break;

        case 3:
            if (flat == 1) {
                s3Up();
// s1Up();
            }
            else {
                s3Down();
// s1Down();
            }
            break;

        case 4:
            if (flat == 1) {
                s4Up();
// s1Up();
            }
            else {
                s4Down();
// s1Down();
            }
            break;

        case 5:
            if (flat == 1) {
                s5Up();
// s1Up();
            }
            else {
                s5Down();
// s1Down();
            }
            break;

        case 6:
            if (flat == 1) {
                s6Up();
// s1Up();
```

```
            }
            else {
                s6Down();
// s1Down();
            }
            break;

        default:
            stopMotors();

    }


    return div;
}

int masterControl(uint16_t *fft, int numPoints, double *freqs){

    int div;
    int m2 = 0;
    int m3 = 0;
    int k, j;
    int ind[SORT] = {0};
    uint16_t mag[SORT] = {0};
    int startpoint = 50/ADC_RATE*ADC_BUFFER_SIZE;

    /* Find max points */
    for (k=startpoint; k<numPoints; k++){
        if (fft[k] > mag[0]) {
            for (j=1; j<SORT; j++){
                ind[SORT-j] = ind[SORT-j-1];
                mag[SORT-j] = mag[SORT-j-1];
            }
            ind[0] = k;
            mag[0] = fft[k];
        }
        else if (fft[k] > mag[1]) {
            for (j=1; j<SORT-1; j++){
                ind[SORT-j] = ind[SORT-j-1];
                mag[SORT-j] = mag[SORT-j-1];
            }
            ind[1] = k;
            mag[1] = fft[k];
        }
        else if (fft[k] > mag[2]) {
            for (j=1; j<SORT-2; j++){
                ind[SORT-j] = ind[SORT-j-1];
                mag[SORT-j] = mag[SORT-j-1];
            }
            ind[2] = k;
            mag[2] = fft[k];
        }
        else if (fft[k] > mag[3]) {
            for (j=1; j<SORT-3; j++){
```

```
            ind[SORT-j] = ind[SORT-j-1];
            mag[SORT-j] = mag[SORT-j-1];
        }
        ind[3] = k;
        mag[3] = fft[k];
    }
    else if (fft[k] > mag[4]) {
        for (j=1; j<SORT-4; j++){
            ind[SORT-j] = ind[SORT-j-1];
            mag[SORT-j] = mag[SORT-j-1];
        }
        ind[4] = k;
        mag[4] = fft[k];
    }
    else if (fft[k] > mag[5]) {
        ind[5] = k;
        mag[5] = fft[k];
    }
}

/* Calculate frequencies */
for (k=0; k<SORT; k++){
    peaks[k] = ind[k]*ADC_RATE/ADC_BUFFER_SIZE;
}

/* Find multiplicities */
for (k=1; k<SORT; k++){
    if ((ind[k]*2)+4 > ind[0] && (ind[k]*2)-4 < ind[0]){
        m2 = 1;
    }
    else if ((ind[k]*3)+4 > ind[0] && (ind[k]*3)-4 < ind[0]){
        m3 = 1;
    }
}

/* Set fundamental value */
if (m2 == 1) {
    fundamental = peaks[0]/2;
}
else if (m3 == 1){
    fundamental = peaks[0] / 3;
}
else {
    fundamental = peaks[0];
}


if (fundamental < 87.31-1.5*threshold && fundamental > 61.74-1.5*threshold){
    mode = 1;
}
else if (fundamental < 116.5-1.5*threshold && fundamental > 87.31-1.5*threshold){
    mode = 2;
}
```

```
else if (fundamental < 155.6-1.5*threshold && fundamental > 116.5-1.5*threshold && m2
    ↪  == 0){
    mode = 3;
}
else if (fundamental < 207.7-1.5*threshold && fundamental > 155.6-1.5*threshold){
    mode = 4;
}
else if (fundamental < 261.6-1.5*threshold && fundamental > 207.7-1.5*threshold){
    mode = 5;
}
else if (fundamental < 329.6+4*threshold && fundamental > 261.6-1.5*threshold){
    mode = 6;
}
else if (fundamental > 329.6+2*threshold) {
    mode = 7;
}
else {
    mode = 0;
}

/* Error Catching */
if (last_fundamental != 0 && abs(fundamental-last_fundamental)>25){
    mode = 7;
}

/* Magnitude Threshold */
if (mag[0] < 50){
    mode = 0;
}

switch (mode) {

    case 1:
        div = motorControl(fundamental, freqs[0]);
        break;

    case 2:
        div = motorControl(fundamental, freqs[1]);
        break;

    case 3:
        div = motorControl(fundamental, freqs[2]);
        break;

    case 4:
        div = motorControl(fundamental, freqs[3]);
        break;

    case 5:
        div = motorControl(fundamental, freqs[4]);
        break;

    case 6:
        div = motorControl(fundamental, freqs[5]);
```

```
            break;

        case 0:
            div = motorControl(0, 0);
            last_fundamental = 0;
            return div;

        default:
            div = motorControl(0, 0);
            return div;

    }

    last_fundamental = fundamental;

    return div;


}

void init_timer2(void) {

    /* Timer 2 Configuration */
    T2CONbits.TON = 0; // Turn off Timer 2
    T2CONbits.T32 = 0; // 16-bit mode
    T2CONbits.TCS = 0; // Internal clock selected
    T2CONbits.TGATE = 0;
    T2CONbits.TCKPS = 0b10; // Clock prescaler 1:1

    /* Timer 2 Value Setting*/
    TMR2 = 0; // Reset Timer 2 Value
    PR2 = 15475; // Timer 2 Period Match (around 40 Hz)

    /* Timer 2 Interrupt Handling */
    IPC1bits.T2IP = 0x01; // Set Timer 2 Interrupt Priority
    IFS0bits.T2IF = 0; // Clear Timer 2 Interrupt Flag
    IEC0bits.T2IE = 0; // Enable Timer 2 Interrupt

    /* Timer 2 Activation */
    T2CONbits.TON = 1; // Turn on Timer 2

}

void resetDisplay(void) {

  int j;
  for(j=0;j<=5;j++) {
      noteSpacing[j] = 25 + j*spacing;
  }

  tft_fillScreen(ILI9340_BLACK);
  tft_setRotation(3);
// tft_setRotation(1);
```

```
  //boxes
  int i;
  for(i=0; i<=5; i++) {
    tft_drawRect(12+i*spacing, 91, noteBoxSize, noteBoxSize, ILI9340_WHITE);
  }

  //notes in boxes
  updateDisplayText(1);

  //save button
  tft_drawRect(235, 210, 73, 30, ILI9340_WHITE);
  tft_fillRect(236, 211, 71, 28, ILI9340_BLACK);
  tft_setCursor(248, 217);
  tft_setTextSize(2);
  tft_writeString("TUNE");
}

void updateDisplayText(int noteRst) {
    int i;
    if (noteRst == 0) { //reset
    //blue out box
    tft_fillRect(12+selectedString*spacing+1,92,noteBoxSize-2,noteBoxSize-2,ILI9340_BLUE)
        ↪ ;
  } else { //select a note
    //black out boxes
      for(i=0; i<=5; i++) {
        tft_fillRect(12+i*spacing+1, 92, noteBoxSize-2, noteBoxSize-2, ILI9340_BLACK);
        tft_drawRect(12+i*spacing, 91, noteBoxSize, noteBoxSize, ILI9340_WHITE);
      }
  }
  //add new text
  tft_setTextColor(ILI9340_WHITE);
  tft_setTextSize(4);
  int start_y = 100;

  for (i=0; i<=5; i++) {
    tft_setCursor(noteSpacing[i], start_y);
    tft_setTextSize(4);
    //tft_writeString(fullNotesArray[setNotesArray[i]]); //write notes into boxes
    tft_write(fullNotesArray[setNotesArray[i]]); //write notes into boxes

    if (isSharp[i] == 1) {
      tft_setCursor(noteSpacing[i]+25,start_y);
      tft_setTextSize(1.2);
      tft_writeString("#"); //add sharps
    }
  }
}

void selectNote(void) {
  //resetDisplay();
int i;
  //clear old selection
for (i=0; i<=5; i++) {
```

```
    if (i!=selectedString) {
      tft_fillRect(10+i*spacing,80-noteBoxSize,noteBoxSize+3,noteBoxSize+3,ILI9340_BLACK);
      tft_fillRect(10+i*spacing,91+noteBoxSize+8,noteBoxSize+3,noteBoxSize+3,ILI9340_BLACK
          ↪ );
      tft_fillRect(12+i*spacing, 91, noteBoxSize, noteBoxSize, ILI9340_BLACK);
      tft_drawRect(12+i*spacing, 91, noteBoxSize, noteBoxSize, ILI9340_WHITE);
    }
  }
  //update new note selection
  tft_fillRect(12+selectedString*spacing, 91, noteBoxSize, noteBoxSize, ILI9340_BLUE);
  tft_fillTriangle(12+selectedString*spacing, 81, 12+selectedString*spacing+noteBoxSize,
      ↪ 81, 12+selectedString*spacing+0.5*noteBoxSize,81-noteBoxSize,ILI9340_WHITE);
  tft_fillTriangle(12+selectedString*spacing, 91+noteBoxSize+10, 12+selectedString*
      ↪ spacing+noteBoxSize, 91+noteBoxSize+10, 12+selectedString*spacing+0.5*
      ↪ noteBoxSize,91+(2*noteBoxSize)+10,ILI9340_WHITE);

  updateDisplayText(0);
}

void rangeMapping(int input_x, int input_y) {
  //double limits_touch[4] = {316, 0, 6, 246};
  //int limits_disp[4] = {0, 240, 0, 320};
  float dispMaxX = 320.0;
  float dispMaxY = 240.0;
  int touchX, touchY, dispX, dispY;

  // map raw touch data to touch screen coordinates
  touchX = 1.0*(input_x-TS_MINX)/(TS_MAXX-TS_MINX)*dispMaxY;
  touchY = 1.0*(input_y-TS_MINY)/(TS_MAXY-TS_MINY)*dispMaxX;

  // flip touch coordinates to map to disp coordinates
  dispX = dispMaxX - touchY;
  dispY = touchX;
// dispX = touchY;
// dispY = dispMaxY - touchX;

  touchLoc[0] = dispX;
  touchLoc[1] = dispY;
  return;
}

void startTuning(void) {

  //change tuning flag variable
  isTuning = 1;

  //reset note boxes to black
  updateDisplayText(1);

  //light up tune button
  tft_fillRect(236, 211, 71, 28, ILI9340_BLUE);
  tft_setCursor(248, 217);
  tft_setTextSize(2);
  tft_writeString("TUNE");
```

```
  //add "STOP" button
  tft_drawRect(25, 210, 73, 30, ILI9340_WHITE);
  tft_fillRect(26, 211, 71, 28, ILI9340_BLACK);
  tft_setCursor(38, 217);
  tft_setTextSize(2);
  tft_writeString("STOP");

  //remove triangles
  tft_fillTriangle(12+selectedString*spacing, 81, 12+selectedString*spacing+noteBoxSize,
      ↪ 81, 12+selectedString*spacing+0.5*noteBoxSize,81-noteBoxSize,ILI9340_BLACK);
  tft_fillTriangle(12+selectedString*spacing, 91+noteBoxSize+10, 12+selectedString*
      ↪ spacing+noteBoxSize, 91+noteBoxSize+10, 12+selectedString*spacing+0.5*
      ↪ noteBoxSize,91+(2*noteBoxSize)+10,ILI9340_BLACK);


// //display tuning progress
// int tuningStringNum = 0;
// tuningDisplay(tuningStringNum);
}

void stopTuning(void) {
  //change tuning flag variable
  isTuning = 0;

  //reset note boxes to black
  selectedString = 0;
  selectNote();

  //remove stop button
  tft_fillRect(25, 210, 73, 30, ILI9340_BLACK);

  //"unpress" tune button
  tft_fillRect(236, 211, 71, 28, ILI9340_BLACK);
  tft_setCursor(248, 217);
  tft_setTextSize(2);
  tft_writeString("TUNE");

}

void tuningDisplay(int tuningStringNum, int mode) {

    if (tuningStringNum>=0 && tuningStringNum <=5){
      switch(mode) { //check status and display accordingly
        case 0: //flat
          tft_setCursor(noteSpacing[tuningStringNum], 100);
          tft_setTextSize(4);
          tft_write(fullNotesArray[setNotesArray[tuningStringNum]]);
          tft_fillCircle(12+tuningStringNum*spacing+0.5*noteBoxSize,65,8,ILI9340_BLACK);
          tft_fillCircle(12+tuningStringNum*spacing+0.5*noteBoxSize,95+noteBoxSize+21,8,
              ↪ ILI9340_RED);
          break;
        case 1: //tuned
          tft_fillCircle(12+tuningStringNum*spacing+0.5*noteBoxSize,65,8,ILI9340_BLACK);
```

```
          tft_fillCircle(12+tuningStringNum*spacing+0.5*noteBoxSize,95+noteBoxSize+21,8,
              ↪ ILI9340_BLACK);
          tft_setTextColor(ILI9340_GREEN);
          tft_setCursor(noteSpacing[tuningStringNum], 100);
          tft_setTextSize(4);
          tft_write(fullNotesArray[setNotesArray[tuningStringNum]]);
          tft_setTextColor(ILI9340_WHITE);
          break;
        case 2: //sharp
          tft_setCursor(noteSpacing[tuningStringNum], 100);
          tft_setTextSize(4);
          tft_write(fullNotesArray[setNotesArray[tuningStringNum]]);
          tft_fillCircle(12+tuningStringNum*spacing+0.5*noteBoxSize,95+noteBoxSize+21,8,
              ↪ ILI9340_BLACK);
          tft_fillCircle(12+tuningStringNum*spacing+0.5*noteBoxSize,65,8,ILI9340_RED);
          break;
        default:
          tft_fillTriangle(12+selectedString*spacing, 81, 12+selectedString*spacing+
              ↪ noteBoxSize, 81, 12+selectedString*spacing+0.5*noteBoxSize,81-noteBoxSize
              ↪ ,ILI9340_BLACK);
          tft_fillTriangle(12+selectedString*spacing, 91+noteBoxSize+10, 12+selectedString
              ↪ *spacing+noteBoxSize, 91+noteBoxSize+10, 12+selectedString*spacing+0.5*
              ↪ noteBoxSize,91+(2*noteBoxSize)+10,ILI9340_BLACK);
          tft_fillRect(10+tuningStringNum*spacing,80-noteBoxSize,noteBoxSize+3,noteBoxSize
              ↪ +3,ILI9340_BLACK);
          tft_setTextColor(ILI9340_WHITE);
          tft_setCursor(noteSpacing[tuningStringNum], 100);
          tft_setTextSize(4);
          tft_write(fullNotesArray[setNotesArray[tuningStringNum]]);
      }

    }

        tft_writecommand(ILI9340_INVOFF);
        tft_writecommand(ILI9340_DISPON);

}

void freqPicker(void) {
   double frequencies0[6] = {82.41, 77.88, 73.42, 69.30, 65.41, 61.74};
   double frequencies1[5] = {110.0, 103.8, 98.01, 92.51, 87.31};
   double frequencies2[5] = {146.8, 138.6, 130.8, 123.5, 116.5};
   double frequencies3[5] = {196.0, 185.0, 174.6, 164.8, 155.6};
   double frequencies4[4] = {246.9, 233.1, 220.0, 207.7};
   double frequencies5[5] = {329.6, 311.1, 293.7, 277.2, 261.6};
   int i=0;

    if (setNotesArray[i]==4 && isSharp[i]==0)
        frequencies[i] = frequencies0[0];
    else if (setNotesArray[i]==3 && isSharp[i]==1)
        frequencies[i] = frequencies0[1];
    else if (setNotesArray[i]==3 && isSharp[i]==0)
        frequencies[i] = frequencies0[2];
    else if (setNotesArray[i]==2 && isSharp[i]==1)
```

```
    frequencies[i] = frequencies0[3];
else if (setNotesArray[i]==2 && isSharp[i]==0)
    frequencies[i] = frequencies0[4];
else if (setNotesArray[i]==1)
    frequencies[i] = frequencies0[5];
else
    frequencies[i] = frequencies0[0];

i=1;
if (setNotesArray[i]==0 && isSharp[i]==0)
    frequencies[i] = frequencies1[0];
else if (setNotesArray[i]==6 && isSharp[i]==1)
    frequencies[i] = frequencies1[1];
else if (setNotesArray[i]==6 && isSharp[i]==0)
    frequencies[i] = frequencies1[2];
else if (setNotesArray[i]==5 && isSharp[i]==1)
    frequencies[i] = frequencies1[3];
else if (setNotesArray[i]==5 && isSharp[i]==0)
    frequencies[i] = frequencies1[4];
else
    frequencies[i] = frequencies1[0];

i=2;
if (setNotesArray[i]==3 && isSharp[i]==0)
    frequencies[i] = frequencies2[0];
else if (setNotesArray[i]==2 && isSharp[i]==1)
    frequencies[i] = frequencies2[1];
else if (setNotesArray[i]==2 && isSharp[i]==0)
    frequencies[i] = frequencies2[2];
else if (setNotesArray[i]==1)
    frequencies[i] = frequencies2[3];
else if (setNotesArray[i]==0 && isSharp[i]==1)
    frequencies[i] = frequencies2[4];
else
    frequencies[i] = frequencies2[0];

i=3;
if (setNotesArray[i]==6 && isSharp[i]==0)
    frequencies[i] = frequencies3[0];
else if (setNotesArray[i]==5 && isSharp[i]==1)
    frequencies[i] = frequencies3[1];
else if (setNotesArray[i]==5 && isSharp[i]==0)
    frequencies[i] = frequencies3[2];
else if (setNotesArray[i]==4)
    frequencies[i] = frequencies3[3];
else if (setNotesArray[i]==3 && isSharp[i]==1)
    frequencies[i] = frequencies3[4];
else
    frequencies[i] = frequencies3[0];

i=4;
if (setNotesArray[i]==1)
    frequencies[i] = frequencies4[0];
else if (setNotesArray[i]==0 && isSharp[i]==1)
```

```
        frequencies[i] = frequencies4[1];
    else if (setNotesArray[i]==0 && isSharp[i]==0)
        frequencies[i] = frequencies4[2];
    else if (setNotesArray[i]==6 && isSharp[i]==1)
        frequencies[i] = frequencies4[3];
    else
        frequencies[i] = frequencies4[0];

    i=5;
    if (setNotesArray[i]==4 && isSharp[i]==0)
        frequencies[i] = frequencies5[0];
    else if (setNotesArray[i]==3 && isSharp[i]==1)
        frequencies[i] = frequencies5[1];
    else if (setNotesArray[i]==3 && isSharp[i]==0)
        frequencies[i] = frequencies5[2];
    else if (setNotesArray[i]==2 && isSharp[i]==1)
        frequencies[i] = frequencies5[3];
    else if (setNotesArray[i]==2 && isSharp[i]==0)
        frequencies[i] = frequencies5[4];
    else
        frequencies[i] = frequencies5[0];


}

void updateScreenMemoryArray(void) {

   double frequencies0[6] = {82.41, 77.88, 73.42, 69.30, 65.41, 61.74};
   double frequencies1[5] = {110.0, 103.8, 98.01, 92.51, 87.31};
   double frequencies2[5] = {146.8, 138.6, 130.8, 123.5, 116.5};
   double frequencies3[5] = {196.0, 185.0, 174.6, 164.8, 155.6};
   double frequencies4[4] = {246.9, 233.1, 220.0, 207.7};
   double frequencies5[5] = {329.6, 311.1, 293.7, 277.2, 261.6};


    int i;

    for (i=0; i<6; i++){
        if (abs(frequencies[0]-frequencies0[i]) < 1) {
            switch (i){
                case 0:
                    setNotesArray[0] = 4;
                    isSharp[0] = 0;
                    break;
                case 1:
                    setNotesArray[0] = 3;
                    isSharp[0] = 1;
                    break;
                case 2:
                    setNotesArray[0] = 3;
                    isSharp[0] = 0;
                    break;
                case 3:
                    setNotesArray[0] = 2;
```

```
                isSharp[0] = 1;
                break;
            case 4:
                setNotesArray[0] = 2;
                isSharp[0] = 0;
                break;
            case 5:
                setNotesArray[0] = 1;
                isSharp[0] = 0;
                break;
        }
    }
}

for (i=0; i<5; i++){
    if (abs(frequencies[1]-frequencies1[i]) < 1) {
        switch (i){
            case 0:
                setNotesArray[1] = 0;
                isSharp[1] = 0;
                break;
            case 1:
                setNotesArray[1] = 6;
                isSharp[1] = 1;
                break;
            case 2:
                setNotesArray[1] = 6;
                isSharp[1] = 0;
                break;
            case 3:
                setNotesArray[1] = 5;
                isSharp[1] = 1;
                break;
            case 4:
                setNotesArray[1] = 5;
                isSharp[1] = 0;
                break;
        }
    }
}

for (i=0; i<5; i++){
    if (abs(frequencies[2]-frequencies2[i]) < 1) {
        switch (i){
            case 0:
                setNotesArray[2] = 3;
                isSharp[2] = 0;
                break;
            case 1:
                setNotesArray[2] = 2;
                isSharp[2] = 1;
                break;
            case 2:
                setNotesArray[2] = 2;
```

```
                isSharp[2] = 0;
                break;
            case 3:
                setNotesArray[2] = 1;
                isSharp[2] = 0;
                break;
            case 4:
                setNotesArray[2] = 0;
                isSharp[2] = 1;
                break;
        }
    }
}

for (i=0; i<5; i++){
    if (abs(frequencies[3]-frequencies3[i]) < 1) {
        switch (i){
            case 0:
                setNotesArray[3] = 6;
                isSharp[3] = 0;
                break;
            case 1:
                setNotesArray[3] = 5;
                isSharp[3] = 1;
                break;
            case 2:
                setNotesArray[3] = 5;
                isSharp[3] = 0;
                break;
            case 3:
                setNotesArray[3] = 4;
                isSharp[3] = 0;
                break;
            case 4:
                setNotesArray[3] = 3;
                isSharp[3] = 1;
                break;
        }
    }
}

for (i=0; i<4; i++){
    if (abs(frequencies[4]-frequencies4[i]) < 1) {
        switch (i){
            case 0:
                setNotesArray[4] = 1;
                isSharp[4] = 0;
                break;
            case 1:
                setNotesArray[4] = 0;
                isSharp[4] = 1;
                break;
            case 2:
                setNotesArray[4] = 0;
```

```
                isSharp[4] = 0;
                break;
            case 3:
                setNotesArray[4] = 6;
                isSharp[4] = 1;
                break;
        }
    }
}

for (i=0; i<5; i++){
    if (abs(frequencies[5]-frequencies5[i]) < 1) {
        switch (i){
            case 0:
                setNotesArray[5] = 4;
                isSharp[5] = 0;
                break;
            case 1:
                setNotesArray[5] = 3;
                isSharp[5] = 1;
                break;
            case 2:
                setNotesArray[5] = 3;
                isSharp[5] = 0;
                break;
            case 3:
                setNotesArray[5] = 2;
                isSharp[5] = 1;
                break;
            case 4:
                setNotesArray[5] = 2;
                isSharp[5] = 0;
                break;
        }
    }
}

}
```

C.1.2 fft.c

```
/*
 * File: fft.c
 * Author: Kevin Schneier
 *
 */

#include "fft.h"

/*****************************************************************************
 Function from fft.s file.
 *****************************************************************************/
extern void FFT2(int16_t* pFirstRe, int16_t* pFirstIm, int16_t* pSecondRe, int16_t*
    ↪ pSecondIm, int16_t wRe, int16_t wIm);

/*****************************************************************************
 FFT harmonics are permutated after the transform.
 This table is sued to order them.
 *****************************************************************************/
const int16_t bitPermutationTable[FFT_POINTS_NUMBER] =

#if (FFT_POINTS_NUMBER == 8)
{0,4,2,6,1,5,3,7};

#elif (FFT_POINTS_NUMBER == 16)
{0,8,4,12,2,10,6,14,1,9,5,13,3,11,7,15};

#elif (FFT_POINTS_NUMBER == 32)
{0,16,8,24,4,20,12,28,2,18,10,26,6,22,14,30,1,17,9,25,5,21,13,29,3,19,11,27,7,23,15,31};

#elif (FFT_POINTS_NUMBER == 64)
{0,32,16,48,8,40,24,56,4,36,20,52,12,44,28,60,2,34,18,50,10,42,26,58,6,38,22,54,14,46,30,62,1,33,17,49,9
    ↪
45,29,61,3,35,19,51,11,43,27,59,7,39,23,55,15,47,31,63};

#elif (FFT_POINTS_NUMBER == 128)
{0,64,32,96,16,80,48,112,8,72,40,104,24,88,56,120,4,68,36,100,20,84,52,116,12,76,44,108,28,92,60,124,2,6
    ↪
74,42,106,26,90,58,122,6,70,38,102,22,86,54,118,14,78,46,110,30,94,62,126,1,65,33,97,17,81,49,113,9,73,4
    ↪
37,101,21,85,53,117,13,77,45,109,29,93,61,125,3,67,35,99,19,83,51,115,11,75,43,107,27,91,59,123,7,71,39,
    ↪
47,111,31,95,63,127};

#elif (FFT_POINTS_NUMBER == 256)
{0,128,64,192,32,160,96,224,16,144,80,208,48,176,112,240,8,136,72,200,40,168,104,232,24,152,88,216,56,18
    ↪
36,164,100,228,20,148,84,212,52,180,116,244,12,140,76,204,44,172,108,236,28,156,92,220,60,188,124,252,2,
    ↪
226,18,146,82,210,50,178,114,242,10,138,74,202,42,170,106,234,26,154,90,218,58,186,122,250,6,134,70,198,
    ↪
86,214,54,182,118,246,14,142,78,206,46,174,110,238,30,158,94,222,62,190,126,254,1,129,65,193,33,161,97,2
    ↪
```

```
113,241,9,137,73,201,41,169,105,233,25,153,89,217,57,185,121,249,5,133,69,197,37,165,101,229,21,149,85,2
    ↪
77,205,45,173,109,237,29,157,93,221,61,189,125,253,3,131,67,195,35,163,99,227,19,147,83,211,51,179,115,2
    ↪
107,235,27,155,91,219,59,187,123,251,7,135,71,199,39,167,103,231,23,151,87,215,55,183,119,247,15,143,79,
    ↪
159,95,223,63,191,127,255};

#elif (FFT_POINTS_NUMBER == 512)
{0,256,128,384,64,320,192,448,32,288,160,416,96,352,224,480,16,272,144,400,80,336,208,464,48,304,176,432
    ↪
136,392,72,328,200,456,40,296,168,424,104,360,232,488,24,280,152,408,88,344,216,472,56,312,184,440,120,3
    ↪
68,324,196,452,36,292,164,420,100,356,228,484,20,276,148,404,84,340,212,468,52,308,180,436,116,372,244,5
    ↪
204,460,44,300,172,428,108,364,236,492,28,284,156,412,92,348,220,476,60,316,188,444,124,380,252,508,2,25
    ↪
34,290,162,418,98,354,226,482,18,274,146,402,82,338,210,466,50,306,178,434,114,370,242,498,10,266,138,39
    ↪
170,426,106,362,234,490,26,282,154,410,90,346,218,474,58,314,186,442,122,378,250,506,6,262,134,390,70,32
    ↪
102,358,230,486,22,278,150,406,86,342,214,470,54,310,182,438,118,374,246,502,14,270,142,398,78,334,206,4
    ↪
238,494,30,286,158,414,94,350,222,478,62,318,190,446,126,382,254,510,1,257,129,385,65,321,193,449,33,289
    ↪
17,273,145,401,81,337,209,465,49,305,177,433,113,369,241,497,9,265,137,393,73,329,201,457,41,297,169,425
    ↪
153,409,89,345,217,473,57,313,185,441,121,377,249,505,5,261,133,389,69,325,197,453,37,293,165,421,101,35
    ↪
85,341,213,469,53,309,181,437,117,373,245,501,13,269,141,397,77,333,205,461,45,301,173,429,109,365,237,4
    ↪
221,477,61,317,189,445,125,381,253,509,3,259,131,387,67,323,195,451,35,291,163,419,99,355,227,483,19,275
    ↪
51,307,179,435,115,371,243,499,11,267,139,395,75,331,203,459,43,299,171,427,107,363,235,491,27,283,155,4
    ↪
187,443,123,379,251,507,7,263,135,391,71,327,199,455,39,295,167,423,103,359,231,487,23,279,151,407,87,34
    ↪
119,375,247,503,15,271,143,399,79,335,207,463,47,303,175,431,111,367,239,495,31,287,159,415,95,351,223,4
    ↪
255,511};

#elif (FFT_POINTS_NUMBER == 1024)
{0,512,256,768,128,640,384,896,64,576,320,832,192,704,448,960,32,544,288,800,160,672,416,928,96,608,352,
    ↪
16,528,272,784,144,656,400,912,80,592,336,848,208,720,464,976,48,560,304,816,176,688,432,944,112,624,368
    ↪
8,520,264,776,136,648,392,904,72,584,328,840,200,712,456,968,40,552,296,808,168,680,424,936,104,616,360,
    ↪
24,536,280,792,152,664,408,920,88,600,344,856,216,728,472,984,56,568,312,824,184,696,440,952,120,632,376
    ↪
4,516,260,772,132,644,388,900,68,580,324,836,196,708,452,964,36,548,292,804,164,676,420,932,100,612,356,
    ↪
20,532,276,788,148,660,404,916,84,596,340,852,212,724,468,980,52,564,308,820,180,692,436,948,116,628,372
    ↪
```

```
12,524,268,780,140,652,396,908,76,588,332,844,204,716,460,972,44,556,300,812,172,684,428,940,108,620,364
    ↪
28,540,284,796,156,668,412,924,92,604,348,860,220,732,476,988,60,572,316,828,188,700,444,956,124,636,380
    ↪
2,514,258,770,130,642,386,898,66,578,322,834,194,706,450,962,34,546,290,802,162,674,418,930,98,610,354,8
    ↪
530,274,786,146,658,402,914,82,594,338,850,210,722,466,978,50,562,306,818,178,690,434,946,114,626,370,88
    ↪
522,266,778,138,650,394,906,74,586,330,842,202,714,458,970,42,554,298,810,170,682,426,938,106,618,362,87
    ↪
538,282,794,154,666,410,922,90,602,346,858,218,730,474,986,58,570,314,826,186,698,442,954,122,634,378,89
    ↪
518,262,774,134,646,390,902,70,582,326,838,198,710,454,966,38,550,294,806,166,678,422,934,102,614,358,87
    ↪
534,278,790,150,662,406,918,86,598,342,854,214,726,470,982,54,566,310,822,182,694,438,950,118,630,374,88
    ↪
526,270,782,142,654,398,910,78,590,334,846,206,718,462,974,46,558,302,814,174,686,430,942,110,622,366,87
    ↪
542,286,798,158,670,414,926,94,606,350,862,222,734,478,990,62,574,318,830,190,702,446,958,126,638,382,89
    ↪
513,257,769,129,641,385,897,65,577,321,833,193,705,449,961,33,545,289,801,161,673,417,929,97,609,353,865
    ↪
529,273,785,145,657,401,913,81,593,337,849,209,721,465,977,49,561,305,817,177,689,433,945,113,625,369,88
    ↪
521,265,777,137,649,393,905,73,585,329,841,201,713,457,969,41,553,297,809,169,681,425,937,105,617,361,87
    ↪
537,281,793,153,665,409,921,89,601,345,857,217,729,473,985,57,569,313,825,185,697,441,953,121,633,377,88
    ↪
517,261,773,133,645,389,901,69,581,325,837,197,709,453,965,37,549,293,805,165,677,421,933,101,613,357,86
    ↪
533,277,789,149,661,405,917,85,597,341,853,213,725,469,981,53,565,309,821,181,693,437,949,117,629,373,88
    ↪
525,269,781,141,653,397,909,77,589,333,845,205,717,461,973,45,557,301,813,173,685,429,941,109,621,365,87
    ↪
541,285,797,157,669,413,925,93,605,349,861,221,733,477,989,61,573,317,829,189,701,445,957,125,637,381,89
    ↪
515,259,771,131,643,387,899,67,579,323,835,195,707,451,963,35,547,291,803,163,675,419,931,99,611,355,867
    ↪
531,275,787,147,659,403,915,83,595,339,851,211,723,467,979,51,563,307,819,179,691,435,947,115,627,371,88
    ↪
523,267,779,139,651,395,907,75,587,331,843,203,715,459,971,43,555,299,811,171,683,427,939,107,619,363,87
    ↪
539,283,795,155,667,411,923,91,603,347,859,219,731,475,987,59,571,315,827,187,699,443,955,123,635,379,89
    ↪
519,263,775,135,647,391,903,71,583,327,839,199,711,455,967,39,551,295,807,167,679,423,935,103,615,359,87
    ↪
535,279,791,151,663,407,919,87,599,343,855,215,727,471,983,55,567,311,823,183,695,439,951,119,631,375,88
    ↪
527,271,783,143,655,399,911,79,591,335,847,207,719,463,975,47,559,303,815,175,687,431,943,111,623,367,87
    ↪
543,287,799,159,671,415,927,95,607,351,863,223,735,479,991,63,575,319,831,191,703,447,959,127,639,383,89
    ↪

#elif (FFT_POINTS_NUMBER == 2048)
```

{0,1024,512,1536,256,1280,768,1792,128,1152,640,1664,384,1408,896,1920,64,1088,576,1600,320,1344,832,185
↪
448,1472,960,1984,32,1056,544,1568,288,1312,800,1824,160,1184,672,1696,416,1440,928,1952,96,1120,608,163
↪
224,1248,736,1760,480,1504,992,2016,16,1040,528,1552,272,1296,784,1808,144,1168,656,1680,400,1424,912,19
↪
336,1360,848,1872,208,1232,720,1744,464,1488,976,2000,48,1072,560,1584,304,1328,816,1840,176,1200,688,17
↪
112,1136,624,1648,368,1392,880,1904,240,1264,752,1776,496,1520,1008,2032,8,1032,520,1544,264,1288,776,18
↪
392,1416,904,1928,72,1096,584,1608,328,1352,840,1864,200,1224,712,1736,456,1480,968,1992,40,1064,552,157
↪
168,1192,680,1704,424,1448,936,1960,104,1128,616,1640,360,1384,872,1896,232,1256,744,1768,488,1512,1000,
↪
280,1304,792,1816,152,1176,664,1688,408,1432,920,1944,88,1112,600,1624,344,1368,856,1880,216,1240,728,17
↪
56,1080,568,1592,312,1336,824,1848,184,1208,696,1720,440,1464,952,1976,120,1144,632,1656,376,1400,888,19
↪
504,1528,1016,2040,4,1028,516,1540,260,1284,772,1796,132,1156,644,1668,388,1412,900,1924,68,1092,580,160
↪
1220,708,1732,452,1476,964,1988,36,1060,548,1572,292,1316,804,1828,164,1188,676,1700,420,1444,932,1956,1
↪
1380,868,1892,228,1252,740,1764,484,1508,996,2020,20,1044,532,1556,276,1300,788,1812,148,1172,660,1684,4
↪
596,1620,340,1364,852,1876,212,1236,724,1748,468,1492,980,2004,52,1076,564,1588,308,1332,820,1844,180,12
↪
1972,116,1140,628,1652,372,1396,884,1908,244,1268,756,1780,500,1524,1012,2036,12,1036,524,1548,268,1292,
↪
1676,396,1420,908,1932,76,1100,588,1612,332,1356,844,1868,204,1228,716,1740,460,1484,972,1996,44,1068,55
↪
172,1196,684,1708,428,1452,940,1964,108,1132,620,1644,364,1388,876,1900,236,1260,748,1772,492,1516,1004,
↪
284,1308,796,1820,156,1180,668,1692,412,1436,924,1948,92,1116,604,1628,348,1372,860,1884,220,1244,732,17
↪
1084,572,1596,316,1340,828,1852,188,1212,700,1724,444,1468,956,1980,124,1148,636,1660,380,1404,892,1916,
↪
1532,1020,2044,2,1026,514,1538,258,1282,770,1794,130,1154,642,1666,386,1410,898,1922,66,1090,578,1602,32
↪
706,1730,450,1474,962,1986,34,1058,546,1570,290,1314,802,1826,162,1186,674,1698,418,1442,930,1954,98,112
↪
1890,226,1250,738,1762,482,1506,994,2018,18,1042,530,1554,274,1298,786,1810,146,1170,658,1682,402,1426,9
↪
1618,338,1362,850,1874,210,1234,722,1746,466,1490,978,2002,50,1074,562,1586,306,1330,818,1842,178,1202,6
↪
1970,114,1138,626,1650,370,1394,882,1906,242,1266,754,1778,498,1522,1010,2034,10,1034,522,1546,266,1290,
↪
1674,394,1418,906,1930,74,1098,586,1610,330,1354,842,1866,202,1226,714,1738,458,1482,970,1994,42,1066,55
↪
170,1194,682,1706,426,1450,938,1962,106,1130,618,1642,362,1386,874,1898,234,1258,746,1770,490,1514,1002,
↪
1306,794,1818,154,1178,666,1690,410,1434,922,1946,90,1114,602,1626,346,1370,858,1882,218,1242,730,1754,4
↪
570,1594,314,1338,826,1850,186,1210,698,1722,442,1466,954,1978,122,1146,634,1658,378,1402,890,1914,250,1
↪

```
1018,2042,6,1030,518,1542,262,1286,774,1798,134,1158,646,1670,390,1414,902,1926,70,1094,582,1606,326,135
↪
1734,454,1478,966,1990,38,1062,550,1574,294,1318,806,1830,166,1190,678,1702,422,1446,934,1958,102,1126,6
↪
1894,230,1254,742,1766,486,1510,998,2022,22,1046,534,1558,278,1302,790,1814,150,1174,662,1686,406,1430,9
↪
342,1366,854,1878,214,1238,726,1750,470,1494,982,2006,54,1078,566,1590,310,1334,822,1846,182,1206,694,17
↪
1142,630,1654,374,1398,886,1910,246,1270,758,1782,502,1526,1014,2038,14,1038,526,1550,270,1294,782,1806,
↪
1422,910,1934,78,1102,590,1614,334,1358,846,1870,206,1230,718,1742,462,1486,974,1998,46,1070,558,1582,30
↪
686,1710,430,1454,942,1966,110,1134,622,1646,366,1390,878,1902,238,1262,750,1774,494,1518,1006,2030,30,1
↪
798,1822,158,1182,670,1694,414,1438,926,1950,94,1118,606,1630,350,1374,862,1886,222,1246,734,1758,478,15
↪
1598,318,1342,830,1854,190,1214,702,1726,446,1470,958,1982,126,1150,638,1662,382,1406,894,1918,254,1278,
↪
2046,1,1025,513,1537,257,1281,769,1793,129,1153,641,1665,385,1409,897,1921,65,1089,577,1601,321,1345,833
↪
449,1473,961,1985,33,1057,545,1569,289,1313,801,1825,161,1185,673,1697,417,1441,929,1953,97,1121,609,163
↪
1249,737,1761,481,1505,993,2017,17,1041,529,1553,273,1297,785,1809,145,1169,657,1681,401,1425,913,1937,8
↪
849,1873,209,1233,721,1745,465,1489,977,2001,49,1073,561,1585,305,1329,817,1841,177,1201,689,1713,433,14
↪
1649,369,1393,881,1905,241,1265,753,1777,497,1521,1009,2033,9,1033,521,1545,265,1289,777,1801,137,1161,6
↪
73,1097,585,1609,329,1353,841,1865,201,1225,713,1737,457,1481,969,1993,41,1065,553,1577,297,1321,809,183
↪
1449,937,1961,105,1129,617,1641,361,1385,873,1897,233,1257,745,1769,489,1513,1001,2025,25,1049,537,1561,
↪
1177,665,1689,409,1433,921,1945,89,1113,601,1625,345,1369,857,1881,217,1241,729,1753,473,1497,985,2009,5
↪
825,1849,185,1209,697,1721,441,1465,953,1977,121,1145,633,1657,377,1401,889,1913,249,1273,761,1785,505,1
↪
1541,261,1285,773,1797,133,1157,645,1669,389,1413,901,1925,69,1093,581,1605,325,1349,837,1861,197,1221,7
↪
37,1061,549,1573,293,1317,805,1829,165,1189,677,1701,421,1445,933,1957,101,1125,613,1637,357,1381,869,18
↪
1509,997,2021,21,1045,533,1557,277,1301,789,1813,149,1173,661,1685,405,1429,917,1941,85,1109,597,1621,34
↪
725,1749,469,1493,981,2005,53,1077,565,1589,309,1333,821,1845,181,1205,693,1717,437,1461,949,1973,117,11
↪
885,1909,245,1269,757,1781,501,1525,1013,2037,13,1037,525,1549,269,1293,781,1805,141,1165,653,1677,397,1
↪
589,1613,333,1357,845,1869,205,1229,717,1741,461,1485,973,1997,45,1069,557,1581,301,1325,813,1837,173,11
↪
941,1965,109,1133,621,1645,365,1389,877,1901,237,1261,749,1773,493,1517,1005,2029,29,1053,541,1565,285,1
↪
669,1693,413,1437,925,1949,93,1117,605,1629,349,1373,861,1885,221,1245,733,1757,477,1501,989,2013,61,108
↪
1853,189,1213,701,1725,445,1469,957,1981,125,1149,637,1661,381,1405,893,1917,253,1277,765,1789,509,1533,
↪
```

```
1539,259,1283,771,1795,131,1155,643,1667,387,1411,899,1923,67,1091,579,1603,323,1347,835,1859,195,1219,7
    ↪
1987,35,1059,547,1571,291,1315,803,1827,163,1187,675,1699,419,1443,931,1955,99,1123,611,1635,355,1379,86
    ↪
483,1507,995,2019,19,1043,531,1555,275,1299,787,1811,147,1171,659,1683,403,1427,915,1939,83,1107,595,161
    ↪
1235,723,1747,467,1491,979,2003,51,1075,563,1587,307,1331,819,1843,179,1203,691,1715,435,1459,947,1971,1
    ↪
1395,883,1907,243,1267,755,1779,499,1523,1011,2035,11,1035,523,1547,267,1291,779,1803,139,1163,651,1675,
    ↪
1099,587,1611,331,1355,843,1867,203,1227,715,1739,459,1483,971,1995,43,1067,555,1579,299,1323,811,1835,1
    ↪
939,1963,107,1131,619,1643,363,1387,875,1899,235,1259,747,1771,491,1515,1003,2027,27,1051,539,1563,283,1
    ↪
667,1691,411,1435,923,1947,91,1115,603,1627,347,1371,859,1883,219,1243,731,1755,475,1499,987,2011,59,108
    ↪
1851,187,1211,699,1723,443,1467,955,1979,123,1147,635,1659,379,1403,891,1915,251,1275,763,1787,507,1531,
    ↪
1543,263,1287,775,1799,135,1159,647,1671,391,1415,903,1927,71,1095,583,1607,327,1351,839,1863,199,1223,7
    ↪
39,1063,551,1575,295,1319,807,1831,167,1191,679,1703,423,1447,935,1959,103,1127,615,1639,359,1383,871,18
    ↪
1511,999,2023,23,1047,535,1559,279,1303,791,1815,151,1175,663,1687,407,1431,919,1943,87,1111,599,1623,34
    ↪
727,1751,471,1495,983,2007,55,1079,567,1591,311,1335,823,1847,183,1207,695,1719,439,1463,951,1975,119,11
    ↪
887,1911,247,1271,759,1783,503,1527,1015,2039,15,1039,527,1551,271,1295,783,1807,143,1167,655,1679,399,1
    ↪
591,1615,335,1359,847,1871,207,1231,719,1743,463,1487,975,1999,47,1071,559,1583,303,1327,815,1839,175,11
    ↪
943,1967,111,1135,623,1647,367,1391,879,1903,239,1263,751,1775,495,1519,1007,2031,31,1055,543,1567,287,1
    ↪
671,1695,415,1439,927,1951,95,1119,607,1631,351,1375,863,1887,223,1247,735,1759,479,1503,991,2015,63,108
    ↪
831,1855,191,1215,703,1727,447,1471,959,1983,127,1151,639,1663,383,1407,895,1919,255,1279,767,1791,511,1
    ↪
#else
{};
#error "The selected points number FFT_POINTS_NUMBER in fft.h is not supported."
#endif

/*****************************************************************************
 Twiddle Coefficients.
 *****************************************************************************/
const int16_t twiddleCoefficients[3*FFT_POINTS_NUMBER/4] =

#if (FFT_POINTS_NUMBER == 8)
{0,23170,32767,23170,0,-23170};

#elif (FFT_POINTS_NUMBER == 16)
{0,12539,23170,30273,32767,30273,23170,12539,0,-12539,-23170,-30273};

#elif (FFT_POINTS_NUMBER == 32)
{0,6393,12539,18204,23170,27245,30273,32137,32767,32137,30273,27245,23170,18204,12539,6393,0,-6393,-1253
```

```
    ↪
-27245,-30273,-32137};

#elif (FFT_POINTS_NUMBER == 64)
{0,3212,6393,9512,12539,15446,18204,20787,23170,25329,27245,28898,30273,31356,32137,32609,32767,32609,32
    ↪
28898,27245,25329,23170,20787,18204,15446,12539,9512,6393,3212,0,-3212,-6393,-9512,-12539,-15446,-18204,
    ↪
-25329,-27245,-28898,-30273,-31356,-32137,-32609};

#elif (FFT_POINTS_NUMBER == 128)
{0,1608,3212,4808,6393,7962,9512,11039,12539,14010,15446,16846,18204,19519,20787,22005,23170,24279,25329
    ↪
28898,29621,30273,30852,31356,31785,32137,32412,32609,32728,32767,32728,32609,32412,32137,31785,31356,30
    ↪
28898,28105,27245,26319,25329,24279,23170,22005,20787,19519,18204,16846,15446,14010,12539,11039,9512,796
    ↪
1608,0,-1608,-3212,-4808,-6393,-7962,-9512,-11039,-12539,-14010,-15446,-16846,-18204,-19519,-20787,-2200
    ↪
-25329,-26319,-27245,-28105,-28898,-29621,-30273,-30852,-31356,-31785,-32137,-32412,-32609,-32728};
    ↪

#elif (FFT_POINTS_NUMBER == 256)
{0,804,1608,2410,3212,4011,4808,5602,6393,7179,7962,8739,9512,10278,11039,11793,12539,13279,14010,14732,
    ↪
16846,17530,18204,18868,19519,20159,20787,21403,22005,22594,23170,23731,24279,24811,25329,25832,26319,26
    ↪
27683,28105,28510,28898,29268,29621,29956,30273,30571,30852,31113,31356,31580,31785,31971,32137,32285,32
    ↪
32609,32678,32728,32757,32767,32757,32728,32678,32609,32521,32412,32285,32137,31971,31785,31580,31356,31
    ↪
30571,30273,29956,29621,29268,28898,28510,28105,27683,27245,26790,26319,25832,25329,24811,24279,23731,23
    ↪
22005,21403,20787,20159,19519,18868,18204,17530,16846,16151,15446,14732,14010,13279,12539,11793,11039,10
    ↪
8739,7962,7179,6393,5602,4808,4011,3212,2410,1608,804,0,-804,-1608,-2410,-3212,-4011,-4808,-5602,-6393,-
    ↪
-8739,-9512,-10278,-11039,-11793,-12539,-13279,-14010,-14732,-15446,-16151,-16846,-17530,-18204,-18868,-
    ↪
-20787,-21403,-22005,-22594,-23170,-23731,-24279,-24811,-25329,-25832,-26319,-26790,-27245,-27683,-28105
    ↪
-29268,-29621,-29956,-30273,-30571,-30852,-31113,-31356,-31580,-31785,-31971,-32137,-32285,-32412,-32521
    ↪
-32728,-32757};

#elif (FFT_POINTS_NUMBER == 512)
{0,402,804,1206,1608,2009,2410,2811,3212,3612,4011,4410,4808,5205,5602,5998,6393,6786,7179,7571,7962,835
    ↪
9512,9896,10278,10659,11039,11417,11793,12167,12539,12910,13279,13645,14010,14372,14732,15090,15446,1580
    ↪
16846,17189,17530,17869,18204,18537,18868,19195,19519,19841,20159,20475,20787,21096,21403,21705,22005,22
    ↪
23170,23452,23731,24007,24279,24547,24811,25072,25329,25582,25832,26077,26319,26556,26790,27019,27245,27
    ↪
```

```
28105,28310,28510,28706,28898,29085,29268,29447,29621,29791,29956,30117,30273,30424,30571,30714,30852,30
    ↪
31356,31470,31580,31685,31785,31880,31971,32057,32137,32213,32285,32351,32412,32469,32521,32567,32609,32
    ↪
32728,32745,32757,32765,32767,32765,32757,32745,32728,32705,32678,32646,32609,32567,32521,32469,32412,32
    ↪
32137,32057,31971,31880,31785,31685,31580,31470,31356,31237,31113,30985,30852,30714,30571,30424,30273,30
    ↪
29621,29447,29268,29085,28898,28706,28510,28310,28105,27896,27683,27466,27245,27019,26790,26556,26319,26
    ↪
25329,25072,24811,24547,24279,24007,23731,23452,23170,22884,22594,22301,22005,21705,21403,21096,20787,20
    ↪
19519,19195,18868,18537,18204,17869,17530,17189,16846,16499,16151,15800,15446,15090,14732,14372,14010,13
    ↪
12539,12167,11793,11417,11039,10659,10278,9896,9512,9126,8739,8351,7962,7571,7179,6786,6393,5998,5602,52
    ↪
3612,3212,2811,2410,2009,1608,1206,804,402,0,-402,-804,-1206,-1608,-2009,-2410,-2811,-3212,-3612,-4011,-
    ↪
-5602,-5998,-6393,-6786,-7179,-7571,-7962,-8351,-8739,-9126,-9512,-9896,-10278,-10659,-11039,-11417,-117
    ↪
-12910,-13279,-13645,-14010,-14372,-14732,-15090,-15446,-15800,-16151,-16499,-16846,-17189,-17530,-17869
    ↪
-18868,-19195,-19519,-19841,-20159,-20475,-20787,-21096,-21403,-21705,-22005,-22301,-22594,-22884,-23170
    ↪
-24007,-24279,-24547,-24811,-25072,-25329,-25582,-25832,-26077,-26319,-26556,-26790,-27019,-27245,-27466
    ↪
-28105,-28310,-28510,-28706,-28898,-29085,-29268,-29447,-29621,-29791,-29956,-30117,-30273,-30424,-30571
    ↪
-30985,-31113,-31237,-31356,-31470,-31580,-31685,-31785,-31880,-31971,-32057,-32137,-32213,-32285,-32351
    ↪
-32521,-32567,-32609,-32646,-32678,-32705,-32728,-32745,-32757,-32765};

#elif (FFT_POINTS_NUMBER == 1024)
{0,201,402,603,804,1005,1206,1407,1608,1809,2009,2210,2410,2611,2811,3012,3212,3412,3612,3811,4011,4210,
    ↪
5007,5205,5404,5602,5800,5998,6195,6393,6590,6786,6983,7179,7375,7571,7767,7962,8157,8351,8545,8739,8933
    ↪
9704,9896,10087,10278,10469,10659,10849,11039,11228,11417,11605,11793,11980,12167,12353,12539,12725,1291
    ↪
13645,13828,14010,14191,14372,14553,14732,14912,15090,15269,15446,15623,15800,15976,16151,16325,16499,16
    ↪
17189,17360,17530,17700,17869,18037,18204,18371,18537,18703,18868,19032,19195,19357,19519,19680,19841,20
    ↪
20475,20631,20787,20942,21096,21250,21403,21554,21705,21856,22005,22154,22301,22448,22594,22739,22884,23
    ↪
23452,23592,23731,23870,24007,24143,24279,24413,24547,24680,24811,24942,25072,25201,25329,25456,25582,25
    ↪
26077,26198,26319,26438,26556,26674,26790,26905,27019,27133,27245,27356,27466,27575,27683,27790,27896,28
    ↪
28310,28411,28510,28609,28706,28803,28898,28992,29085,29177,29268,29358,29447,29534,29621,29706,29791,29
    ↪
30117,30195,30273,30349,30424,30498,30571,30643,30714,30783,30852,30919,30985,31050,31113,31176,31237,31
    ↪
31470,31526,31580,31633,31685,31736,31785,31833,31880,31926,31971,32014,32057,32098,32137,32176,32213,32
```

```
    ↪
32351,32382,32412,32441,32469,32495,32521,32545,32567,32589,32609,32628,32646,32663,32678,32692,32705,32
    ↪
32745,32752,32757,32761,32765,32766,32767,32766,32765,32761,32757,32752,32745,32737,32728,32717,32705,32
    ↪
32646,32628,32609,32589,32567,32545,32521,32495,32469,32441,32412,32382,32351,32318,32285,32250,32213,32
    ↪
32057,32014,31971,31926,31880,31833,31785,31736,31685,31633,31580,31526,31470,31414,31356,31297,31237,31
    ↪
30985,30919,30852,30783,30714,30643,30571,30498,30424,30349,30273,30195,30117,30037,29956,29874,29791,29
    ↪
29447,29358,29268,29177,29085,28992,28898,28803,28706,28609,28510,28411,28310,28208,28105,28001,27896,27
    ↪
27466,27356,27245,27133,27019,26905,26790,26674,26556,26438,26319,26198,26077,25955,25832,25708,25582,25
    ↪
25072,24942,24811,24680,24547,24413,24279,24143,24007,23870,23731,23592,23452,23311,23170,23027,22884,22
    ↪
22301,22154,22005,21856,21705,21554,21403,21250,21096,20942,20787,20631,20475,20317,20159,20000,19841,19
    ↪
19195,19032,18868,18703,18537,18371,18204,18037,17869,17700,17530,17360,17189,17018,16846,16673,16499,16
    ↪
15800,15623,15446,15269,15090,14912,14732,14553,14372,14191,14010,13828,13645,13462,13279,13094,12910,12
    ↪
12167,11980,11793,11605,11417,11228,11039,10849,10659,10469,10278,10087,9896,9704,9512,9319,9126,8933,87
    ↪
7962,7767,7571,7375,7179,6983,6786,6590,6393,6195,5998,5800,5602,5404,5205,5007,4808,4609,4410,4210,4011
    ↪
3212,3012,2811,2611,2410,2210,2009,1809,1608,1407,1206,1005,804,603,402,201,0,-201,-402,-603,-804,-1005,
    ↪
-1809,-2009,-2210,-2410,-2611,-2811,-3012,-3212,-3412,-3612,-3811,-4011,-4210,-4410,-4609,-4808,-5007,-5
    ↪
-5800,-5998,-6195,-6393,-6590,-6786,-6983,-7179,-7375,-7571,-7767,-7962,-8157,-8351,-8545,-8739,-8933,-9
    ↪
-9704,-9896,-10087,-10278,-10469,-10659,-10849,-11039,-11228,-11417,-11605,-11793,-11980,-12167,-12353,-
    ↪
-13094,-13279,-13462,-13645,-13828,-14010,-14191,-14372,-14553,-14732,-14912,-15090,-15269,-15446,-15623
    ↪
-16325,-16499,-16673,-16846,-17018,-17189,-17360,-17530,-17700,-17869,-18037,-18204,-18371,-18537,-18703
    ↪
-19357,-19519,-19680,-19841,-20000,-20159,-20317,-20475,-20631,-20787,-20942,-21096,-21250,-21403,-21554
    ↪
-22154,-22301,-22448,-22594,-22739,-22884,-23027,-23170,-23311,-23452,-23592,-23731,-23870,-24007,-24143
    ↪
-24680,-24811,-24942,-25072,-25201,-25329,-25456,-25582,-25708,-25832,-25955,-26077,-26198,-26319,-26438
    ↪
-26905,-27019,-27133,-27245,-27356,-27466,-27575,-27683,-27790,-27896,-28001,-28105,-28208,-28310,-28411
    ↪
-28803,-28898,-28992,-29085,-29177,-29268,-29358,-29447,-29534,-29621,-29706,-29791,-29874,-29956,-30037
    ↪
-30349,-30424,-30498,-30571,-30643,-30714,-30783,-30852,-30919,-30985,-31050,-31113,-31176,-31237,-31297
    ↪
-31526,-31580,-31633,-31685,-31736,-31785,-31833,-31880,-31926,-31971,-32014,-32057,-32098,-32137,-32176
    ↪
-32318,-32351,-32382,-32412,-32441,-32469,-32495,-32521,-32545,-32567,-32589,-32609,-32628,-32646,-32663
```

```
    ↪
-32717,-32728,-32737,-32745,-32752,-32757,-32761,-32765,-32766};

#elif (FFT_POINTS_NUMBER == 2048)
{0,101,201,302,402,503,603,704,804,905,1005,1106,1206,1307,1407,1507,1608,1708,1809,1909,2009,2110,2210,
    ↪
2611,2711,2811,2911,3012,3112,3212,3312,3412,3512,3612,3712,3811,3911,4011,4111,4210,4310,4410,4509,4609
    ↪
5007,5106,5205,5305,5404,5503,5602,5701,5800,5899,5998,6096,6195,6294,6393,6491,6590,6688,6786,6885,6983
    ↪
7375,7473,7571,7669,7767,7864,7962,8059,8157,8254,8351,8448,8545,8642,8739,8836,8933,9030,9126,9223,9319
    ↪
9704,9800,9896,9992,10087,10183,10278,10374,10469,10564,10659,10754,10849,10944,11039,11133,11228,11322,
    ↪
11699,11793,11886,11980,12074,12167,12260,12353,12446,12539,12632,12725,12817,12910,13002,13094,13187,13
    ↪
13554,13645,13736,13828,13919,14010,14101,14191,14282,14372,14462,14553,14643,14732,14822,14912,15001,15
    ↪
15358,15446,15535,15623,15712,15800,15888,15976,16063,16151,16238,16325,16413,16499,16586,16673,16759,16
    ↪
17104,17189,17275,17360,17445,17530,17615,17700,17784,17869,17953,18037,18121,18204,18288,18371,18454,18
    ↪
18785,18868,18950,19032,19113,19195,19276,19357,19438,19519,19600,19680,19761,19841,19921,20000,20080,20
    ↪
20396,20475,20553,20631,20709,20787,20865,20942,21019,21096,21173,21250,21326,21403,21479,21554,21630,21
    ↪
21930,22005,22079,22154,22227,22301,22375,22448,22521,22594,22667,22739,22812,22884,22956,23027,23099,23
    ↪
23382,23452,23522,23592,23662,23731,23801,23870,23938,24007,24075,24143,24211,24279,24346,24413,24480,24
    ↪
24746,24811,24877,24942,25007,25072,25137,25201,25265,25329,25393,25456,25519,25582,25645,25708,25770,25
    ↪
26016,26077,26138,26198,26259,26319,26378,26438,26497,26556,26615,26674,26732,26790,26848,26905,26962,27
    ↪
27189,27245,27300,27356,27411,27466,27521,27575,27629,27683,27737,27790,27843,27896,27949,28001,28053,28
    ↪
28259,28310,28360,28411,28460,28510,28560,28609,28658,28706,28755,28803,28850,28898,28945,28992,29039,29
    ↪
29223,29268,29313,29358,29403,29447,29491,29534,29578,29621,29664,29706,29749,29791,29832,29874,29915,29
    ↪
30077,30117,30156,30195,30234,30273,30311,30349,30387,30424,30462,30498,30535,30571,30607,30643,30679,30
    ↪
30818,30852,30885,30919,30952,30985,31017,31050,31082,31113,31145,31176,31206,31237,31267,31297,31327,31
    ↪
31442,31470,31498,31526,31553,31580,31607,31633,31659,31685,31710,31736,31760,31785,31809,31833,31857,31
    ↪
31949,31971,31993,32014,32036,32057,32077,32098,32118,32137,32157,32176,32195,32213,32232,32250,32267,32
    ↪
32335,32351,32367,32382,32397,32412,32427,32441,32455,32469,32482,32495,32508,32521,32533,32545,32556,32
    ↪
32599,32609,32619,32628,32637,32646,32655,32663,32671,32678,32685,32692,32699,32705,32711,32717,32722,32
    ↪
32741,32745,32748,32752,32755,32757,32759,32761,32763,32765,32766,32766,32767,32767,32767,32766,32766,32
    ↪
```

```
32759,32757,32755,32752,32748,32745,32741,32737,32732,32728,32722,32717,32711,32705,32699,32692,32685,32
    ↪
32655,32646,32637,32628,32619,32609,32599,32589,32578,32567,32556,32545,32533,32521,32508,32495,32482,32
    ↪
32427,32412,32397,32382,32367,32351,32335,32318,32302,32285,32267,32250,32232,32213,32195,32176,32157,32
    ↪
32077,32057,32036,32014,31993,31971,31949,31926,31903,31880,31857,31833,31809,31785,31760,31736,31710,31
    ↪
31607,31580,31553,31526,31498,31470,31442,31414,31385,31356,31327,31297,31267,31237,31206,31176,31145,31
    ↪
31017,30985,30952,30919,30885,30852,30818,30783,30749,30714,30679,30643,30607,30571,30535,30498,30462,30
    ↪
30311,30273,30234,30195,30156,30117,30077,30037,29997,29956,29915,29874,29832,29791,29749,29706,29664,29
    ↪
29491,29447,29403,29358,29313,29268,29223,29177,29131,29085,29039,28992,28945,28898,28850,28803,28755,28
    ↪
28560,28510,28460,28411,28360,28310,28259,28208,28157,28105,28053,28001,27949,27896,27843,27790,27737,27
    ↪
27521,27466,27411,27356,27300,27245,27189,27133,27076,27019,26962,26905,26848,26790,26732,26674,26615,26
    ↪
26378,26319,26259,26198,26138,26077,26016,25955,25893,25832,25770,25708,25645,25582,25519,25456,25393,25
    ↪
25137,25072,25007,24942,24877,24811,24746,24680,24613,24547,24480,24413,24346,24279,24211,24143,24075,24
    ↪
23801,23731,23662,23592,23522,23452,23382,23311,23241,23170,23099,23027,22956,22884,22812,22739,22667,22
    ↪
22375,22301,22227,22154,22079,22005,21930,21856,21781,21705,21630,21554,21479,21403,21326,21250,21173,21
    ↪
20865,20787,20709,20631,20553,20475,20396,20317,20238,20159,20080,20000,19921,19841,19761,19680,19600,19
    ↪
19276,19195,19113,19032,18950,18868,18785,18703,18620,18537,18454,18371,18288,18204,18121,18037,17953,17
    ↪
17615,17530,17445,17360,17275,17189,17104,17018,16932,16846,16759,16673,16586,16499,16413,16325,16238,16
    ↪
15888,15800,15712,15623,15535,15446,15358,15269,15180,15090,15001,14912,14822,14732,14643,14553,14462,14
    ↪
14101,14010,13919,13828,13736,13645,13554,13462,13370,13279,13187,13094,13002,12910,12817,12725,12632,12
    ↪
12260,12167,12074,11980,11886,11793,11699,11605,11511,11417,11322,11228,11133,11039,10944,10849,10754,10
    ↪
10374,10278,10183,10087,9992,9896,9800,9704,9608,9512,9416,9319,9223,9126,9030,8933,8836,8739,8642,8545,
    ↪
8157,8059,7962,7864,7767,7669,7571,7473,7375,7277,7179,7081,6983,6885,6786,6688,6590,6491,6393,6294,6195
    ↪
5800,5701,5602,5503,5404,5305,5205,5106,5007,4907,4808,4708,4609,4509,4410,4310,4210,4111,4011,3911,3811
    ↪
3412,3312,3212,3112,3012,2911,2811,2711,2611,2511,2410,2310,2210,2110,2009,1909,1809,1708,1608,1507,1407
    ↪
1005,905,804,704,603,503,402,302,201,101,0,-101,-201,-302,-402,-503,-603,-704,-804,-905,-1005,-1106,-120
    ↪
-1507,-1608,-1708,-1809,-1909,-2009,-2110,-2210,-2310,-2410,-2511,-2611,-2711,-2811,-2911,-3012,-3112,-3
    ↪
-3512,-3612,-3712,-3811,-3911,-4011,-4111,-4210,-4310,-4410,-4509,-4609,-4708,-4808,-4907,-5007,-5106,-5
    ↪
```

```
-5503,-5602,-5701,-5800,-5899,-5998,-6096,-6195,-6294,-6393,-6491,-6590,-6688,-6786,-6885,-6983,-7081,-7
    ↪
-7473,-7571,-7669,-7767,-7864,-7962,-8059,-8157,-8254,-8351,-8448,-8545,-8642,-8739,-8836,-8933,-9030,-9
    ↪
-9416,-9512,-9608,-9704,-9800,-9896,-9992,-10087,-10183,-10278,-10374,-10469,-10564,-10659,-10754,-1084
    ↪
-11133,-11228,-11322,-11417,-11511,-11605,-11699,-11793,-11886,-11980,-12074,-12167,-12260,-12353,-1244
    ↪
-12725,-12817,-12910,-13002,-13094,-13187,-13279,-13370,-13462,-13554,-13645,-13736,-13828,-13919,-1401
    ↪
-14282,-14372,-14462,-14553,-14643,-14732,-14822,-14912,-15001,-15090,-15180,-15269,-15358,-15446,-1553
    ↪
-15800,-15888,-15976,-16063,-16151,-16238,-16325,-16413,-16499,-16586,-16673,-16759,-16846,-16932,-1701
    ↪
-17275,-17360,-17445,-17530,-17615,-17700,-17784,-17869,-17953,-18037,-18121,-18204,-18288,-18371,-1845
    ↪
-18703,-18785,-18868,-18950,-19032,-19113,-19195,-19276,-19357,-19438,-19519,-19600,-19680,-19761,-1984
    ↪
-20080,-20159,-20238,-20317,-20396,-20475,-20553,-20631,-20709,-20787,-20865,-20942,-21019,-21096,-2117
    ↪
-21403,-21479,-21554,-21630,-21705,-21781,-21856,-21930,-22005,-22079,-22154,-22227,-22301,-22375,-2244
    ↪
-22667,-22739,-22812,-22884,-22956,-23027,-23099,-23170,-23241,-23311,-23382,-23452,-23522,-23592,-2366
    ↪
-23870,-23938,-24007,-24075,-24143,-24211,-24279,-24346,-24413,-24480,-24547,-24613,-24680,-24746,-2481
    ↪
-25007,-25072,-25137,-25201,-25265,-25329,-25393,-25456,-25519,-25582,-25645,-25708,-25770,-25832,-2589
    ↪
-26077,-26138,-26198,-26259,-26319,-26378,-26438,-26497,-26556,-26615,-26674,-26732,-26790,-26848,-2690
    ↪
-27076,-27133,-27189,-27245,-27300,-27356,-27411,-27466,-27521,-27575,-27629,-27683,-27737,-27790,-2784
    ↪
-28001,-28053,-28105,-28157,-28208,-28259,-28310,-28360,-28411,-28460,-28510,-28560,-28609,-28658,-2870
    ↪
-28850,-28898,-28945,-28992,-29039,-29085,-29131,-29177,-29223,-29268,-29313,-29358,-29403,-29447,-2949
    ↪
-29621,-29664,-29706,-29749,-29791,-29832,-29874,-29915,-29956,-29997,-30037,-30077,-30117,-30156,-3019
    ↪
-30311,-30349,-30387,-30424,-30462,-30498,-30535,-30571,-30607,-30643,-30679,-30714,-30749,-30783,-3081
    ↪
-30919,-30952,-30985,-31017,-31050,-31082,-31113,-31145,-31176,-31206,-31237,-31267,-31297,-31327,-3135
    ↪
-31442,-31470,-31498,-31526,-31553,-31580,-31607,-31633,-31659,-31685,-31710,-31736,-31760,-31785,-3180
    ↪
-31880,-31903,-31926,-31949,-31971,-31993,-32014,-32036,-32057,-32077,-32098,-32118,-32137,-32157,-3217
    ↪
-32232,-32250,-32267,-32285,-32302,-32318,-32335,-32351,-32367,-32382,-32397,-32412,-32427,-32441,-3245
    ↪
-32495,-32508,-32521,-32533,-32545,-32556,-32567,-32578,-32589,-32599,-32609,-32619,-32628,-32637,-3264
    ↪
-32671,-32678,-32685,-32692,-32699,-32705,-32711,-32717,-32722,-32728,-32732,-32737,-32741,-32745,-3274
    ↪
-32757,-32759,-32761,-32763,-32765,-32766,-32766,-32767};
#else
```

```
{};
#error "The␣selected␣points␣number␣FFT_POINTS_NUMBER␣in␣fft.h␣is␣not␣supported."
#endif

void FFT(int16_t* bufferRe, int16_t* bufferIm)
{

    int16_t bl; // current butterfly size
    int16_t l; // half current butterfly size
    int16_t k; // current level number
    int16_t m; // coefficient counter
    int16_t i; // first point offset
    int16_t j; // second point offset
    int16_t wRe; // current twiddle coefficient (real)
    int16_t wIm; // current twiddle coefficient (imaginary)
    int16_t temp;


    bl = FFT_POINTS_NUMBER; // current butterfly size
    l = FFT_POINTS_NUMBER>>1; // current butterfly size divided by 2
        k = 0; // level number

        while (l > 0)
        {
               for (m=0; m<l; m++)
               {
                      j = m << k; // coefficient offset

                      wRe = twiddleCoefficients[j+FFT_POINTS_NUMBER/4];
                      wIm = -twiddleCoefficients[j];

                      for (i=m; i<FFT_POINTS_NUMBER; i+=bl) // cycle for all butterflies
                      {
                             j = i + l;
               // here:
               // i - offset for the first point
               // j - offset for the second point
               FFT2(bufferRe+i, bufferIm+i, bufferRe+j, bufferIm+j, wRe, wIm);
                      }
               }

               k++;
               l >>= 1;
        bl >>= 1;
        }

    for(i=1; i<(FFT_POINTS_NUMBER-1); i++)
    {
        j = bitPermutationTable[i];

        if(j <= i)
        {
            continue;
        }
```

```
        temp = bufferRe[i];
        bufferRe[i] = bufferRe[j];
        bufferRe[j] = temp;

        temp = bufferIm[i];
        bufferIm[i] = bufferIm[j];
        bufferIm[j] = temp;
    }
}

/*****************************************************************************
 A = SQRT(Re^2 + Im^2)
 *****************************************************************************/
uint16_t FFTAmplitude(int16_t re, int16_t im)
{
uint32_t _bit = ((uint32_t)1) << 30;
uint32_t num;
uint32_t res = 0;

    num = ((uint32_t)re)*re + ((uint32_t)im)*im;

    // 32 bit integer square root calculation.
    while (_bit > num)
        _bit >>= 2;

    while (_bit != 0) {
        if (num >= res + _bit) {
            num -= res + _bit;
            res = (res >> 1) + _bit;
        }
        else
            res >>= 1;
        _bit >>= 2;
    }
    return ((uint16_t)res);
}

/*****************************************************************************
 D = (180/Pi)*ARCTAN(Im/Re)

 Second order:
  atan_approx(x) = (Pi/2)*(b*x + x*x)/(1 + 2*b*x + x*x)
  where b = 0.596227, with a maximum approximation error of 0.1620

 Third order:
  atan_approx(x) = (Pi/2)*(c*x + x*x + x*x*x)/(1 + (c+1)*x + (c+1)x*x + x*x*x)
  where c = (1 + sqrt(17))/8 and the maximum approximation error is 0.00811

 *****************************************************************************/
int16_t FFTPhase(int16_t re, int16_t im)
{
typedef union
{
```

```
    uint32_t w;
    struct
    {
        uint8_t lo;
        uint8_t hi;
        uint8_t up;
        uint8_t reserved;
    };
} DATA;

// constants in 16.16 bit fixed point format
#define POINT_POS 8
#define ONE (((uint32_t)1)<<POINT_POS)
#define HALF ((uint32_t)(ONE*0.5))
#define B ((uint32_t)(ONE*0.596227))

int32_t bx;
int32_t xx;
int32_t result;
int32_t x;
uint16_t sign = 0;
uint16_t quadrant = 0;

   if(im == 0)
   {
       return 0;
   }

   // im/re
   if(re == 0)
   {
       if(im < 0)
       {
           return ((int16_t)-90);
       }else{
           return ((int16_t)90);
       }
   }
   x = im;
   x <<= POINT_POS;
   x /= re;

   // arctan(-x) = -arctan(x)
   if(x < 0)
   {
       sign = 1;
       x = -x;
   }

   if(re < 0)
   {
       quadrant = 1;
   }
```

```
    // b*x
    bx = B;
    bx *= x;
    // shift after multiplication
    bx >>= POINT_POS;

    // x^2
    xx = x;
    xx *= x;
    // shift after multiplication
    xx >>= POINT_POS;

    // bx + x^2
    result = bx;
    result += xx;


    // (bx + x^2)/(1+2bx+x^2)
    // shift before division
    result <<= POINT_POS;
    result /= (ONE + (bx<<1) + xx);

    result *= 90;

    if(result&HALF)
    {
        result += ONE;
    }

    result >>= POINT_POS;

    if(quadrant)
    {
        // add 90 degree
        result += 90;
    }

    if(sign)
    {
        return ((int16_t)-result);
    }

    return ((int16_t)result);
}
```

C.1.3 fft_.s

```
/*
 * File: fft_.s
 * Author: Kevin Schneier
 *
 */

.global _FFT2

.section .text

_FFT2:
    push.d w8
    push.d w10

    ; w4 -> coeff RE
    ; w5 -> coeff IM
    ; w0 -> w8 - first RE address
    ; w1 -> w9 - first IM address
    ; w2 -> w10 - second RE address
    ; w3 -> w11 - second IM address
    mov w0, w8
    mov w1, w9
    mov w2, w10
    mov w3, w11

    ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
    ; PLUS PATH
    ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

    ; w0 - first RE
    ; w1 - first IM
    ; w2 - second RE
    ; w3 - second IM
    mov [w8], w0
    mov [w9], w1
    mov [w10], w2
    mov [w11], w3

    asr w0, #1, w0
    asr w1, #1, w1
    asr w2, #1, w2
    asr w3, #1, w3

    add w0, w2, w6 ; real

    ; SAVE W6 RESULT (REAL)
    mov w6, [w8]

    add w1, w3, w7 ; imaginary

    ; SAVE W7 RESULT (IMAGINARY)
    mov w7, [w9]
```

```
    ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
    ; MINUS PATH
    ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

    sub w0, w2, w0 ; real
    sub w1, w3, w1 ; imaginary

    ; multiply by coefficient
    mul.ss w0, w4, w2 ; re by re
    mul.ss w1, w5, w6 ; im by im
    sub w2, w6, w2
    subb w3, w7, w3

    ; SAVE RESULT W3 HERE (REAL)
    sl w3, #1, w3
    btsc w2, #15
    bset w3, #0
    mov w3, [w10]

    ; multiply by coefficient
    mul.ss w0, w5, w2 ; re by im
    mul.ss w1, w4, w6 ; im by re
    add w2, w6, w2
    addc w3, w7, w3

    ; SAVE RESULT W3 HERE (IMAGINARY)
    sl w3, #1, w3
    btsc w2, #15
    bset w3, #0
    mov w3, [w11]

    pop.d w10
    pop.d w8


    return

.end
```

C.1.4 fundamental.c

```
/*
 * File: fundamental.c
 * Author: Kevin Schneier
 *
 */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include "fundamental.h"

double extract(uint16_t *fft, int Fs, int n) {
       double f_peaks[FPLENGTH];
       int i, j, k, m;
       for (j = 0; j < 5; j++) {
              //f_peaks[j] = strings[stringNum-1][j];
       }
       int length_f_peaks = FPLENGTH;
       int guess[FPLENGTH][5*FPLENGTH] = {{0}};
       int guessNum[FPLENGTH] = {0};
       double int_ratio_threshold = .05;
       int modeNums[50];
       int modeMax;
       int mode;
       double fund_sum = 0;
       double fundamental;
// double ratio;
// double lowerBound;
// double upperBound;
    double freq_x[256]; //Frequency range
    int peakTempLength = 0; //Length of temporary peak array
    double fPeaksTemp[256] = {0}; //Temporary peak array
    double ampCutoff = 20; //Power level above which FFT value is considered a peak
    double sameFreqs[20] = {0};
    double mean;
    double num;


    //Create array of frequency values whose values correspond to those of created FFT
        ↪ power array
    //CHECK: how to compute power for our FFT
    for (j = 0; j < n; j++) {
        freq_x[j] = j*Fs/n;
    }

    //Find frequency values at which FFT power exceeds threshold
        for (j = 0; j <= n; j++) {
               if (fft[j] > ampCutoff) { // && freq_x[j] < Fs/2 to eliminate higher
                   ↪ frequencies
            fPeaksTemp[peakTempLength] = freq_x[j];
                      peakTempLength++;
```

```
            }
    }

    //Find array indices of frequencies that are close together (the 'same' peak)
    k = 0;
    for (j = 0; j < peakTempLength - 1; j++) {
        if (fPeaksTemp[j+1] - fPeaksTemp[j] < 2)
            sameFreqs[k] = sameFreqs[k] + 1;
        else {
            k++;
            sameFreqs[k] = sameFreqs[k-1] + 1;
        }
    }
    length_f_peaks = k;
    //printf("length_f_peaks = %d\n", length_f_peaks);

    //Average out frequencies that are close together
    for (j = 0; j <= k; j++) {
        mean = 0;
        num = 0;
        if (j == 0) {
            for (i = 0; i <= sameFreqs[0]; i++) {
                mean = mean + fPeaksTemp[i];
                num++;
            }
        }
        else {
            for (i = sameFreqs[j-1]+1; i <= sameFreqs[j]; i++) {
                mean = mean + fPeaksTemp[i];
                num++;
            }
        }
        f_peaks[j] = mean/num;
        //printf("Peak %d: %f Hz\n", j, f_peaks[j]);
    }

    /*
        if ((stringNum == 3) || (stringNum == 6)) {
                length_f_peaks = 4;
        }
        if (stringNum == 5) {
                length_f_peaks = 3;
        }
     */

    //Store harmonic number guesses when peak ratio is an integer ratio
        for (j = length_f_peaks-1; j >= 0; j--) {
                for (k = j-1; k >= 0; k--) {
                        for (m = (length_f_peaks+1); m > 0; m--) {
                                for (n = m-1; n > 0; n--) {
                                        if ((f_peaks[j]/f_peaks[k] >= (double)m/(double)n -
                                            ↪ int_ratio_threshold)       && (f_peaks[j]/
                                            ↪ f_peaks[k] <= (double)m/(double)n +
                                            ↪ int_ratio_threshold)) {
```

```
                                        guess[j][guessNum[j]] = m;
                                        guess[k][guessNum[k]] = n;
                                        guessNum[j]++;
                        guessNum[k]++;
                                        //printf("Guess for peak %d: %d\n", j, m);
                                        //printf("Guess for peak %d: %d\n", k, n);
                                        //ratio = f_peaks[j]/f_peaks[k];
                                        //printf("Ratio: %f\n", ratio);
                                        //lowerBound = (double)m/(double)n - .05;
                                        //printf("Lower Bound: %f\n", lowerBound);
                                        //upperBound = (double)m/(double)n + .05;
                                        //printf("Upper Bound: %f\n", upperBound);
                                    }
                                }
                        }
                }
        }

    //Find the most popular harmonic number guess for each peak
    //Make guesses for fundamental value based on each peak divided by its harmonic
        ↪ number
    //Average all fundamental guesses
    num = 0;
        for (j = 0; j <= length_f_peaks; j++) {
                //printf("Peak %d:\n", j);
                memset(modeNums, 0, sizeof(modeNums)); //Set harmonic number counter bins
                    ↪ to zero
                modeMax = 0;
                mode = 0;
                for (k = 0; k < guessNum[j]; k++) {
                        modeNums[guess[j][k]]++; //Increment the proper bin for each
                            ↪ harmonic guess
                        if (modeNums[guess[j][k]] > modeMax) { //If the newly incremented
                            ↪ bin is highest count value
                            modeMax = modeNums[guess[j][k]]; //Set max count value to
                                ↪ this
                            mode = guess[j][k]; //Set new mode harmonic guess value
                            //printf("Mode Max: %d\n", modeMax);
                            //printf("Mode: %d\n", mode);
                        }
                }
                //printf("%d\n", mode);
        if (guessNum[j] != 0) {
            fund_sum = fund_sum + f_peaks[j]/mode;
            num++;
        }
        }

        fundamental = fund_sum/(double)num;



        return fundamental;
}
```

```
int indexExtract(uint16_t *fft, int numPoints) {
    int i, j;
    uint16_t value = 0;
    for (i=0; i<numPoints; i++){
        if (fft[i] > value) {
            j = i;
            value = fft[i];
        }
    }
    if (value < 40){
        return 0;
    }
    return j;
}

//int *extractAll(uint16_t *fft, int numPoints) {
// /* This function finds the 6 highest peaks in
// the FFT of our strum; it then returns
// them in a sorted list. This function
// returns the INDICES of the 6 highest peaks. */
//}
```

C.1.5 glcdfont.c

```
/*
 * File: glcdfont.c
 * Author: Kevin Schneier
 *
 */

#ifndef FONT5X7_H
#define FONT5X7_H

// Standard ASCII 5x7 font

static const unsigned char font[] = {
       0x00, 0x00, 0x00, 0x00, 0x00,
       0x3E, 0x5B, 0x4F, 0x5B, 0x3E,
       0x3E, 0x6B, 0x4F, 0x6B, 0x3E,
       0x1C, 0x3E, 0x7C, 0x3E, 0x1C,
       0x18, 0x3C, 0x7E, 0x3C, 0x18,
       0x1C, 0x57, 0x7D, 0x57, 0x1C,
       0x1C, 0x5E, 0x7F, 0x5E, 0x1C,
       0x00, 0x18, 0x3C, 0x18, 0x00,
       0xFF, 0xE7, 0xC3, 0xE7, 0xFF,
       0x00, 0x18, 0x24, 0x18, 0x00,
       0xFF, 0xE7, 0xDB, 0xE7, 0xFF,
       0x30, 0x48, 0x3A, 0x06, 0x0E,
       0x26, 0x29, 0x79, 0x29, 0x26,
       0x40, 0x7F, 0x05, 0x05, 0x07,
       0x40, 0x7F, 0x05, 0x25, 0x3F,
       0x5A, 0x3C, 0xE7, 0x3C, 0x5A,
       0x7F, 0x3E, 0x1C, 0x1C, 0x08,
       0x08, 0x1C, 0x1C, 0x3E, 0x7F,
       0x14, 0x22, 0x7F, 0x22, 0x14,
       0x5F, 0x5F, 0x00, 0x5F, 0x5F,
       0x06, 0x09, 0x7F, 0x01, 0x7F,
       0x00, 0x66, 0x89, 0x95, 0x6A,
       0x60, 0x60, 0x60, 0x60, 0x60,
       0x94, 0xA2, 0xFF, 0xA2, 0x94,
       0x08, 0x04, 0x7E, 0x04, 0x08,
       0x10, 0x20, 0x7E, 0x20, 0x10,
       0x08, 0x08, 0x2A, 0x1C, 0x08,
       0x08, 0x1C, 0x2A, 0x08, 0x08,
       0x1E, 0x10, 0x10, 0x10, 0x10,
       0x0C, 0x1E, 0x0C, 0x1E, 0x0C,
       0x30, 0x38, 0x3E, 0x38, 0x30,
       0x06, 0x0E, 0x3E, 0x0E, 0x06,
       0x00, 0x00, 0x00, 0x00, 0x00,
       0x00, 0x00, 0x5F, 0x00, 0x00,
       0x00, 0x07, 0x00, 0x07, 0x00,
       0x14, 0x7F, 0x14, 0x7F, 0x14,
       0x24, 0x2A, 0x7F, 0x2A, 0x12,
       0x23, 0x13, 0x08, 0x64, 0x62,
       0x36, 0x49, 0x56, 0x20, 0x50,
       0x00, 0x08, 0x07, 0x03, 0x00,
```

```
0x00, 0x1C, 0x22, 0x41, 0x00,
0x00, 0x41, 0x22, 0x1C, 0x00,
0x2A, 0x1C, 0x7F, 0x1C, 0x2A,
0x08, 0x08, 0x3E, 0x08, 0x08,
0x00, 0x80, 0x70, 0x30, 0x00,
0x08, 0x08, 0x08, 0x08, 0x08,
0x00, 0x00, 0x60, 0x60, 0x00,
0x20, 0x10, 0x08, 0x04, 0x02,
0x3E, 0x51, 0x49, 0x45, 0x3E,
0x00, 0x42, 0x7F, 0x40, 0x00,
0x72, 0x49, 0x49, 0x49, 0x46,
0x21, 0x41, 0x49, 0x4D, 0x33,
0x18, 0x14, 0x12, 0x7F, 0x10,
0x27, 0x45, 0x45, 0x45, 0x39,
0x3C, 0x4A, 0x49, 0x49, 0x31,
0x41, 0x21, 0x11, 0x09, 0x07,
0x36, 0x49, 0x49, 0x49, 0x36,
0x46, 0x49, 0x49, 0x29, 0x1E,
0x00, 0x00, 0x14, 0x00, 0x00,
0x00, 0x40, 0x34, 0x00, 0x00,
0x00, 0x08, 0x14, 0x22, 0x41,
0x14, 0x14, 0x14, 0x14, 0x14,
0x00, 0x41, 0x22, 0x14, 0x08,
0x02, 0x01, 0x59, 0x09, 0x06,
0x3E, 0x41, 0x5D, 0x59, 0x4E,
0x7C, 0x12, 0x11, 0x12, 0x7C,
0x7F, 0x49, 0x49, 0x49, 0x36,
0x3E, 0x41, 0x41, 0x41, 0x22,
0x7F, 0x41, 0x41, 0x41, 0x3E,
0x7F, 0x49, 0x49, 0x49, 0x41,
0x7F, 0x09, 0x09, 0x09, 0x01,
0x3E, 0x41, 0x41, 0x51, 0x73,
0x7F, 0x08, 0x08, 0x08, 0x7F,
0x00, 0x41, 0x7F, 0x41, 0x00,
0x20, 0x40, 0x41, 0x3F, 0x01,
0x7F, 0x08, 0x14, 0x22, 0x41,
0x7F, 0x40, 0x40, 0x40, 0x40,
0x7F, 0x02, 0x1C, 0x02, 0x7F,
0x7F, 0x04, 0x08, 0x10, 0x7F,
0x3E, 0x41, 0x41, 0x41, 0x3E,
0x7F, 0x09, 0x09, 0x09, 0x06,
0x3E, 0x41, 0x51, 0x21, 0x5E,
0x7F, 0x09, 0x19, 0x29, 0x46,
0x26, 0x49, 0x49, 0x49, 0x32,
0x03, 0x01, 0x7F, 0x01, 0x03,
0x3F, 0x40, 0x40, 0x40, 0x3F,
0x1F, 0x20, 0x40, 0x20, 0x1F,
0x3F, 0x40, 0x38, 0x40, 0x3F,
0x63, 0x14, 0x08, 0x14, 0x63,
0x03, 0x04, 0x78, 0x04, 0x03,
0x61, 0x59, 0x49, 0x4D, 0x43,
0x00, 0x7F, 0x41, 0x41, 0x41,
0x02, 0x04, 0x08, 0x10, 0x20,
0x00, 0x41, 0x41, 0x41, 0x7F,
```

```
0x04, 0x02, 0x01, 0x02, 0x04,
0x40, 0x40, 0x40, 0x40, 0x40,
0x00, 0x03, 0x07, 0x08, 0x00,
0x20, 0x54, 0x54, 0x78, 0x40,
0x7F, 0x28, 0x44, 0x44, 0x38,
0x38, 0x44, 0x44, 0x44, 0x28,
0x38, 0x44, 0x44, 0x28, 0x7F,
0x38, 0x54, 0x54, 0x54, 0x18,
0x00, 0x08, 0x7E, 0x09, 0x02,
0x18, 0xA4, 0xA4, 0x9C, 0x78,
0x7F, 0x08, 0x04, 0x04, 0x78,
0x00, 0x44, 0x7D, 0x40, 0x00,
0x20, 0x40, 0x40, 0x3D, 0x00,
0x7F, 0x10, 0x28, 0x44, 0x00,
0x00, 0x41, 0x7F, 0x40, 0x00,
0x7C, 0x04, 0x78, 0x04, 0x78,
0x7C, 0x08, 0x04, 0x04, 0x78,
0x38, 0x44, 0x44, 0x44, 0x38,
0xFC, 0x18, 0x24, 0x24, 0x18,
0x18, 0x24, 0x24, 0x18, 0xFC,
0x7C, 0x08, 0x04, 0x04, 0x08,
0x48, 0x54, 0x54, 0x54, 0x24,
0x04, 0x04, 0x3F, 0x44, 0x24,
0x3C, 0x40, 0x40, 0x20, 0x7C,
0x1C, 0x20, 0x40, 0x20, 0x1C,
0x3C, 0x40, 0x30, 0x40, 0x3C,
0x44, 0x28, 0x10, 0x28, 0x44,
0x4C, 0x90, 0x90, 0x90, 0x7C,
0x44, 0x64, 0x54, 0x4C, 0x44,
0x00, 0x08, 0x36, 0x41, 0x00,
0x00, 0x00, 0x77, 0x00, 0x00,
0x00, 0x41, 0x36, 0x08, 0x00,
0x02, 0x01, 0x02, 0x04, 0x02,
0x3C, 0x26, 0x23, 0x26, 0x3C,
0x1E, 0xA1, 0xA1, 0x61, 0x12,
0x3A, 0x40, 0x40, 0x20, 0x7A,
0x38, 0x54, 0x54, 0x55, 0x59,
0x21, 0x55, 0x55, 0x79, 0x41,
0x21, 0x54, 0x54, 0x78, 0x41,
0x21, 0x55, 0x54, 0x78, 0x40,
0x20, 0x54, 0x55, 0x79, 0x40,
0x0C, 0x1E, 0x52, 0x72, 0x12,
0x39, 0x55, 0x55, 0x55, 0x59,
0x39, 0x54, 0x54, 0x54, 0x59,
0x39, 0x55, 0x54, 0x54, 0x58,
0x00, 0x00, 0x45, 0x7C, 0x41,
0x00, 0x02, 0x45, 0x7D, 0x42,
0x00, 0x01, 0x45, 0x7C, 0x40,
0xF0, 0x29, 0x24, 0x29, 0xF0,
0xF0, 0x28, 0x25, 0x28, 0xF0,
0x7C, 0x54, 0x55, 0x45, 0x00,
0x20, 0x54, 0x54, 0x7C, 0x54,
0x7C, 0x0A, 0x09, 0x7F, 0x49,
0x32, 0x49, 0x49, 0x49, 0x32,
```

```
0x32, 0x48, 0x48, 0x48, 0x32,
0x32, 0x4A, 0x48, 0x48, 0x30,
0x3A, 0x41, 0x41, 0x21, 0x7A,
0x3A, 0x42, 0x40, 0x20, 0x78,
0x00, 0x9D, 0xA0, 0xA0, 0x7D,
0x39, 0x44, 0x44, 0x44, 0x39,
0x3D, 0x40, 0x40, 0x40, 0x3D,
0x3C, 0x24, 0xFF, 0x24, 0x24,
0x48, 0x7E, 0x49, 0x43, 0x66,
0x2B, 0x2F, 0xFC, 0x2F, 0x2B,
0xFF, 0x09, 0x29, 0xF6, 0x20,
0xC0, 0x88, 0x7E, 0x09, 0x03,
0x20, 0x54, 0x54, 0x79, 0x41,
0x00, 0x00, 0x44, 0x7D, 0x41,
0x30, 0x48, 0x48, 0x4A, 0x32,
0x38, 0x40, 0x40, 0x22, 0x7A,
0x00, 0x7A, 0x0A, 0x0A, 0x72,
0x7D, 0x0D, 0x19, 0x31, 0x7D,
0x26, 0x29, 0x29, 0x2F, 0x28,
0x26, 0x29, 0x29, 0x29, 0x26,
0x30, 0x48, 0x4D, 0x40, 0x20,
0x38, 0x08, 0x08, 0x08, 0x08,
0x08, 0x08, 0x08, 0x08, 0x38,
0x2F, 0x10, 0xC8, 0xAC, 0xBA,
0x2F, 0x10, 0x28, 0x34, 0xFA,
0x00, 0x00, 0x7B, 0x00, 0x00,
0x08, 0x14, 0x2A, 0x14, 0x22,
0x22, 0x14, 0x2A, 0x14, 0x08,
0xAA, 0x00, 0x55, 0x00, 0xAA,
0xAA, 0x55, 0xAA, 0x55, 0xAA,
0x00, 0x00, 0x00, 0xFF, 0x00,
0x10, 0x10, 0x10, 0xFF, 0x00,
0x14, 0x14, 0x14, 0xFF, 0x00,
0x10, 0x10, 0xFF, 0x00, 0xFF,
0x10, 0x10, 0xF0, 0x10, 0xF0,
0x14, 0x14, 0x14, 0xFC, 0x00,
0x14, 0x14, 0xF7, 0x00, 0xFF,
0x00, 0x00, 0xFF, 0x00, 0xFF,
0x14, 0x14, 0xF4, 0x04, 0xFC,
0x14, 0x14, 0x17, 0x10, 0x1F,
0x10, 0x10, 0x1F, 0x10, 0x1F,
0x14, 0x14, 0x14, 0x1F, 0x00,
0x10, 0x10, 0x10, 0xF0, 0x00,
0x00, 0x00, 0x00, 0x1F, 0x10,
0x10, 0x10, 0x10, 0x1F, 0x10,
0x10, 0x10, 0x10, 0xF0, 0x10,
0x00, 0x00, 0x00, 0xFF, 0x10,
0x10, 0x10, 0x10, 0x10, 0x10,
0x10, 0x10, 0x10, 0xFF, 0x10,
0x00, 0x00, 0x00, 0xFF, 0x14,
0x00, 0x00, 0xFF, 0x00, 0xFF,
0x00, 0x00, 0x1F, 0x10, 0x17,
0x00, 0x00, 0xFC, 0x04, 0xF4,
0x14, 0x14, 0x17, 0x10, 0x17,
```

```
        0x14, 0x14, 0xF4, 0x04, 0xF4,
        0x00, 0x00, 0xFF, 0x00, 0xF7,
        0x14, 0x14, 0x14, 0x14, 0x14,
        0x14, 0x14, 0xF7, 0x00, 0xF7,
        0x14, 0x14, 0x14, 0x17, 0x14,
        0x10, 0x10, 0x1F, 0x10, 0x1F,
        0x14, 0x14, 0x14, 0xF4, 0x14,
        0x10, 0x10, 0xF0, 0x10, 0xF0,
        0x00, 0x00, 0x1F, 0x10, 0x1F,
        0x00, 0x00, 0x00, 0x1F, 0x14,
        0x00, 0x00, 0x00, 0xFC, 0x14,
        0x00, 0x00, 0xF0, 0x10, 0xF0,
        0x10, 0x10, 0xFF, 0x10, 0xFF,
        0x14, 0x14, 0x14, 0xFF, 0x14,
        0x10, 0x10, 0x10, 0x1F, 0x00,
        0x00, 0x00, 0x00, 0xF0, 0x10,
        0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
        0xF0, 0xF0, 0xF0, 0xF0, 0xF0,
        0xFF, 0xFF, 0xFF, 0x00, 0x00,
        0x00, 0x00, 0x00, 0xFF, 0xFF,
        0x0F, 0x0F, 0x0F, 0x0F, 0x0F,
        0x38, 0x44, 0x44, 0x38, 0x44,
        0x7C, 0x2A, 0x2A, 0x3E, 0x14,
        0x7E, 0x02, 0x02, 0x06, 0x06,
        0x02, 0x7E, 0x02, 0x7E, 0x02,
        0x63, 0x55, 0x49, 0x41, 0x63,
        0x38, 0x44, 0x44, 0x3C, 0x04,
        0x40, 0x7E, 0x20, 0x1E, 0x20,
        0x06, 0x02, 0x7E, 0x02, 0x02,
        0x99, 0xA5, 0xE7, 0xA5, 0x99,
        0x1C, 0x2A, 0x49, 0x2A, 0x1C,
        0x4C, 0x72, 0x01, 0x72, 0x4C,
        0x30, 0x4A, 0x4D, 0x4D, 0x30,
        0x30, 0x48, 0x78, 0x48, 0x30,
        0xBC, 0x62, 0x5A, 0x46, 0x3D,
        0x3E, 0x49, 0x49, 0x49, 0x00,
        0x7E, 0x01, 0x01, 0x01, 0x7E,
        0x2A, 0x2A, 0x2A, 0x2A, 0x2A,
        0x44, 0x44, 0x5F, 0x44, 0x44,
        0x40, 0x51, 0x4A, 0x44, 0x40,
        0x40, 0x44, 0x4A, 0x51, 0x40,
        0x00, 0x00, 0xFF, 0x01, 0x03,
        0xE0, 0x80, 0xFF, 0x00, 0x00,
        0x08, 0x08, 0x6B, 0x6B, 0x08,
        0x36, 0x12, 0x36, 0x24, 0x36,
        0x06, 0x0F, 0x09, 0x0F, 0x06,
        0x00, 0x00, 0x18, 0x18, 0x00,
        0x00, 0x00, 0x10, 0x10, 0x00,
        0x30, 0x40, 0xFF, 0x01, 0x01,
        0x00, 0x1F, 0x01, 0x01, 0x1E,
        0x00, 0x19, 0x1D, 0x17, 0x12,
        0x00, 0x3C, 0x3C, 0x3C, 0x3C,
        0x00, 0x00, 0x00, 0x00, 0x00
};
```

```
#endif // FONT5X7_H
```

C.1.6 touchCapability.c

```
#include <xc.h>
#include "touchCapability.h"

#define pressureThreshhold 10
#define rxplate 285 // resistance between X+ and X-

int x,y,z;
int mean=-99;

#define NUMSAMPLES 2

int * getPoint(void) {
  int samples[NUMSAMPLES];
  uint8_t i, valid;

  valid = 1;

  ADPCFGbits.PCFG3 = 0; // enables AN3
  TRISBbits.TRISB3 = 1; // sets AN3 to input
  TRISDbits.TRISD8 = 1; // sets RD8 to input
  TRISDbits.TRISD9 = 0; // sets RD9 to output
  ADPCFGbits.PCFG2 = 1; // disables AN2
  TRISBbits.TRISB2 = 0; // sets RB2 to output

  LATDbits.LATD9 = 1; // sets RD9 to high
  LATBbits.LATB2 = 0; // sets RB2 to low

  // X+ high, X- low, read Y+
   for (i=0; i<NUMSAMPLES; i++) {
       samples[i]=-99;
        while(touchReady==0);
        samples[i] = ADC_AN3_val;
        touchReady = 0;
   }

   // Allow small amount of measurement noise, because capacitive
   // coupling to a TFT display's signals can induce some noise.
   if (samples[0] - samples[1] < -20 || samples[0] - samples[1] > 20) {
     valid = 0;
   } else {
     samples[1] = (samples[0] + samples[1]) >> 1; // average 2 samples
   }

   x = (1023-samples[NUMSAMPLES/2]);

   TRISDbits.TRISD9 = 1; // sets RD9 to input
   ADPCFGbits.PCFG2 = 0; // enables AN2
   TRISBbits.TRISB2 = 1; // sets An2 to input
   ADPCFGbits.PCFG3 = 1; // disables AN3
   TRISBbits.TRISB3 = 0; // sets RB3 to output
   TRISDbits.TRISD8 = 0; // sets RD8 to output
```

```
LATDbits.LATD8 = 0;
LATBbits.LATB3 = 1;

// Y+ high, Y- low, read X-
for (i=0; i<NUMSAMPLES; i++) {
   samples[i]=-99;
   while(touchReady==0);
   samples[i] = ADC_AN2_val;
   touchReady=0;
}

// Allow small amount of measurement noise, because capacitive
// coupling to a TFT display's signals can induce some noise.
if (samples[0] - samples[1] < -20 || samples[0] - samples[1] > 20) {
  valid = 0;
} else {
  samples[1] = (samples[0] + samples[1]) >> 1; // average 2 samples
}

y = (1023-samples[NUMSAMPLES/2]);

// Set X+ to ground
// Set Y- to VCC
// Hi-Z X- and Y+
TRISDbits.TRISD9 = 0; // sets RD9 to output
ADPCFGbits.PCFG3 = 0; // enables AN3
TRISBbits.TRISB3 = 1; // sets AN3 to input

LATDbits.LATD9 = 0;
LATDbits.LATD8 = 1;

// Y- high, X+ low, read Y+ and X-
int z1, z2;
z1=-99;
while(touchReady==0);
z1 = ADC_AN2_val;
z2 = ADC_AN3_val;
touchReady = 0;

if (rxplate != 0) {
  float rtouch;
  rtouch = z2;
  rtouch /= z1;
  rtouch -= 1;
  rtouch *= x;
  rtouch *= rxplate;
  rtouch /= 1024;

  z = rtouch;
} else {
  z = (1023-(z2-z1));
}
```

```
   if (! valid) {
     z = 0;
   }

   coords[0] = x;
   coords[1] = y;
   coords[2] = z;

   return coords;
}
```

C.1.7 touchscreen.c

```
/*
 * File: touchscreen.c
 * Author: Kevin Schneier
 *
 */

#include <stdlib.h>
#include "touchscreen.h"
#include "glcdfont.c"


void tft_drawCircle(short x0, short y0, short r, unsigned short color) {
/* Draw a circle outline with center (x0,y0) and radius r, with given color
 * Parameters:
 * x0: x-coordinate of center of circle. The top-left of the screen
 * has x-coordinate 0 and increases to the right
 * y0: y-coordinate of center of circle. The top-left of the screen
 * has y-coordinate 0 and increases to the bottom
 * r: radius of circle
 * color: 16-bit color value for the circle. Note that the circle
 * isn't filled. So, this is the color of the outline of the circle
 * Returns: Nothing
 */
  short f = 1 - r;
  short ddF_x = 1;
  short ddF_y = -2 * r;
  short x = 0;
  short y = r;

  tft_drawPixel(x0 , y0+r, color);
  tft_drawPixel(x0 , y0-r, color);
  tft_drawPixel(x0+r, y0 , color);
  tft_drawPixel(x0-r, y0 , color);

  while (x<y) {
    if (f >= 0) {
      y--;
      ddF_y += 2;
      f += ddF_y;
    }
    x++;
    ddF_x += 2;
    f += ddF_x;

    tft_drawPixel(x0 + x, y0 + y, color);
    tft_drawPixel(x0 - x, y0 + y, color);
    tft_drawPixel(x0 + x, y0 - y, color);
    tft_drawPixel(x0 - x, y0 - y, color);
    tft_drawPixel(x0 + y, y0 + x, color);
    tft_drawPixel(x0 - y, y0 + x, color);
    tft_drawPixel(x0 + y, y0 - x, color);
    tft_drawPixel(x0 - y, y0 - x, color);
```

```
  }
}

void tft_drawCircleHelper( short x0, short y0, short r, unsigned char cornername,
    ↪ unsigned short color) {
// Helper function for drawing circles and circular objects
  short f = 1 - r;
  short ddF_x = 1;
  short ddF_y = -2 * r;
  short x = 0;
  short y = r;

  while (x<y) {
    if (f >= 0) {
      y--;
      ddF_y += 2;
      f += ddF_y;
    }
    x++;
    ddF_x += 2;
    f += ddF_x;
    if (cornername & 0x4) {
      tft_drawPixel(x0 + x, y0 + y, color);
      tft_drawPixel(x0 + y, y0 + x, color);
    }
    if (cornername & 0x2) {
      tft_drawPixel(x0 + x, y0 - y, color);
      tft_drawPixel(x0 + y, y0 - x, color);
    }
    if (cornername & 0x8) {
      tft_drawPixel(x0 - y, y0 + x, color);
      tft_drawPixel(x0 - x, y0 + y, color);
    }
    if (cornername & 0x1) {
      tft_drawPixel(x0 - y, y0 - x, color);
      tft_drawPixel(x0 - x, y0 - y, color);
    }
  }
}

void tft_fillCircle(short x0, short y0, short r, unsigned short color) {
/* Draw a filled circle with center (x0,y0) and radius r, with given color
 * Parameters:
 * x0: x-coordinate of center of circle. The top-left of the screen
 * has x-coordinate 0 and increases to the right
 * y0: y-coordinate of center of circle. The top-left of the screen
 * has y-coordinate 0 and increases to the bottom
 * r: radius of circle
 * color: 16-bit color value for the circle
 * Returns: Nothing
 */
  tft_drawFastVLine(x0, y0-r, 2*r+1, color);
  tft_fillCircleHelper(x0, y0, r, 3, 0, color);
}
```

```
void tft_fillCircleHelper(short x0, short y0, short r, unsigned char cornername, short
    ↪ delta, unsigned short color) {
// Helper function for drawing filled circles
  short f = 1 - r;
  short ddF_x = 1;
  short ddF_y = -2 * r;
  short x = 0;
  short y = r;

  while (x<y) {
    if (f >= 0) {
      y--;
      ddF_y += 2;
      f += ddF_y;
    }
    x++;
    ddF_x += 2;
    f += ddF_x;

    if (cornername & 0x1) {
      tft_drawFastVLine(x0+x, y0-y, 2*y+1+delta, color);
      tft_drawFastVLine(x0+y, y0-x, 2*x+1+delta, color);
    }
    if (cornername & 0x2) {
      tft_drawFastVLine(x0-x, y0-y, 2*y+1+delta, color);
      tft_drawFastVLine(x0-y, y0-x, 2*x+1+delta, color);
    }
  }
}

void tft_drawLine(short x0, short y0, short x1, short y1, unsigned short color) {
/* Draw a straight line from (x0,y0) to (x1,y1) with given color
 * Parameters:
 * x0: x-coordinate of starting point of line. The x-coordinate of
 * the top-left of the screen is 0. It increases to the right.
 * y0: y-coordinate of starting point of line. The y-coordinate of
 * the top-left of the screen is 0. It increases to the bottom.
 * x1: x-coordinate of ending point of line. The x-coordinate of
 * the top-left of the screen is 0. It increases to the right.
 * y1: y-coordinate of ending point of line. The y-coordinate of
 * the top-left of the screen is 0. It increases to the bottom.
 * color: 16-bit color value for line
 */
  short steep = abs(y1 - y0) > abs(x1 - x0);
  if (steep) {
    swap(x0, y0);
    swap(x1, y1);
  }

  if (x0 > x1) {
    swap(x0, x1);
    swap(y0, y1);
  }
```

```
  short dx, dy;
  dx = x1 - x0;
  dy = abs(y1 - y0);

  short err = dx / 2;
  short ystep;

  if (y0 < y1) {
    ystep = 1;
  } else {
    ystep = -1;
  }

  for (; x0<=x1; x0++) {
    if (steep) {
      tft_drawPixel(y0, x0, color);
    } else {
      tft_drawPixel(x0, y0, color);
    }
    err -= dy;
    if (err < 0) {
      y0 += ystep;
      err += dx;
    }
  }
}

void tft_drawRect(short x, short y, short w, short h, unsigned short color) {
/* Draw a rectangle outline with top left vertex (x,y), width w
 * and height h at given color
 * Parameters:
 * x: x-coordinate of top-left vertex. The x-coordinate of
 * the top-left of the screen is 0. It increases to the right.
 * y: y-coordinate of top-left vertex. The y-coordinate of
 * the top-left of the screen is 0. It increases to the bottom.
 * w: width of the rectangle
 * h: height of the rectangle
 * color: 16-bit color of the rectangle outline
 * Returns: Nothing
 */
  tft_drawFastHLine(x, y, w, color);
  tft_drawFastHLine(x, y+h-1, w, color);
  tft_drawFastVLine(x, y, h, color);
  tft_drawFastVLine(x+w-1, y, h, color);
}

void tft_drawRoundRect(short x, short y, short w, short h, short r, unsigned short color)
    ↪ {
/* Draw a rounded rectangle outline with top left vertex (x,y), width w,
 * height h and radius of curvature r at given color
 * Parameters:
 * x: x-coordinate of top-left vertex. The x-coordinate of
 * the top-left of the screen is 0. It increases to the right.
```

```
 * y: y-coordinate of top-left vertex. The y-coordinate of
 * the top-left of the screen is 0. It increases to the bottom.
 * w: width of the rectangle
 * h: height of the rectangle
 * color: 16-bit color of the rectangle outline
 * Returns: Nothing
 */
  // smarter version
  tft_drawFastHLine(x+r , y , w-2*r, color); // Top
  tft_drawFastHLine(x+r , y+h-1, w-2*r, color); // Bottom
  tft_drawFastVLine(x , y+r , h-2*r, color); // Left
  tft_drawFastVLine(x+w-1, y+r , h-2*r, color); // Right
  // draw four corners
  tft_drawCircleHelper(x+r , y+r , r, 1, color);
  tft_drawCircleHelper(x+w-r-1, y+r , r, 2, color);
  tft_drawCircleHelper(x+w-r-1, y+h-r-1, r, 4, color);
  tft_drawCircleHelper(x+r , y+h-r-1, r, 8, color);
}

void tft_fillRoundRect(short x, short y, short w, short h, short r, unsigned short color)
    ↪ {
  // smarter version
  tft_fillRect(x+r, y, w-2*r, h, color);

  // draw four corners
  tft_fillCircleHelper(x+w-r-1, y+r, r, 1, h-2*r-1, color);
  tft_fillCircleHelper(x+r , y+r, r, 2, h-2*r-1, color);
}

void tft_drawTriangle(short x0, short y0, short x1, short y1, short x2, short y2,
    ↪ unsigned short color) {
/* Draw a triangle outline with vertices (x0,y0),(x1,y1),(x2,y2) with given color
 * Parameters:
 * x0: x-coordinate of one of the 3 vertices
 * y0: y-coordinate of one of the 3 vertices
 * x1: x-coordinate of one of the 3 vertices
 * y1: y-coordinate of one of the 3 vertices
 * x2: x-coordinate of one of the 3 vertices
 * y2: y-coordinate of one of the 3 vertices
 * color: 16-bit color value for outline
 * Returns: Nothing
 */
  tft_drawLine(x0, y0, x1, y1, color);
  tft_drawLine(x1, y1, x2, y2, color);
  tft_drawLine(x2, y2, x0, y0, color);
}

void tft_fillTriangle ( short x0, short y0, short x1, short y1, short x2, short y2,
    ↪ unsigned short color) {
/* Draw a filled triangle with vertices (x0,y0),(x1,y1),(x2,y2) with given color
 * Parameters:
 * x0: x-coordinate of one of the 3 vertices
 * y0: y-coordinate of one of the 3 vertices
 * x1: x-coordinate of one of the 3 vertices
```

```
* y1: y-coordinate of one of the 3 vertices
* x2: x-coordinate of one of the 3 vertices
* y2: y-coordinate of one of the 3 vertices
* color: 16-bit color value
* Returns: Nothing
*/
 short a, b, y, last;

 // Sort coordinates by Y order (y2 >= y1 >= y0)
 if (y0 > y1) {
   swap(y0, y1); swap(x0, x1);
 }
 if (y1 > y2) {
   swap(y2, y1); swap(x2, x1);
 }
 if (y0 > y1) {
   swap(y0, y1); swap(x0, x1);
 }

 if(y0 == y2) { // Handle awkward all-on-same-line case as its own thing
   a = b = x0;
   if(x1 < a) a = x1;
   else if(x1 > b) b = x1;
   if(x2 < a) a = x2;
   else if(x2 > b) b = x2;
   tft_drawFastHLine(a, y0, b-a+1, color);
   return;
 }

 short
   dx01 = x1 - x0,
   dy01 = y1 - y0,
   dx02 = x2 - x0,
   dy02 = y2 - y0,
   dx12 = x2 - x1,
   dy12 = y2 - y1,
   sa = 0,
   sb = 0;

 // For upper part of triangle, find scanline crossings for segments
 // 0-1 and 0-2. If y1=y2 (flat-bottomed triangle), the scanline y1
 // is included here (and second loop will be skipped, avoiding a /0
 // error there), otherwise scanline y1 is skipped here and handled
 // in the second loop...which also avoids a /0 error here if y0=y1
 // (flat-topped triangle).
 if(y1 == y2) last = y1; // Include y1 scanline
 else last = y1-1; // Skip it

 for(y=y0; y<=last; y++) {
   a = x0 + sa / dy01;
   b = x0 + sb / dy02;
   sa += dx01;
   sb += dx02;
   /* longhand:
```

```
    a = x0 + (x1 - x0) * (y - y0) / (y1 - y0);
    b = x0 + (x2 - x0) * (y - y0) / (y2 - y0);
    */
    if(a > b) swap(a,b);
    tft_drawFastHLine(a, y, b-a+1, color);
  }

  // For lower part of triangle, find scanline crossings for segments
  // 0-2 and 1-2. This loop is skipped if y1=y2.
  sa = dx12 * (y - y1);
  sb = dx02 * (y - y0);
  for(; y<=y2; y++) {
    a = x1 + sa / dy12;
    b = x0 + sb / dy02;
    sa += dx12;
    sb += dx02;
    /* longhand:
    a = x1 + (x2 - x1) * (y - y1) / (y2 - y1);
    b = x0 + (x2 - x0) * (y - y0) / (y2 - y0);
    */
    if(a > b) swap(a,b);
    tft_drawFastHLine(a, y, b-a+1, color);
  }
}

void tft_drawBitmap(short x, short y, const unsigned char *bitmap, short w, short h,
    ↪ unsigned short color) {

  short i, j, byteWidth = (w + 7) / 8;

  for(j=0; j<h; j++) {
    for(i=0; i<w; i++ ) {
      if(pgm_read_byte(bitmap + j * byteWidth + i / 8) & (128 >> (i & 7))) {
        tft_drawPixel(x+i, y+j, color);
      }
    }
  }
}

void tft_write(unsigned char c){
  if (c == '\n') {
    cursor_y += textsize*8;
    cursor_x = 0;
  } else if (c == '\r') {
    // skip em
  } else if (c == '\t'){
      int new_x = cursor_x + tabspace;
      if (new_x < _width){
          cursor_x = new_x;
      }
  } else {
    tft_drawChar(cursor_x, cursor_y, c, textcolor, textbgcolor, textsize);
    cursor_x += textsize*6;
    if (wrap && (cursor_x > (_width - textsize*6))) {
```

```
      cursor_y += textsize*8;
      cursor_x = 0;
    }
  }
}

inline void tft_writeString(char* str){
/* Print text onto screen
 * Call tft_setCursor(), tft_setTextColor(), tft_setTextSize()
 * as necessary before printing
 */
    while (*str){
        tft_write(*str++);
    }
}

void tft_drawChar(short x, short y, unsigned char c, unsigned short color, unsigned short
    ↪ bg, unsigned char size) {
    char i, j;
  if((x >= _width) || // Clip right
     (y >= _height) || // Clip bottom
     ((x + 6 * size - 1) < 0) || // Clip left
     ((y + 8 * size - 1) < 0)) // Clip top
    return;

  for (i=0; i<6; i++ ) {
    unsigned char line;
    if (i == 5)
      line = 0x0;
    else
      line = pgm_read_byte(font+(c*5)+i);
    for ( j = 0; j<8; j++) {
      if (line & 0x1) {
        if (size == 1) // default size
          tft_drawPixel(x+i, y+j, color);
        else { // big size
          tft_fillRect(x+(i*size), y+(j*size), size, size, color);
        }
      } else if (bg != color) {
        if (size == 1) // default size
          tft_drawPixel(x+i, y+j, bg);
        else { // big size
          tft_fillRect(x+i*size, y+j*size, size, size, bg);
        }
      }
      line >>= 1;
    }
  }
}

inline void tft_setCursor(short x, short y) {
/* Set cursor for text to be printed
 * Parameters:
 * x = x-coordinate of top-left of text starting
```

```
 * y = y-coordinate of top-left of text starting
 * Returns: Nothing
 */
  cursor_x = x;
  cursor_y = y;
}

inline void tft_setTextSize(unsigned char s) {
/*Set size of text to be displayed
 * Parameters:
 * s = text size (1 being smallest)
 * Returns: nothing
 */
  textsize = (s > 0) ? s : 1;
}

inline void tft_setTextColor(unsigned short c) {
  // For 'transparent' background, we'll set the bg
  // to the same as fg instead of using a flag
  textcolor = textbgcolor = c;
}

inline void tft_setTextColor2(unsigned short c, unsigned short b) {
/* Set color of text to be displayed
 * Parameters:
 * c = 16-bit color of text
 * b = 16-bit color of text background
 */
  textcolor = c;
  textbgcolor = b;
}

inline void tft_setTextWrap(char w) {
  wrap = w;
}

void tft_gfx_setRotation(unsigned char x) {
/* Set display rotation in 90 degree steps
 * Parameters:
 * x: dictate direction of rotation
 * 0 = no rotation (0 degree rotation)
 * 1 = rotate 90 degree clockwise
 * 2 = rotate 180 degree
 * 3 = rotate 90 degree anticlockwise
 * Returns: Nothing
 */
  rotation = (x & 3);
  switch(rotation) {
   case 0:
   case 2:
    _width = ILI9340_TFTWIDTH;
    _height = ILI9340_TFTHEIGHT;
    break;
   case 1:
```

```
   case 3:
    _width = ILI9340_TFTHEIGHT;;
    _height = ILI9340_TFTWIDTH;
    break;
  }
}


/* MASTER *****************************************************************/


inline void Mode16(void){ // configure SPI1 for 16-bit mode
    SPI1STATbits.SPIEN = 0;
    SPI1CON1bits.MODE16 = 1;
    SPI1STATbits.SPIEN = 1;
}

inline void Mode8(void){ // configure SPI1 for 8-bit mode
    SPI1STATbits.SPIEN = 0;
    SPI1CON1bits.MODE16 = 0;
    SPI1STATbits.SPIEN = 1;
}

unsigned char tft_spiwrite(unsigned char c){ // Transfer to SPI
    SPI1BUF = c;
    while(!SPI1STATbits.SPIRBF){}
    return(SPI1BUF);
}

void tft_spiwrite8(unsigned char c) { // Transfer one byte c to SPI
 /* The default mode for me is to transfer 16-bits at once
  * However, it is necessary sometimes to transfer only 8-bits at a time
  * But this is required less often than 16-bits at once
  * So, the default mode is 16-bit mode and is switched to 8-bit mode when
  * required, and then switched back at the end of the function
  */
    Mode8(); // switch to 8-bit mode
    tft_spiwrite(c);
    Mode16(); // switch back to 16-bit mode
}

unsigned char tft_spiwrite16(unsigned short c){ // Transfer two bytes "c" to SPI
    SPI1BUF = c;
    while(!SPI1STATbits.SPIRBF){}
    return(SPI1BUF);
}

void tft_writecommand(unsigned char c) {
    _dc_low();
    _cs_low();

    tft_spiwrite8(c);

    _cs_high();
```

```
}

void tft_writecommand16(unsigned short c) {
    _dc_low();
    _cs_low();

    tft_spiwrite16(c);

    _cs_high();

}

void tft_writedata(unsigned char c) {
    _dc_high();
    _cs_low();

    tft_spiwrite8(c);

    _cs_high();

}

void tft_writedata16(unsigned short c) {
    _dc_high();
    _cs_low();

    tft_spiwrite16(c);

    _cs_high();

}

void tft_begin(void) {

  TRIS_rst = 0;
  _rst_low();
  TRIS_dc = 0;
  TRIS_cs = 0;

  _dc_low();
  _cs_high();


// SpiChnOpen(1, SPI_OPEN_MSTEN | SPI_OPEN_MODE8 | SPI_OPEN_ON |
// SPI_OPEN_DISSDI | SPI_OPEN_CKE_REV , PBCLK/SPI_freq);

    // Start with 8-bit mode for initialization - move to 16-bit mode once
    // that's done

  _rst_high();
// delay_ms(5);
  _rst_low();
// delay_ms(20);
```

```
  _rst_high();
// delay_ms(150);

  tft_writecommand(0xEF);
  tft_writedata(0x03);
  tft_writedata(0x80);
  tft_writedata(0x02);

  tft_writecommand(0xCF);
  tft_writedata(0x00);
  tft_writedata(0xC1);
  tft_writedata(0x30);

  tft_writecommand(0xED);
  tft_writedata(0x64);
  tft_writedata(0x03);
  tft_writedata(0x12);
  tft_writedata(0x81);

  tft_writecommand(0xE8);
  tft_writedata(0x85);
  tft_writedata(0x00);
  tft_writedata(0x78);

  tft_writecommand(0xCB);
  tft_writedata(0x39);
  tft_writedata(0x2C);
  tft_writedata(0x00);
  tft_writedata(0x34);
  tft_writedata(0x02);

  tft_writecommand(0xF7);
  tft_writedata(0x20);

  tft_writecommand(0xEA);
  tft_writedata(0x00);
  tft_writedata(0x00);

  tft_writecommand(ILI9340_PWCTR1); //Power control
  tft_writedata(0x23); //VRH[5:0]

  tft_writecommand(ILI9340_PWCTR2); //Power control
  tft_writedata(0x10); //SAP[2:0];BT[3:0]

  tft_writecommand(ILI9340_VMCTR1); //VCM control
  tft_writedata(0x3e);
  tft_writedata(0x28);

  tft_writecommand(ILI9340_VMCTR2); //VCM control2
  tft_writedata(0x86);

  tft_writecommand(ILI9340_MADCTL); // Memory Access Control
  tft_writedata(ILI9340_MADCTL_MX | ILI9340_MADCTL_BGR);
```

```
  tft_writecommand(ILI9340_PIXFMT);
  tft_writedata(0x55);

  tft_writecommand(ILI9340_FRMCTR1);
  tft_writedata(0x00);
  tft_writedata(0x18);

  tft_writecommand(ILI9340_DFUNCTR); // Display Function Control
  tft_writedata(0x08);
  tft_writedata(0x82);
  tft_writedata(0x27);

  tft_writecommand(0xF2); // 3Gamma Function Disable
  tft_writedata(0x00);

  tft_writecommand(ILI9340_GAMMASET); //Gamma curve selected
  tft_writedata(0x01);

  tft_writecommand(ILI9340_GMCTRP1); //Set Gamma
  tft_writedata(0x0F);
  tft_writedata(0x31);
  tft_writedata(0x2B);
  tft_writedata(0x0C);
  tft_writedata(0x0E);
  tft_writedata(0x08);
  tft_writedata(0x4E);
  tft_writedata(0xF1);
  tft_writedata(0x37);
  tft_writedata(0x07);
  tft_writedata(0x10);
  tft_writedata(0x03);
  tft_writedata(0x0E);
  tft_writedata(0x09);
  tft_writedata(0x00);

  tft_writecommand(ILI9340_GMCTRN1); //Set Gamma
  tft_writedata(0x00);
  tft_writedata(0x0E);
  tft_writedata(0x14);
  tft_writedata(0x03);
  tft_writedata(0x11);
  tft_writedata(0x07);
  tft_writedata(0x31);
  tft_writedata(0xC1);
  tft_writedata(0x48);
  tft_writedata(0x08);
  tft_writedata(0x0F);
  tft_writedata(0x0C);
  tft_writedata(0x31);
  tft_writedata(0x36);
  tft_writedata(0x0F);

  tft_writecommand(ILI9340_SLPOUT); //Exit Sleep
// delay_ms(120);
```

```
  tft_writecommand(ILI9340_DISPON); //Display on

  // Now move to 16-bit mode to speed things up for display
  Mode16();
}

void tft_setAddrWindow(unsigned short x0, unsigned short y0, unsigned short x1, unsigned
    ↪ short y1) {

  tft_writecommand(ILI9340_CASET); // Column addr set
  tft_writedata16(x0);
  tft_writedata16(x1);

  tft_writecommand(ILI9340_PASET); // Row addr set
  tft_writedata16(y0);
  tft_writedata16(y1);

  tft_writecommand(ILI9340_RAMWR); // write to RAM
}

void tft_pushColor(unsigned short color) {
  _dc_high();
  _cs_low();

  tft_spiwrite16(color);

  _cs_high();
}

void tft_drawPixel(short x, short y, unsigned short color) {
/* Draw a pixel at location (x,y) with given color
 * Parameters:
 * x: x-coordinate of pixel to draw; top left of screen is x=0
 * and x increases to the right
 * y: y-coordinate of pixel to draw; top left of screen is y=0
 * and y increases to the bottom
 * color: 16-bit color value
 * Returns: Nothing
 */

  if((x < 0) ||(x >= _width) || (y < 0) || (y >= _height)) return;

  tft_setAddrWindow(x,y,x+1,y+1);

  _dc_high();
  _cs_low();

  tft_spiwrite16(color);

  _cs_high();
}

void tft_drawFastVLine(short x, short y, short h, unsigned short color) {
/* Draw a vertical line at location from (x,y) to (x,y+h-1) with color
```

```
 * Parameters:
 * x: x-coordinate line to draw; top left of screen is x=0
 * and x increases to the right
 * y: y-coordinate of starting point of line; top left of screen is y=0
 * and y increases to the bottom
 * h: height of line to draw
 * color: 16-bit color value
 * Returns: Nothing
 */

  // Rudimentary clipping
  if((x >= _width) || (y >= _height)) return;

  if((y+h-1) >= _height)
    h = _height-y;

  tft_setAddrWindow(x, y, x, y+h-1);

  _dc_high();
  _cs_low();

  while (h--) {
      tft_spiwrite16(color);
  }

  _cs_high();
}

void tft_drawFastHLine(short x, short y, short w, unsigned short color) {
/* Draw a horizontal line at location from (x,y) to (x+w-1,y) with color
 * Parameters:
 * x: x-coordinate starting point of line; top left of screen is x=0
 * and x increases to the right
 * y: y-coordinate of starting point of line; top left of screen is y=0
 * and y increases to the bottom
 * w: width of line to draw
 * color: 16-bit color value
 * Returns: Nothing
 */

  // Rudimentary clipping
  if((x >= _width) || (y >= _height)) return;
  if((x+w-1) >= _width) w = _width-x;
  tft_setAddrWindow(x, y, x+w-1, y);

  _dc_high();
  _cs_low();

  while (w--) {
      tft_spiwrite16(color);
  }

  _cs_high();
}
```

```
void tft_fillScreen(unsigned short color) {
/* Fill entire screen with given color
 * Parameters:
 * color: 16-bit color value
 * Returns: Nothing
 */
   tft_fillRect(0, 0, _width, _height, color);
}

void tft_fillRect(short x, short y, short w, short h, unsigned short color) {
/* Draw a filled rectangle with starting top-left vertex (x,y),
 * width w and height h with given color
 * Parameters:
 * x: x-coordinate of top-left vertex; top left of screen is x=0
 * and x increases to the right
 * y: y-coordinate of top-left vertex; top left of screen is y=0
 * and y increases to the bottom
 * w: width of rectangle
 * h: height of rectangle
 * color: 16-bit color value
 * Returns: Nothing
 */

  // rudimentary clipping (drawChar w/big text requires this)
  if((x >= _width) || (y >= _height)) return;
  if((x + w - 1) >= _width) w = _width - x;
  if((y + h - 1) >= _height) h = _height - y;

  tft_setAddrWindow(x, y, x+w-1, y+h-1);

  _dc_high();
  _cs_low();

  for(y=h; y>0; y--) {
    for(x=w; x>0; x--) {
        tft_spiwrite16(color);
    }
  }

  _cs_high();
}

inline unsigned short tft_Color565(unsigned char r, unsigned char g, unsigned char b) {
/* Pass 8-bit (each) R,G,B, get back 16-bit packed color
 * Parameters:
 * r: 8-bit R/red value from RGB
 * g: 8-bit g/green value from RGB
 * b: 8-bit b/blue value from RGB
 * Returns:
 * 16-bit packed color value for color info
 */
  return ((r & 0xF8) << 8) | ((g & 0xFC) << 3) | (b >> 3);
}
```

```
void tft_setRotation(unsigned char m) {
  unsigned char rotation;
  tft_writecommand(ILI9340_MADCTL);
  rotation = m % 4; // can't be higher than 3
  switch (rotation) {
   case 0:
     tft_writedata(ILI9340_MADCTL_MX | ILI9340_MADCTL_BGR);
     _width = ILI9340_TFTWIDTH;
     _height = ILI9340_TFTHEIGHT;
     break;
   case 1:
     tft_writedata(ILI9340_MADCTL_MV | ILI9340_MADCTL_BGR);
     _width = ILI9340_TFTHEIGHT;
     _height = ILI9340_TFTWIDTH;
     break;
  case 2:
    tft_writedata(ILI9340_MADCTL_MY | ILI9340_MADCTL_BGR);
     _width = ILI9340_TFTWIDTH;
     _height = ILI9340_TFTHEIGHT;
    break;
   case 3:
     tft_writedata(ILI9340_MADCTL_MV | ILI9340_MADCTL_MY | ILI9340_MADCTL_MX |
         ↪ ILI9340_MADCTL_BGR);
     _width = ILI9340_TFTHEIGHT;
     _height = ILI9340_TFTWIDTH;
     break;
  }
}

void testText() {
  tft_fillScreen(ILI9340_BLACK);
  tft_setCursor(0, 0);
  tft_setTextColor(ILI9340_WHITE); tft_setTextSize(1);
  tft_writeString("Hello␣World!\n");
  tft_setTextColor(ILI9340_YELLOW); tft_setTextSize(2);
  tft_setTextColor(ILI9340_RED); tft_setTextSize(3);
  tft_writeString("DEADBEEF\n");
  tft_setTextColor(ILI9340_GREEN);
  tft_setTextSize(5);
  tft_writeString("Groop\n");
  tft_setTextSize(2);
  tft_writeString("I␣implore␣thee\n");
  tft_setTextSize(1);

  tft_writeString("my␣foonting␣turlingdromes.\n");
  tft_writeString("And␣hooptiously␣drangle␣me\n");
  tft_writeString("with␣crinkly␣bindlewurdles,\n");
  tft_writeString("Or␣I␣will␣rend␣thee\n");
  tft_writeString("in␣the␣gobberwarts\n");
  tft_writeString("with␣my␣blurglecruncheon,\n");
  tft_writeString("see␣if␣I␣don't!\n");
}
```

```
void testFillScreen() {

  tft_fillScreen(ILI9340_BLUE);

}

//void delay_ms(unsigned long i){
///* Create a software delay about i ms long
// * Parameters:
// * i: equal to number of milliseconds for delay
// * Returns: Nothing
// * Note: Uses Core Timer. Core Timer is cleared at the initialiazion of
// * this function. So, applications sensitive to the Core Timer are going
// * to be affected
// */
// unsigned int j;
// j = dTime_ms * i;
// WriteCoreTimer(0);
// while (ReadCoreTimer() < j);
//}
//
//void delay_us(unsigned long i){
///* Create a software delay about i us long
// * Parameters:
// * i: equal to number of microseconds for delay
// * Returns: Nothing
// * Note: Uses Core Timer. Core Timer is cleared at the initialiazion of
// * this function. So, applications sensitive to the Core Timer are going
// * to be affected
// */
// unsigned int j;
// j = dTime_us * i;
// WriteCoreTimer(0);
// while (ReadCoreTimer() < j);
//}

void tft_invertDisplay(int i) {
  tft_writecommand(i ? ILI9340_INVON : ILI9340_INVOFF);
}


////////// stuff not actively being used, but kept for posterity


//unsigned char tft_spiread(void) {
// unsigned char r = 0;
//
// /*
// * ADD SPI INTERFACE CODE
    ↪ ----------------------------------------------*/**************************
// */
// //Serial.print("read: 0x"); Serial.print(r, HEX);
//
```

```
// return r;
//}

// unsigned char tft_readdata(void) {
// unsigned char r;
// _dc_high();
// _cs_low();
// r = tft_spiread();
// _cs_high();
// return r;
//
//}
//
//
// unsigned char tft_readcommand8(unsigned char c) {
// _dc_low();
//// _sclk = 0;
// _cs_low();
// tft_spiwrite8(c);
//
// _dc_high();
// unsigned char r = tft_spiread();
// _cs_high();
// return r;
//
// /*
// digitalWrite(_dc, LOW);
// digitalWrite(_sclk, LOW);
// digitalWrite(_cs, LOW);
// spiwrite(c);
//
// digitalWrite(_dc, HIGH);
// unsigned char r = spiread();
// digitalWrite(_cs, HIGH);
// return r;
// */
//}
```

C.1.8 config.h

```
/*
 * File: config.h
 * Author: Kevin Schneier
 *
 */

#ifndef CONFIG_H
#define CONFIG_H


#pragma config FNOSC = FRCPLL
#pragma config FCKSM = CSECMD
#pragma config OSCIOFNC = ON
#pragma config FWDTEN = OFF
#pragma config FPWRT = PWR128
#pragma config ICS = PGD1
#pragma config JTAGEN = OFF



#endif /* XC_HEADER_TEMPLATE_H */
```

C.1.9 fft.h

```
/*
 * File: fft.h
 * Author: Kevin Schneier
 *
 */

#ifndef _FFT_H_
#define _FFT_H_

#include <xc.h>
#include <stdint.h>

/*************************************************************************
 This parameter defines number of points/samples for the calculation.
 The valid values are 8, 16, 32, 64, 128, 256, 512, 1024 and 2048.
 If required transform size doesnt match exactly the valid sizes
 then the size should be rounded to largest valid size and
 not used points should be set to zero.
 *************************************************************************/
#define FFT_POINTS_NUMBER 1024

/*************************************************************************
 Function:

 void FFT(int16_t* pRe, int16_t* pIm)

 Description:

 This function calculates FFT for the points provided in real and imaginary parts arrays.
 In most cases the input signal is real (not complex), the data points must be stored in
 the real part array and the imaginary array elements must be set to zero.
 The sizes of the real and imaginary arrays must be equal to the transform size defined
 by FFT_POINTS_NUMBER in fft.h file. After calculation the result (harmonics) are written
 to the same input data arrays.

 Parameters:

 int16_t* re  pointer to array with the input signal vector (real part).
 int16_t* im  pointer to array with the input signal vector (imaginary part),
            for real signals this vector values must be set to zero.

 Returned data:

 int16_t* re  pointer to array where the harmonics are stored (real part).
 int16_t* im  pointer to array where the harmonics are stored (imaginary part).
*************************************************************************/
void FFT(int16_t* pRe, int16_t* pIm);

/*************************************************************************
 Function:

 uint16_t FFTAmplitude(int16_t re, int16_t im)
```

```
 Description:

 This function calculates absolute value (harmonic amplitude) of
 the complex number (A = SQRT(Re^2 + Im^2)).

 Parameters:

 int16_t re  real part of the complex number.
 int16_t im imaginary part of the complex number.

 Returned data:

 The function returns the absolute value (harmonic amplitude)
 for the complex number entered.
*************************************************************************/
uint16_t FFTAmplitude(int16_t re, int16_t im);

/*************************************************************************
 Function:

 int16_t FFTPhase(int16_t re, int16_t im);

 Description:

 This function calculates an argument (harmonic phase) of the complex number
 (D = (180/PI)*ARCTAN(Im/Re)) in degrees with maximum error +-1 degree.

 Parameters:

 short re  real part of the complex number.
 short im  imaginary part of the complex number.

 Returned data:
 The function returns an argument (harmonic phase) in degrees
 for the complex number entered.
*************************************************************************/
int16_t FFTPhase(int16_t re, int16_t im);

#endif
```

C.1.10 fundamental.h

```
/*
 * File: fundamental.h
 * Author: Kevin Schneier
 *
 */

#ifndef FUNDAMENTAL_H
#define FUNDAMENTAL_H

#include <xc.h> // include processor files - each processor file is guarded.

#define PI 3.14159265358979323846264
#define FPLENGTH 50

double extract(uint16_t *fft, int Fs, int n);
int indexExtract(uint16_t *fft, int numPoints);
//int *extractAll(uint16_t *fft, int numPoints);


#endif /* XC_HEADER_TEMPLATE_H */
```

C.1.11 touchCapability.h

```
// Touch screen library with X Y and Z (pressure) readings as well
// as oversampling to avoid 'bouncing'
// (c) ladyada / adafruit
// Code under MIT License

#ifndef TOUCHCAPABILITY_H
#define TOUCHCAPABILITY_H

#include <stdint.h>

extern int ADC_AN3_val;
extern int ADC_AN2_val;
extern int touchReady;
extern int coords[];

int *getPoint(void);

#endif
```

C.1.12 touchscreen.h

```
/*
 * File: main.c
 * Author: Kevin Schneier
 *
 */

#ifndef TOUCHSCREEN_H
#define TOUCHSCREEN_H

#include <xc.h> // include processor files - each processor file is guarded.


#define pgm_read_byte(addr) (*(const unsigned char *)(addr))
#define swap(a, b) { short t = a; a = b; b = t; }

#define PBCLK 40000000 // peripheral bus clock
#define SPI_freq 20000000

#define tabspace 4 // number of spaces for a tab
#define DELAY 0x80
#define dTime_ms PBCLK/2000
#define dTime_us PBCLK/2000000

/* Master ***************************/
#define _dc LATDbits.LATD9
#define TRIS_dc TRISDbits.TRISD9
#define _dc_high() {_dc = 1;}
#define _dc_low() {_dc = 0;}

#define _cs LATBbits.LATB14
#define TRIS_cs TRISBbits.TRISB14
#define _cs_high() {_cs = 1;}
#define _cs_low() {_cs = 0;}

#define _rst LATBbits.LATB2
#define TRIS_rst TRISBbits.TRISB2
#define _rst_high() {_rst = 1;}
#define _rst_low() {_rst = 0;}

#define ILI9340_TFTWIDTH 240
#define ILI9340_TFTHEIGHT 320

#define ILI9340_NOP 0x00
#define ILI9340_SWRESET 0x01
#define ILI9340_RDDID 0x04
#define ILI9340_RDDST 0x09

#define ILI9340_SLPIN 0x10
#define ILI9340_SLPOUT 0x11
#define ILI9340_PTLON 0x12
#define ILI9340_NORON 0x13
```

```
#define ILI9340_RDMODE 0x0A
#define ILI9340_RDMADCTL 0x0B
#define ILI9340_RDPIXFMT 0x0C
#define ILI9340_RDIMGFMT 0x0A
#define ILI9340_RDSELFDIAG 0x0F

#define ILI9340_INVOFF 0x20
#define ILI9340_INVON 0x21
#define ILI9340_GAMMASET 0x26
#define ILI9340_DISPOFF 0x28
#define ILI9340_DISPON 0x29

#define ILI9340_CASET 0x2A
#define ILI9340_PASET 0x2B
#define ILI9340_RAMWR 0x2C
#define ILI9340_RAMRD 0x2E

#define ILI9340_PTLAR 0x30
#define ILI9340_MADCTL 0x36


#define ILI9340_MADCTL_MY 0x80
#define ILI9340_MADCTL_MX 0x40
#define ILI9340_MADCTL_MV 0x20
#define ILI9340_MADCTL_ML 0x10
#define ILI9340_MADCTL_RGB 0x00
#define ILI9340_MADCTL_BGR 0x08
#define ILI9340_MADCTL_MH 0x04

#define ILI9340_PIXFMT 0x3A

#define ILI9340_FRMCTR1 0xB1
#define ILI9340_FRMCTR2 0xB2
#define ILI9340_FRMCTR3 0xB3
#define ILI9340_INVCTR 0xB4
#define ILI9340_DFUNCTR 0xB6

#define ILI9340_PWCTR1 0xC0
#define ILI9340_PWCTR2 0xC1
#define ILI9340_PWCTR3 0xC2
#define ILI9340_PWCTR4 0xC3
#define ILI9340_PWCTR5 0xC4
#define ILI9340_VMCTR1 0xC5
#define ILI9340_VMCTR2 0xC7

#define ILI9340_RDID1 0xDA
#define ILI9340_RDID2 0xDB
#define ILI9340_RDID3 0xDC
#define ILI9340_RDID4 0xDD

#define ILI9340_GMCTRP1 0xE0
#define ILI9340_GMCTRN1 0xE1
/*
#define ILI9340_PWCTR6 0xFC
```

```
*/

// Color definitions
#define ILI9340_BLACK 0x0000
#define ILI9340_BLUE 0x001F
#define ILI9340_RED 0xF800
#define ILI9340_GREEN 0x07E0
#define ILI9340_CYAN 0x07FF
#define ILI9340_MAGENTA 0xF81F
#define ILI9340_YELLOW 0xFFE0
#define ILI9340_WHITE 0xFFFF



extern unsigned short _width;
extern unsigned short _height;
extern unsigned short cursor_y;
extern unsigned short cursor_x;
extern unsigned short textsize;
extern unsigned short textcolor;
extern unsigned short textbgcolor;
extern unsigned short wrap;
extern unsigned short rotation;


/* Function Declarations */
/* GFX */
void tft_drawLine(short x0, short y0, short x1, short y1, unsigned short color);
void tft_drawRect(short x, short y, short w, short h, unsigned short color);

void tft_drawCircle(short x0, short y0, short r, unsigned short color);
void tft_drawCircleHelper(short x0, short y0, short r, unsigned char cornername,
     unsigned short color);
void tft_fillCircle(short x0, short y0, short r, unsigned short color);
void tft_fillCircleHelper(short x0, short y0, short r, unsigned char cornername,
     short delta, unsigned short color);
void tft_drawTriangle(short x0, short y0, short x1, short y1,
     short x2, short y2, unsigned short color);
void tft_fillTriangle(short x0, short y0, short x1, short y1,
     short x2, short y2, unsigned short color);
void tft_drawRoundRect(short x0, short y0, short w, short h,
     short radius, unsigned short color);
void tft_fillRoundRect(short x0, short y0, short w, short h, short radius, unsigned short
    ↪  color);
void tft_drawBitmap(short x, short y, const unsigned char *bitmap, short w, short h,
    ↪ unsigned short color);
void tft_drawChar(short x, short y, unsigned char c, unsigned short color, unsigned short
    ↪  bg, unsigned char size);
void tft_setCursor(short x, short y);
void tft_setTextColor(unsigned short c);
void tft_setTextColor2(unsigned short c, unsigned short bg);
void tft_setTextSize(unsigned char s);
```

```
void tft_setTextWrap(char w);
void tft_gfx_setRotation(unsigned char r);
void tft_write(unsigned char c);
void tft_writeString(char* str); // This is the function to use to write a string


/* Master */
void tft_init_hw(void);
unsigned char tft_spiwrite(unsigned char c);
void tft_spiwrite8(unsigned char c);
unsigned char tft_spiwrite16(unsigned short c);
void tft_writecommand(unsigned char c);
void tft_writecommand16(unsigned short c);
void tft_writedata(unsigned char c);
void tft_writedata16(unsigned short c);
void tft_commandList(unsigned char *addr);
void tft_begin(void);
void tft_setAddrWindow(unsigned short x0, unsigned short y0, unsigned short x1, unsigned
    ↪ short y1);
void tft_pushColor(unsigned short color);
void tft_drawPixel(short x, short y, unsigned short color);
void tft_drawFastVLine(short x, short y, short h, unsigned short color);
void tft_drawFastHLine(short x, short y, short w, unsigned short color);
void tft_fillScreen(unsigned short color);
void tft_fillRect(short x, short y, short w, short h, unsigned short color);
unsigned short tft_Color565(unsigned char r, unsigned char g, unsigned char b);
void tft_setRotation(unsigned char m);
//unsigned char tft_spiread(void);
//unsigned char tft_readdata(void);
//unsigned char tft_readcommand8(unsigned char c);
void testFillScreen(void);
void testText(void);
void tft_invertDisplay(int i);

#endif /* XC_HEADER_TEMPLATE_H */
```

C.2 Xcode

C.2.1 AppDelegate.swift

```
//
// AppDelegate.swift
// uTune
//
// Created by Kevin Schneier on 3/11/19.
//

import UIKit

@UIApplicationMain
class AppDelegate: UIResponder, UIApplicationDelegate {

    var window: UIWindow?


    func application(_ application: UIApplication, didFinishLaunchingWithOptions
        ↪ launchOptions: [UIApplicationLaunchOptionsKey: Any]?) -> Bool {
            // Override point for customization after application launch.

        // Delay start so that we can see launch screen
        RunLoop.current.run(until: NSDate(timeIntervalSinceNow: 0.25) as Date)
        return true
    }

    func applicationWillResignActive(_ application: UIApplication) {
        // Sent when the application is about to move from active to inactive state. This
            ↪ can occur for certain types of temporary interruptions (such as an incoming
            ↪  phone call or SMS message) or when the user quits the application and it
            ↪ begins the transition to the background state.
        // Use this method to pause ongoing tasks, disable timers, and invalidate graphics
            ↪  rendering callbacks. Games should use this method to pause the game.
    }

    func applicationDidEnterBackground(_ application: UIApplication) {
        // Use this method to release shared resources, save user data, invalidate timers,
            ↪  and store enough application state information to restore your application
            ↪  to its current state in case it is terminated later.
        // If your application supports background execution, this method is called
            ↪ instead of applicationWillTerminate: when the user quits.
    }

    func applicationWillEnterForeground(_ application: UIApplication) {
        // Called as part of the transition from the background to the active state; here
            ↪ you can undo many of the changes made on entering the background.
    }

    func applicationDidBecomeActive(_ application: UIApplication) {
        // Restart any tasks that were paused (or not yet started) while the application
            ↪ was inactive. If the application was previously in the background,
            ↪ optionally refresh the user interface.
```

```
    }

    func applicationWillTerminate(_ application: UIApplication) {
        // Called when the application is about to terminate. Save data if appropriate.
            ↪ See also applicationDidEnterBackground:.changes made
    }


}
```

C.2.2 BLECentralViewController.swift

```
//
// BLECentralViewController.swift
// uTune
//
// Created by Kevin Schneier on 3/11/19.
//

import Foundation
import UIKit
import CoreBluetooth


var txCharacteristic : CBCharacteristic?
var rxCharacteristic : CBCharacteristic?
var blePeripheral : CBPeripheral?
var characteristicASCIIValue = NSString()



class BLECentralViewController : UIViewController, CBCentralManagerDelegate,
    ↪ CBPeripheralDelegate, UITableViewDelegate, UITableViewDataSource{

    //Data
    var centralManager : CBCentralManager!
    var RSSIs = [NSNumber]()
    var data = NSMutableData()
    var writeData: String = ""
    var peripherals: [CBPeripheral] = []
    var characteristicValue = [CBUUID: NSData]()
    var timer = Timer()
    var characteristics = [String : CBCharacteristic]()


    //UI
    @IBOutlet weak var baseTableView: UITableView!
    @IBOutlet weak var refreshButton: UIBarButtonItem!

    @IBAction func refreshAction(_ sender: AnyObject) {
        disconnectFromDevice()
        self.peripherals = []
        self.RSSIs = []
        self.baseTableView.reloadData()
        startScan()
    }

    override func viewDidLoad() {
        super.viewDidLoad()
        self.baseTableView.delegate = self
        self.baseTableView.dataSource = self
        self.baseTableView.reloadData()

        /*Our key player in this app will be our CBCentralManager. CBCentralManager
```

```
            ↪ objects are used to manage discovered or connected remote peripheral
            ↪ devices (represented by CBPeripheral objects), including scanning for,
            ↪ discovering, and connecting to advertising peripherals.
         */
        centralManager = CBCentralManager(delegate: self, queue: nil)
// let backButton = UIBarButtonItem(title: "Disconnect", style: .plain, target: nil,
    ↪ action: nil)
// navigationItem.backBarButtonItem = backButton
    }

    override func viewDidAppear(_ animated: Bool) {
        disconnectFromDevice()
        super.viewDidAppear(animated)
        refreshScanView()
        print("View Cleared")
    }

    override func viewWillDisappear(_ animated: Bool) {
        super.viewWillDisappear(animated)
        print("Stop Scanning")
        centralManager?.stopScan()
    }

    /*Okay, now that we have our CBCentalManager up and running, it's time to start
        ↪ searching for devices. You can do this by calling the "scanForPeripherals"
        ↪ method.*/

    func startScan() {
        peripherals = []
        print("Now Scanning...")
        self.timer.invalidate()
        centralManager?.scanForPeripherals(withServices: [Advert_UUID] , options: [
            ↪ CBCentralManagerScanOptionAllowDuplicatesKey:false])
        Timer.scheduledTimer(timeInterval: 17, target: self, selector: #selector(self.
            ↪ cancelScan), userInfo: nil, repeats: false)
    }

    /*We also need to stop scanning at some point so we'll also create a function that
        ↪ calls "stopScan"*/
    @objc func cancelScan() {
        self.centralManager?.stopScan()
        print("Scan Stopped")
        print("Number of Peripherals Found: \(peripherals.count)")
    }

    func refreshScanView() {
        baseTableView.reloadData()
    }

    //-Terminate all Peripheral Connection
    /*
     Call this when things either go wrong, or you're done with the connection.
     This cancels any subscriptions if there are any, or straight disconnects if not.
     (didUpdateNotificationStateForCharacteristic will cancel the connection if a
```

```
     ↪ subscription is involved)
 */
func disconnectFromDevice () {
    if blePeripheral != nil {
        // We have a connection to the device but we are not subscribed to the
            ↪ Transfer Characteristic for some reason.
        // Therefore, we will just disconnect from the peripheral
        centralManager?.cancelPeripheralConnection(blePeripheral!)
    }
}


func restoreCentralManager() {
    //Restores Central Manager delegate if something went wrong
    centralManager?.delegate = self
}

/*
 Called when the central manager discovers a peripheral while scanning. Also, once
     ↪ peripheral is connected, cancel scanning.
 */
func centralManager(_ central: CBCentralManager, didDiscover peripheral: CBPeripheral
    ↪ ,advertisementData: [String : Any], rssi RSSI: NSNumber) {

    blePeripheral = peripheral
    self.peripherals.append(peripheral)
    self.RSSIs.append(RSSI)
    peripheral.delegate = self
    self.baseTableView.reloadData()
    if blePeripheral == nil {
        print("Found new pheripheral devices with services")
        print("Peripheral name: \(String(describing: peripheral.name))")
        print("***********************************")
        print ("Advertisement Data : \(advertisementData)")
    }
}

//Peripheral Connections: Connecting, Connected, Disconnected

//-Connection
func connectToDevice () {
    centralManager?.connect(blePeripheral!, options: nil)
}

/*
 Invoked when a connection is successfully created with a peripheral.
 This method is invoked when a call to connect(_:options:) is successful. You
     ↪ typically implement this method to set the peripherals delegate and to
     ↪ discover its services.
 */
//-Connected
func centralManager(_ central: CBCentralManager, didConnect peripheral: CBPeripheral)
    ↪  {
    print("*****************************")
```

```
        print("Connection complete")
        print("Peripheral info: \(String(describing: blePeripheral))")

        //Stop Scan- We don't need to scan once we've connected to a peripheral. We got
            ↪ what we came for.
        centralManager?.stopScan()
        print("Scan Stopped")

        //Erase data that we might have
        data.length = 0

        //Discovery callback
        peripheral.delegate = self
        //Only look for services that matches transmit uuid
        peripheral.discoverServices([BLEService_UUID])

        // Transition to the "Schemes" tab
        let index = 1
        animateTabBarChange(tabBarController: self.tabBarController!, to: self.
            ↪ tabBarController!.viewControllers![index])
        self.tabBarController?.selectedIndex = index

    }


    /*
     Invoked when the central manager fails to create a connection with a peripheral.
     */

    func centralManager(_ central: CBCentralManager, didFailToConnect peripheral:
        ↪ CBPeripheral, error: Error?) {
        if error != nil {
            print("Failed to connect to peripheral")
            return
        }
    }

    func disconnectAllConnection() {
        centralManager.cancelPeripheralConnection(blePeripheral!)
    }

    /*
     Invoked when you discover the peripherals available services.
     This method is invoked when your app calls the discoverServices(_:) method. If the
         ↪ services of the peripheral are successfully discovered, you can access them
         ↪ through the peripherals services property. If successful, the error parameter
         ↪ is nil. If unsuccessful, the error parameter returns the cause of the failure.
     */
    func peripheral(_ peripheral: CBPeripheral, didDiscoverServices error: Error?) {
        print("*******************************************************")

        if ((error) != nil) {
            print("Error discovering services: \(error!.localizedDescription)")
            return
```

```
    }

    guard let services = peripheral.services else {
        return
    }
    //We need to discover the all characteristic
    for service in services {

        peripheral.discoverCharacteristics(nil, for: service)
        // bleService = service
    }
    print("Discovered Services: \(services)")
}

/*
 Invoked when you discover the characteristics of a specified service.
 This method is invoked when your app calls the discoverCharacteristics(_:for:)
     ↪ method. If the characteristics of the specified service are successfully
     ↪ discovered, you can access them through the service's characteristics property
     ↪ . If successful, the error parameter is nil. If unsuccessful, the error
     ↪ parameter returns the cause of the failure.
 */

func peripheral(_ peripheral: CBPeripheral, didDiscoverCharacteristicsFor service:
    ↪ CBService, error: Error?) {

    print("*******************************************************")

    if ((error) != nil) {
        print("Error discovering services: \(error!.localizedDescription)")
        return
    }

    guard let characteristics = service.characteristics else {
        return
    }

    print("Found \(characteristics.count) characteristics!")

    for characteristic in characteristics {
        //looks for the right characteristic

        if characteristic.uuid.isEqual(BLE_Characteristic_uuid_Rx) {
            rxCharacteristic = characteristic

            //Once found, subscribe to the this particular characteristic...
            peripheral.setNotifyValue(true, for: rxCharacteristic!)
            // We can return after calling CBPeripheral.setNotifyValue because
                ↪ CBPeripheralDelegate's
            // didUpdateNotificationStateForCharacteristic method will be called
                ↪ automatically

            // I TOOK OUT THE FOLLOWING LINE, I DON'T KNOW IF THAT'S RIGHT THO
```

```
// peripheral.readValue(for: characteristic)



            print("Rx Characteristic: \(characteristic.uuid)")
        }
        if characteristic.uuid.isEqual(BLE_Characteristic_uuid_Tx){
            txCharacteristic = characteristic
            print("Tx Characteristic: \(characteristic.uuid)")
        }
        peripheral.discoverDescriptors(for: characteristic)
    }
}

// Getting Values From Characteristic

/*After you've found a characteristic of a service that you are interested in, you
    ↪ can read the characteristic's value by calling the peripheral "
    ↪ readValueForCharacteristic" method within the "didDiscoverCharacteristicsFor
    ↪ service" delegate.
 */
func peripheral(_ peripheral: CBPeripheral, didUpdateValueFor characteristic:
    ↪ CBCharacteristic, error: Error?) {

    if characteristic == rxCharacteristic {
        if let ASCIIstring = NSString(data: characteristic.value!, encoding: String.
            ↪ Encoding.utf8.rawValue) {
            characteristicASCIIValue = ASCIIstring
            print("Value Recieved: \((characteristicASCIIValue as String))")
            NotificationCenter.default.post(name:NSNotification.Name(rawValue: "Notify
                ↪ "), object: nil)

        }
    }
}


func peripheral(_ peripheral: CBPeripheral, didDiscoverDescriptorsFor characteristic:
    ↪  CBCharacteristic, error: Error?) {
    print("*******************************************************")

    if error != nil {
        print("\(error.debugDescription)")
        return
    }
    if ((characteristic.descriptors) != nil) {

        for x in characteristic.descriptors!{
            let descript = x as CBDescriptor!
            print("function name: DidDiscoverDescriptorForChar \(String(describing:
                ↪ descript?.description))")
            print("Rx Value \(String(describing: rxCharacteristic?.value))")
            print("Tx Value \(String(describing: txCharacteristic?.value))")
        }
```

```
    }
}


func peripheral(_ peripheral: CBPeripheral, didUpdateNotificationStateFor
    ↪ characteristic: CBCharacteristic, error: Error?) {
    print("*******************************************************")

    if (error != nil) {
        print("Error changing notification state:\(String(describing: error?.
            ↪ localizedDescription))")

    } else {
        print("Characteristic's value subscribed")
    }

    if (characteristic.isNotifying) {
        print ("Subscribed. Notification has begun for: \(characteristic.uuid)")
    }
}



func centralManager(_ central: CBCentralManager, didDisconnectPeripheral peripheral:
    ↪ CBPeripheral, error: Error?) {
    print("Disconnected")
}


func peripheral(_ peripheral: CBPeripheral, didWriteValueFor characteristic:
    ↪ CBCharacteristic, error: Error?) {
    guard error == nil else {
        print("Error discovering services: error")
        return
    }
    print("Message sent")
}

func peripheral(_ peripheral: CBPeripheral, didWriteValueFor descriptor: CBDescriptor
    ↪ , error: Error?) {
    guard error == nil else {
        print("Error discovering services: error")
        return
    }
    print("Succeeded!")
}

//Table View Functions
func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
    return self.peripherals.count
}

func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) ->
    ↪ UITableViewCell {
```

```
    //Connect to device where the peripheral is connected
    let cell = tableView.dequeueReusableCell(withIdentifier: "BlueCell") as!
        ↪ PeripheralTableViewCell
    let peripheral = self.peripherals[indexPath.row]
    let RSSI = self.RSSIs[indexPath.row]


    if peripheral.name == nil {
        cell.peripheralLabel.text = "nil"
    } else {
        cell.peripheralLabel.text = peripheral.name
    }
    cell.rssiLabel.text = "RSSI: \(RSSI)"

    return cell
}

func tableView(_ tableView: UITableView, didSelectRowAt indexPath: IndexPath) {
    blePeripheral = peripherals[indexPath.row]
    connectToDevice()
}

/*
 Invoked when the central managers state is updated.
 This is where we kick off the scan if Bluetooth is turned on.
 */
func centralManagerDidUpdateState(_ central: CBCentralManager) {
    if central.state == CBManagerState.poweredOn {
        // We will just handle it the easy way here: if Bluetooth is on, proceed...
            ↪ start scan!
        print("Bluetooth Enabled")
        startScan()

    } else {
        //If Bluetooth is off, display a UI alert message saying "Bluetooth is not
            ↪ enable" and "Make sure that your bluetooth is turned on"
        print("Bluetooth Disabled- Make sure your Bluetooth is turned on")

        let alertVC = UIAlertController(title: "Bluetooth is not enabled", message: "
            ↪ Make sure that your bluetooth is turned on", preferredStyle:
            ↪ UIAlertControllerStyle.alert)
        let action = UIAlertAction(title: "ok", style: UIAlertActionStyle.default,
            ↪ handler: { (action: UIAlertAction) -> Void in
            self.dismiss(animated: true, completion: nil)
        })
        alertVC.addAction(action)
        self.present(alertVC, animated: true, completion: nil)
    }
}

// Functions for animating transition to "Schemes" tab programatically
func tabBarController(_ tabBarController: UITabBarController, shouldSelect
    ↪ viewController: UIViewController) -> Bool {
    animateTabBarChange(tabBarController: tabBarController, to: viewController)
```

```
        return true
    }

    func animateTabBarChange(tabBarController: UITabBarController, to viewController:
        ↪ UIViewController) {
        let fromView: UIView = tabBarController.selectedViewController!.view
        let toView: UIView = viewController.view

        // do whatever animation you like
        UIView.transition(from: fromView, to: toView, duration: 0.75, options: [.
            ↪ transitionFlipFromRight], completion: nil)

    }

}
```

C.2.3 PeripheralTableViewCell.swift

```
//
// PeripheralTableViewCell.swift
// uTune
//
// Created by Kevin Schneier on 3/11/19.
//

import UIKit

class PeripheralTableViewCell: UITableViewCell {

    @IBOutlet weak var peripheralLabel: UILabel!
    @IBOutlet weak var rssiLabel: UILabel!

    override func awakeFromNib() {
        super.awakeFromNib()
        // Initialization code
    }

    override func setSelected(_ selected: Bool, animated: Bool) {
        super.setSelected(selected, animated: animated)

        // Configure the view for the selected state
    }

}
```

C.2.4 SchemeTableViewController.swift

```
//
// SchemeTableViewController.swift
// uTune
//
// Created by Kevin Schneier on 3/11/19.
//

import UIKit
import os.log

class SchemeTableViewController: UITableViewController {

    //MARK: Properties
    var schemes = [Scheme]()

    override func viewDidLoad() {
        super.viewDidLoad()

        // Use the edit button item provided by the table view controller
        navigationItem.leftBarButtonItem = editButtonItem

        // Load any saved schemes, otherwise load sample data.
        if let savedSchemes = loadSchemes() {
            schemes += savedSchemes
        }
        else {
            // Load the sample data
            loadSampleSchemes()
        }

    }

// override func viewDidAppear(_ animated: true) {
//
// }

    // MARK: - Table view data source

    override func numberOfSections(in tableView: UITableView) -> Int {
        return 1
    }

    override func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int)
        ↪  -> Int {
        return schemes.count
    }

    override func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath)
        ↪ -> UITableViewCell {

        let cellIdentifier = "SchemeTableViewCell"
```

```
        guard let cell = tableView.dequeueReusableCell(withIdentifier: cellIdentifier, for
            ↪ : indexPath) as? SchemeTableViewCell else {
            fatalError("The dequeued cell is not an instance of SchemeTableViewCell.")
        }


        let scheme = schemes[indexPath.row]
        cell.schemeLabel.text = scheme.name

        return cell
    }


    // Override to support conditional editing of the table view.
    override func tableView(_ tableView: UITableView, canEditRowAt indexPath: IndexPath)
        ↪ -> Bool {
        // Return false if you do not want the specified item to be editable.
        return true
    }

    // Override to support editing the table view.
    override func tableView(_ tableView: UITableView, commit editingStyle:
        ↪ UITableViewCell.EditingStyle, forRowAt indexPath: IndexPath) {
        if editingStyle == .delete {
            // Delete the row from the data source
            schemes.remove(at: indexPath.row)
            saveSchemes()
            tableView.deleteRows(at: [indexPath], with: .fade)
        } else if editingStyle == .insert {
            // Create a new instance of the appropriate class, insert it into the array,
                ↪ and add a new row to the table view
        }
    }

    /*
     // Override to support rearranging the table view.
     override func tableView(_ tableView: UITableView, moveRowAt fromIndexPath: IndexPath
         ↪ , to: IndexPath) {

     }
     */

    /*
     // Override to support conditional rearranging of the table view.
     override func tableView(_ tableView: UITableView, canMoveRowAt indexPath: IndexPath)
         ↪  -> Bool {
     // Return false if you do not want the item to be re-orderable.
     return true
     }
     */

    // MARK: - Navigation

    // In a storyboard-based application, you will often want to do a little preparation
```

```
    ↪ before navigation
override func prepare(for segue: UIStoryboardSegue, sender: Any?) {

    super.prepare(for: segue, sender: sender)

    switch(segue.identifier ?? "") {

    case "AddItem":
        os_log("Adding a new scheme.", log: OSLog.default, type: .debug)

    case "ShowDetail":
        guard let schemeDetailViewController = segue.destination as?
            ↪ SchemeViewController else {
            fatalError("Unexpected destination: \(segue.destination)")
        }

        guard let selectedSchemeCell = sender as? SchemeTableViewCell else {
            fatalError("Unexpected sender: \(sender)")
        }

        guard let indexPath = tableView.indexPath(for: selectedSchemeCell) else {
            fatalError("The selected cell is not being displayed by the table")
        }

        let selectedScheme = schemes[indexPath.row]
        schemeDetailViewController.scheme = selectedScheme

        // THIS WAS PUT IN FOR THE CANCEL BUTTON
        schemeDetailViewController.segueUsed = "ShowDetail"


    default:
        fatalError("Unexpected Segue Identifier; \(segue.identifier)")
    }
}


//MARK: Actions
@IBAction func unwindToSchemeList(sender: UIStoryboardSegue) {

    if let sourceViewController = sender.source as? SchemeViewController, let scheme =
        ↪  sourceViewController.scheme {

        if let selectedIndexPath = tableView.indexPathForSelectedRow {

            // Update an existing scheme.
            schemes[selectedIndexPath.row] = scheme
            tableView.reloadRows(at: [selectedIndexPath], with: .none)
        }
        else {
            // Add a new scheme.
            let newIndexPath = IndexPath(row: schemes.count, section: 0)

            schemes.append(scheme)
```

```
                tableView.insertRows(at: [newIndexPath], with: .automatic)
            }

            // Save the schemes.
            saveSchemes()

        }
    }

    //MARK: Private Methods
    private func loadSampleSchemes() {
        guard let scheme1 = Scheme(name: "EADGBE", frequencies: [82.41, 110.0, 146.8,
            ↪ 196.0, 246.9, 329.6]) else {
            fatalError("Unable to instantiate scheme1")
        }

        guard let scheme2 = Scheme(name: "DADGAD", frequencies: [73.42, 110.0, 146.8,
            ↪ 196.0, 220.0, 293.7]) else {
            fatalError("Unable to instantiate scheme2")
        }

        schemes += [scheme1, scheme2]
    }
    private func saveSchemes() {
        // +archivedDataWithRootObject:requiringSecureCoding:error:

        let isSuccessfulSave = NSKeyedArchiver.archiveRootObject(schemes, toFile: Scheme.
            ↪ ArchiveURL.path)


        if isSuccessfulSave {
            os_log("Schemes successfully saved.", log: OSLog.default, type: .debug)
        } else {
            os_log("Failed to save meals...", log: OSLog.default, type: .error)
        }

    }

    private func loadSchemes() -> [Scheme]? {
        return NSKeyedUnarchiver.unarchiveObject(withFile: Scheme.ArchiveURL.path) as? [
            ↪ Scheme]
    }


}
```

C.2.5 TuningViewController.swift

```
//
// TuningViewController.swift
// uTune
//
// Created by Kevin Schneier on 3/18/19.
//

import UIKit
import os.log
import CoreBluetooth

class TuningViewController: UIViewController, CBPeripheralManagerDelegate {

    @IBOutlet weak var doneButton: UIButton!



    override func viewDidLoad() {
        super.viewDidLoad()

        // Do any additional setup after loading the view.
    }

    /*
    // MARK: - Navigation

    // In a storyboard-based application, you will often want to do a little preparation
        ↪ before navigation
    override func prepare(for segue: UIStoryboardSegue, sender: Any?) {
        // Get the new view controller using segue.destination.
        // Pass the selected object to the new view controller.
    }
    */

    // MARK: - Actions
    @IBAction func pressDoneButton(_ sender: Any) {
        for _ in 0...5 {
            writeValue(data: "0.000" + " ")
        }

        backTwo()
    }

    // MARK: - Private Methods

    func writeValue(data: String){
        let valueString = (data as NSString).data(using: String.Encoding.utf8.rawValue)
        //change the "data" to valueString
        if let blePeripheral = blePeripheral{
            if let txCharacteristic = txCharacteristic {
                blePeripheral.writeValue(valueString!, for: txCharacteristic, type:
                    ↪ CBCharacteristicWriteType.withResponse)
```

```
            }
        }
    }

    func backTwo() {
        let viewControllers: [UIViewController] = self.navigationController!.
            ↪ viewControllers as [UIViewController]
        self.navigationController!.popToViewController(viewControllers[viewControllers.
            ↪ count - 3], animated: true)
    }

// func backOne() {
// let viewControllers: [UIViewController] = self.navigationController!.viewControllers
    ↪ as [UIViewController]
// self.navigationController!.popToViewController(viewControllers[viewControllers.count -
    ↪  2], animated: true)
// }

    func peripheralManagerDidUpdateState(_ peripheral: CBPeripheralManager) {
        if peripheral.state == .poweredOn {
            return
        }
        print("Peripheral Manager is running")
    }

}
```

C.2.6 SchemeTableViewCell.swift

```
//
// SchemeTableViewCell.swift
// uTune
//
// Created by Kevin Schneier on 3/11/19.
//

import UIKit

class SchemeTableViewCell: UITableViewCell {

    //MARK: Properties
    @IBOutlet weak var schemeLabel: UILabel!


    override func awakeFromNib() {
        super.awakeFromNib()
        // Initialization code
    }

    override func setSelected(_ selected: Bool, animated: Bool) {
        super.setSelected(selected, animated: animated)

        // Configure the view for the selected state
    }

}
```

C.2.7 RotateViewController.swift

```
//
// RotateViewController.swift
// uTune
//
// Created by Kevin Schneier on 3/11/19.
//

import UIKit

class RotateViewController: UIViewController {

    //MARK: Properties
    @IBOutlet weak var imageView: UIImageView!

    override func viewDidLoad() {
        super.viewDidLoad()

        // Do any additional setup after loading the view.
        UIView.animate(withDuration: 2.0, animations: {
            self.imageView.transform = CGAffineTransform(rotationAngle: .pi)
        }, completion: {finished in self.performSegue(withIdentifier: "BeginStart", sender
            ↪ : self)})

    }

// func moveToNext(){
// self.performSegue(withIdentifier: "BeginStart", sender: self)
// }

// override func viewDidAppear(_ animated: Bool) {
//
// self.performSegue(withIdentifier: "BeginStart", sender: self)
// }

// override func viewWillDisappear(_ animated: Bool) {
// print("got here")
// self.performSegue(withIdentifier: "BeginStart", sender: self)
// }


    /*
    // MARK: - Navigation

    // In a storyboard-based application, you will often want to do a little preparation
        ↪ before navigation
    override func prepare(for segue: UIStoryboardSegue, sender: Any?) {
        // Get the new view controller using segue.destination.
        // Pass the selected object to the new view controller.
    }
    */
}
```

C.2.8 SchemeViewController.swift

```
//
// SchemeViewController.swift
// uTune
//
// Created by Kevin Schneier on 3/11/19.
//

import UIKit
import os.log
import CoreBluetooth


class SchemeViewController: UIViewController, UITextFieldDelegate,
    ↪ UINavigationControllerDelegate, CBPeripheralManagerDelegate, UIPickerViewDelegate,
    ↪  UIPickerViewDataSource {

    //MARK: Properties
    @IBOutlet weak var schemeTextField: UITextField!
    @IBOutlet weak var saveButton: UIBarButtonItem!
    @IBOutlet weak var tuneButton: UIButton!

    // Scheme variable
    var scheme: Scheme?

    // Frequency vector variable to be put in scheme (start with default values for
        ↪ EADGBE)
    var frequencies: [Double] = [82.41, 110.0, 146.8, 196.0, 246.9, 329.6]

    // THIS WAS PUT IN FOR THE CANCEL BUTTON
    var segueUsed: String?

    // Bluetooth stuff
    var peripheralManager: CBPeripheralManager?
    var peripheral: CBPeripheral!
    var connectionStatus = false

    // Picker View stuff
    var schemePicker: UIPickerView!
    var pickerData = [["E", "D\u{266F}", "D", "C\u{266F}", "C", "B"], ["A", "G\u{266F}",
        ↪ "G", "F\u{266F}", "F"], ["D", "C\u{266F}", "C", "B", "A\u{266F}"], ["G", "F\u
        ↪ {266F}", "F", "E", "D\u{266F}"], ["B", "A\u{266F}", "A", "G\u{266F}"], ["E", "D
        ↪ \u{266F}", "D", "C\u{266F}", "C"]]
    var totalFrequencies = [[82.41, 77.78, 73.42, 69.30, 65.41, 61.74], [110.0, 103.8,
        ↪ 98.01, 92.51, 87.31], [146.8, 138.6, 130.8, 123.5, 116.5], [196.0, 185.0,
        ↪ 174.6, 164.8, 155.6], [246.9, 233.1, 220.0, 207.7], [329.6, 311.1, 293.7,
        ↪ 277.2, 261.6]]



    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view, typically from a nib.
```

```
        // Handle the text field's user input through delegate callbacks
        schemeTextField.delegate = self

        // Set up views if editing an existing scheme.
        if let scheme = scheme {
            navigationItem.title = scheme.name
            schemeTextField.text = scheme.name
            frequencies = scheme.frequencies
        }

        // Update Connection Status
        if blePeripheral?.state == CBPeripheralState.connected {
            connectionStatus = true
        }

        // Update save and tune buttons
        updateSaveButtonState()
        updateTuneButtonState()

        // Create and start the peripheral manager
        peripheralManager = CBPeripheralManager(delegate: self, queue: nil)
    }

    override func viewDidAppear(_ animated: Bool) {
        // Update Connection Status
        if blePeripheral?.state == CBPeripheralState.connected {
            connectionStatus = true
        }
        else {
            connectionStatus = false
        }
        // Update tune button
        updateTuneButtonState()

    }

    override func viewWillDisappear(_ animated: Bool) {
        super.viewWillDisappear(animated)

        // THIS IS OPTIONAL
        // Remove this view controller from the stack
// if segueUsed == "ShowDetail" {
// if let nav = self.navigationController {
// let vcs = nav.viewControllers.filter {(vc) in
// return (vc as? SchemeViewController) == nil
// }
//
// self.navigationController?.setViewControllers(vcs, animated: false)
// }
// }
    }
```

```
func createPicker(_ textField : UITextField){

    // Create picker at bottom of screen
    self.schemePicker = UIPickerView(frame:CGRect(x: 0, y: 0, width: self.view.frame.
        ↪ size.width, height: 350))
    self.schemePicker.delegate = self
    self.schemePicker.dataSource = self
    self.schemePicker.backgroundColor = UIColor.white
    schemeTextField.inputView = self.schemePicker

    // Create ToolBar to go above picker
    let toolBar = UIToolbar()
    toolBar.barStyle = .default
    toolBar.isTranslucent = true
    toolBar.sizeToFit()

    // Add buttons to the toolbar
    let doneButton = UIBarButtonItem(title: "Done", style: .plain, target: self,
        ↪ action: #selector(SchemeViewController.doneClick))
    let spaceButton = UIBarButtonItem(barButtonSystemItem: .flexibleSpace, target: nil
        ↪ , action: nil)
    toolBar.setItems([spaceButton, doneButton], animated: false)
    toolBar.isUserInteractionEnabled = true
    textField.inputAccessoryView = toolBar

    // Preselect the correct rows
    for i in 0...frequencies.count-1 {
        let correctRow = totalFrequencies[i].index(of: frequencies[i])
        schemePicker.selectRow(correctRow!, inComponent: i, animated: false)
    }
}

@objc func doneClick() {
    var string = ""
    for index in 0...frequencies.count-1 {
        string = string + pickerData[index][schemePicker.selectedRow(inComponent:
            ↪ index)]
        frequencies[index] = totalFrequencies[index][schemePicker.selectedRow(
            ↪ inComponent: index)]
    }
    schemeTextField.text = string
    schemeTextField.resignFirstResponder()
}

func textFieldDidBeginEditing(_ textField: UITextField) {
    // Disable the Save button while editing.
    saveButton.isEnabled = false
    tuneButton.isHidden = true
    createPicker(schemeTextField)
}

func textFieldDidEndEditing(_ textField: UITextField) {
    updateSaveButtonState()
    updateTuneButtonState()
```

```
    navigationItem.title = textField.text
}

//MARK: Picker
func numberOfComponents(in pickerView: UIPickerView) -> Int {
    return pickerData.count
}

func pickerView(_ pickerView: UIPickerView, numberOfRowsInComponent component: Int)
    ↪ -> Int {
    return pickerData[component].count
}

func pickerView(_ pickerView: UIPickerView, titleForRow row: Int, forComponent
    ↪ component: Int) -> String? {
    return pickerData[component][row]
}

func pickerView(_ pickerView: UIPickerView, didSelectRow row: Int, inComponent
    ↪ component: Int) {
    // This method is triggered whenever the user makes a change to the picker
        ↪ selection.
    // The parameter named row and component represents what was selected.
}

func pickerView(_ pickerView: UIPickerView, viewForRow row: Int, forComponent
    ↪ component: Int, reusing view: UIView?) -> UIView {
    var label = UILabel()
    if let v = view {
        label = v as! UILabel
    }
    label.font = UIFont (name: "Helvetica Neue", size: 16)
    label.text = pickerData[component][row]
    label.textAlignment = .center
    return label
}


//MARK: Navigation
@IBAction func cancel(_ sender: UIBarButtonItem) {
    // Depending on style of presentation (modal or push presentation), this view
        ↪ controller needs to be dismissed in two different ways.

    // I CHANGED SOME STUFF HERE TO MAKE THE CANCEL BUTTON WORK
    if segueUsed == "ShowDetail" {
        navigationController?.popViewController(animated: true)
    }
    else {
        dismiss(animated: true, completion: nil)
    }
}
```

```
    override func prepare(for segue: UIStoryboardSegue, sender: Any?){

        super.prepare(for:segue, sender:sender)

        // Configure the destination view controller only when the save button is pressed.
        guard let button = sender as? UIBarButtonItem, button === saveButton else {
            os_log("The save button was not pressed, cancelling", log: OSLog.default, type
                ↪ : .debug)
            return
        }

        let name = schemeTextField.text ?? ""

        scheme = Scheme(name: name, frequencies: frequencies)

    }


    //MARK: Actions
    @IBAction func pressTuneButton(_ sender: Any) {
        for index in 0...frequencies.count-1 {
            writeValue(data: frequencies[index].description + " ")
        }
// writeValue(data: "\r\n")
    }



    //MARK: Private Methods
    private func updateSaveButtonState() {
        // Disable the Save button if the text field is empty.
        let text = schemeTextField.text ?? ""
        saveButton.isEnabled = !text.isEmpty
    }

    private func updateTuneButtonState() {
        // Disable the tune button if the text field is empty
        let text = schemeTextField.text ?? ""
        tuneButton.isHidden = text.isEmpty || !connectionStatus
    }


    // Write functions
    func writeValue(data: String){
        let valueString = (data as NSString).data(using: String.Encoding.utf8.rawValue)
        //change the "data" to valueString
        if let blePeripheral = blePeripheral{
            if let txCharacteristic = txCharacteristic {
                blePeripheral.writeValue(valueString!, for: txCharacteristic, type:
                    ↪ CBCharacteristicWriteType.withResponse)
            }
        }
    }
```

```
    func peripheralManagerDidUpdateState(_ peripheral: CBPeripheralManager) {
        if peripheral.state == .poweredOn {
            return
        }
        print("Peripheral Manager is running")
    }

}
```

C.2.9 Scheme.swift

```
//
// Scheme.swift
// uTune
//
// Created by Kevin Schneier on 3/11/19.
//

import UIKit
import os.log

class Scheme: NSObject, NSCoding {

    // MARK: Properties

    var name: String
    var frequencies: [Double]


    // MARK: Archiving Paths
    static let DocumentsDirectory = FileManager().urls(for: .documentDirectory, in: .
        ↪ userDomainMask).first!
    static let ArchiveURL = DocumentsDirectory.appendingPathComponent("schemes")


    // MARK: Types

    struct PropertyKey {
        static let name = "name"
        static let frequencies = "frequencies"
    }

    // MARK: Initialization

// init?(name: String, s1: Double, s2: Double, s3: Double, s4: Double, s5: Double, s6:
    ↪ Double){
    init?(name: String, frequencies: [Double]){
        guard !name.isEmpty else {
            return nil
        }

        self.name = name
        self.frequencies = frequencies
    }


    // MARK: NSCoding
    func encode(with aCoder: NSCoder) {
        aCoder.encode(name, forKey: PropertyKey.name)
        aCoder.encode(frequencies, forKey: PropertyKey.frequencies)
    }

    required convenience init?(coder aDecoder: NSCoder) {
```

```
        // The name is required. If we cannot decode a name string, the initializer should
            ↪ fail.
        guard let name = aDecoder.decodeObject(forKey: PropertyKey.name) as? String else {
            os_log("Unable to decode the name for a Scheme object.", log: OSLog.default,
                ↪ type: .debug)
            return nil
        }

        guard let frequencies = aDecoder.decodeObject(forKey: PropertyKey.frequencies) as?
            ↪ [Double] else {
            os_log("Unable to decode the frequencies for a Scheme object.", log: OSLog.
                ↪ default, type: .debug)
            return nil
        }

        self.init(name: name, frequencies: frequencies)
    }
}
```

C.2.10 UUIDKey.swift

```
//
// UUIDKey.swift
// uTune
//
// Created by Kevin Schneier on 3/11/19.
//

import CoreBluetooth


//Uart Service uuid

// UNUSED
//let kBLEService_UUID = "49535343-4C8A-39B3-2F49-511CFF073B7E"

// MINE - for some reason it doesn't work when I do this
let kBLEService_UUID = "49535343-FE7D-4AE5-8FA9-9FAFD205E455"
let kBLE_Characteristic_uuid_Tx = "49535343-1E4D-4BD9-BA61-23C647249616"
let kBLE_Characteristic_uuid_Rx = "49535343-8841-43F4-A8D4-ECBE34729BB3"

let kAdvert_UUID = "FFF1" // This is just set myself on the PC


let Advert_UUID = CBUUID(string: kAdvert_UUID) // Make it the correct format
let BLEService_UUID = CBUUID(string: kBLEService_UUID)
let BLE_Characteristic_uuid_Tx = CBUUID(string: kBLE_Characteristic_uuid_Tx)//(Property =
    ↪  Write without response)
let BLE_Characteristic_uuid_Rx = CBUUID(string: kBLE_Characteristic_uuid_Rx)// (Property
    ↪ = Read/Notify)

let MaxCharacters = 20
```

C.2.11 Main.storyboard

```
<?xml version="1.0" encoding="UTF-8"?>
<document type="com.apple.InterfaceBuilder3.CocoaTouch.Storyboard.XIB" version="3.0"
    ↪ toolsVersion="14490.70" targetRuntime="iOS.CocoaTouch" propertyAccessControl="none
    ↪ " useAutolayout="YES" useTraitCollections="YES" colorMatched="YES"
    ↪ initialViewController="CYc-vb-nOR">
    <device id="retina4_7" orientation="portrait">
        <adaptation id="fullscreen"/>
    </device>
    <dependencies>
        <deployment identifier="iOS"/>
        <plugIn identifier="com.apple.InterfaceBuilder.IBCocoaTouchPlugin" version
            ↪ ="14490.49"/>
        <capability name="documents saved in the Xcode 8 format" minToolsVersion="8.0"/>
    </dependencies>
    <scenes>
        <!--Connect-->
        <scene sceneID="kgz-Em-9Pz">
            <objects>
                <navigationController id="3Yh-2X-Wy2" sceneMemberID="viewController">
                    <tabBarItem key="tabBarItem" title="Connect" image="icons8-bluetooth-
                        ↪ filled" id="sV1-AV-Eyw"/>
                    <navigationBar key="navigationBar" contentMode="scaleToFill" id="eSA-TS
                        ↪ -cY5">
                        <rect key="frame" x="0.0" y="20" width="375" height="44"/>
                        <autoresizingMask key="autoresizingMask"/>
                    </navigationBar>
                    <connections>
                        <segue destination="s6h-Zo-tC7" kind="relationship" relationship="
                            ↪ rootViewController" id="qAP-Ld-lPC"/>
                    </connections>
                </navigationController>
                <placeholder placeholderIdentifier="IBFirstResponder" id="9J8-8g-Bww"
                    ↪ userLabel="First Responder" sceneMemberID="firstResponder"/>
            </objects>
            <point key="canvasLocation" x="226" y="-16"/>
        </scene>
        <!--Devices-->
        <scene sceneID="ICt-bz-slO">
            <objects>
                <viewController storyboardIdentifier="BLECentralViewController"
                    ↪ automaticallyAdjustsScrollViewInsets="NO" id="s6h-Zo-tC7"
                    ↪ customClass="BLECentralViewController" customModule="uTune"
                    ↪ customModuleProvider="target" sceneMemberID="viewController">
                    <layoutGuides>
                        <viewControllerLayoutGuide type="top" id="LSv-z8-Zyp"/>
                        <viewControllerLayoutGuide type="bottom" id="cCu-cf-R0b"/>
                    </layoutGuides>
                    <view key="view" contentMode="scaleToFill" id="M0w-Ty-hJr">
                        <rect key="frame" x="0.0" y="0.0" width="375" height="667"/>
                        <autoresizingMask key="autoresizingMask" widthSizable="YES"
                            ↪ heightSizable="YES"/>
                        <subviews>
```

```
<tableView clipsSubviews="YES" contentMode="scaleToFill"
    ↪ alwaysBounceVertical="YES" dataMode="prototypes" style="
    ↪ plain" separatorStyle="default" rowHeight="98"
    ↪ sectionHeaderHeight="28" sectionFooterHeight="28"
    ↪ translatesAutoresizingMaskIntoConstraints="NO" id="iJd-RF
    ↪ -aiZ">
    <rect key="frame" x="0.0" y="64" width="375" height="554"/>
    <color key="backgroundColor" white="1" alpha="1" colorSpace
        ↪ ="calibratedWhite"/>
    <prototypes>
        <tableViewCell clipsSubviews="YES" contentMode="
            ↪ scaleToFill" selectionStyle="default"
            ↪ indentationWidth="10" reuseIdentifier="BlueCell"
            ↪ rowHeight="98" id="LJh-m2-NaE" customClass="
            ↪ PeripheralTableViewCell" customModule="uTune"
            ↪ customModuleProvider="target">
            <rect key="frame" x="0.0" y="28" width="375" height
                ↪ ="98"/>
            <autoresizingMask key="autoresizingMask"/>
            <tableViewCellContentView key="contentView" opaque="
                ↪ NO" clipsSubviews="YES" multipleTouchEnabled="
                ↪ YES" contentMode="center" tableViewCell="LJh-
                ↪ m2-NaE" id="AK0-A7-RVY">
                <rect key="frame" x="0.0" y="0.0" width="375"
                    ↪ height="97.5"/>
                <autoresizingMask key="autoresizingMask"/>
                <subviews>
                    <label opaque="NO" userInteractionEnabled="NO"
                        ↪  contentMode="left"
                        ↪ horizontalHuggingPriority="251"
                        ↪ verticalHuggingPriority="251"
                        ↪ fixedFrame="YES" text="RSSI"
                        ↪ textAlignment="natural" lineBreakMode="
                        ↪ tailTruncation" baselineAdjustment="
                        ↪ alignBaselines" adjustsFontSizeToFit="
                        ↪ NO"
                        ↪ translatesAutoresizingMaskIntoConstraints
                        ↪ ="NO" id="0wW-qX-zLz">
                        <rect key="frame" x="14" y="52" width
                            ↪ ="120" height="21"/>
                        <autoresizingMask key="autoresizingMask"
                            ↪ flexibleMaxX="YES" flexibleMaxY="
                            ↪ YES"/>
                        <fontDescription key="fontDescription"
                            ↪ type="system" pointSize="17"/>
                        <color key="textColor" red="0.0" green
                            ↪ ="0.50196081400000003" blue="1"
                            ↪ alpha="1" colorSpace="calibratedRGB
                            ↪ "/>
                        <nil key="highlightedColor"/>
                    </label>
                    <label opaque="NO" userInteractionEnabled="NO"
                        ↪  contentMode="left"
                        ↪ horizontalHuggingPriority="251"
```

```
                                    ↪ verticalHuggingPriority="251"
                                    ↪ fixedFrame="YES" text="Peripheral Label
                                    ↪ " textAlignment="natural" lineBreakMode
                                    ↪ ="tailTruncation" baselineAdjustment="
                                    ↪ alignBaselines" adjustsFontSizeToFit="
                                    ↪ NO"
                                    ↪ translatesAutoresizingMaskIntoConstraints
                                    ↪ ="NO" id="0wd-Uc-DEI">
                                    <rect key="frame" x="14" y="23" width
                                        ↪ ="291" height="21"/>
                                    <autoresizingMask key="autoresizingMask"
                                        ↪ flexibleMaxX="YES" flexibleMaxY="
                                        ↪ YES"/>
                                    <fontDescription key="fontDescription"
                                        ↪ type="system" pointSize="17"/>
                                    <color key="textColor" red="0.0" green
                                        ↪ ="0.50196081400000003" blue="1"
                                        ↪ alpha="1" colorSpace="calibratedRGB
                                        ↪ "/>
                                    <nil key="highlightedColor"/>
                                </label>
                            </subviews>
                        </tableViewCellContentView>
                        <connections>
                            <outlet property="peripheralLabel" destination="0
                                ↪ wd-Uc-DEI" id="NrB-Zn-JsO"/>
                            <outlet property="rssiLabel" destination="0wW-qX-
                                ↪ zLz" id="R3R-b1-YlY"/>
                        </connections>
                    </tableViewCell>
                </prototypes>
                <connections>
                    <outlet property="dataSource" destination="s6h-Zo-tC7" id
                        ↪ ="gRn-Hc-yTZ"/>
                    <outlet property="delegate" destination="s6h-Zo-tC7" id="
                        ↪ sD9-kA-Vcs"/>
                </connections>
            </tableView>
        </subviews>
        <color key="backgroundColor" white="1" alpha="1" colorSpace="
            ↪ calibratedWhite"/>
        <constraints>
            <constraint firstItem="cCu-cf-R0b" firstAttribute="top"
                ↪ secondItem="iJd-RF-aiZ" secondAttribute="bottom" id="5Ou-
                ↪ z8-zmL"/>
            <constraint firstItem="iJd-RF-aiZ" firstAttribute="leading"
                ↪ secondItem="M0w-Ty-hJr" secondAttribute="leading" id="S69
                ↪ -FI-myd"/>
            <constraint firstItem="iJd-RF-aiZ" firstAttribute="top"
                ↪ secondItem="LSv-z8-Zyp" secondAttribute="bottom" id="bi9-
                ↪ Wb-uIJ"/>
            <constraint firstAttribute="trailing" secondItem="iJd-RF-aiZ"
                ↪ secondAttribute="trailing" id="vEz-UJ-4BT"/>
        </constraints>
```

```
                </view>
                <navigationItem key="navigationItem" title="Devices" id="w25-yy-fKD">
                    <barButtonItem key="rightBarButtonItem" systemItem="refresh" id="
                        ↪ wI8-q6-jjP">
                        <connections>
                            <action selector="refreshAction:" destination="s6h-Zo-tC7"
                                ↪ id="0uP-Tx-3ga"/>
                        </connections>
                    </barButtonItem>
                </navigationItem>
                <connections>
                    <outlet property="baseTableView" destination="iJd-RF-aiZ" id="Bqn-
                        ↪ an-Cdy"/>
                    <outlet property="refreshButton" destination="wI8-q6-jjP" id="Cid-
                        ↪ ow-d87"/>
                </connections>
            </viewController>
            <placeholder placeholderIdentifier="IBFirstResponder" id="E6E-yO-YNY"
                ↪ userLabel="First Responder" sceneMemberID="firstResponder"/>
        </objects>
        <point key="canvasLocation" x="1023" y="-16"/>
    </scene>
    <!--Schemes-->
    <scene sceneID="BW6-FP-ygr">
        <objects>
            <tableViewController storyboardIdentifier="SchemeTableViewController" id="9
                ↪ zg-6f-9QW" customClass="SchemeTableViewController" customModule="
                ↪ uTune" customModuleProvider="target" sceneMemberID="viewController">
                <tableView key="view" clipsSubviews="YES" contentMode="scaleToFill"
                    ↪ alwaysBounceVertical="YES" dataMode="prototypes" style="plain"
                    ↪ separatorStyle="default" rowHeight="-1" estimatedRowHeight="-1"
                    ↪ sectionHeaderHeight="28" sectionFooterHeight="28" id="nL5-sE-Q3N
                    ↪ ">
                    <rect key="frame" x="0.0" y="0.0" width="375" height="667"/>
                    <autoresizingMask key="autoresizingMask" widthSizable="YES"
                        ↪ heightSizable="YES"/>
                    <color key="backgroundColor" white="1" alpha="1" colorSpace="custom
                        ↪ " customColorSpace="genericGamma22GrayColorSpace"/>
                    <prototypes>
                        <tableViewCell clipsSubviews="YES" contentMode="scaleToFill"
                            ↪ preservesSuperviewLayoutMargins="YES" selectionStyle="
                            ↪ default" indentationWidth="10" reuseIdentifier="
                            ↪ SchemeTableViewCell" id="ofV-Zj-OxJ" customClass="
                            ↪ SchemeTableViewCell" customModule="uTune"
                            ↪ customModuleProvider="target">
                            <rect key="frame" x="0.0" y="28" width="375" height="44"/>
                            <autoresizingMask key="autoresizingMask"/>
                            <tableViewCellContentView key="contentView" opaque="NO"
                                ↪ clipsSubviews="YES" multipleTouchEnabled="YES"
                                ↪ contentMode="center" preservesSuperviewLayoutMargins
                                ↪ ="YES" insetsLayoutMarginsFromSafeArea="NO"
                                ↪ tableViewCell="ofV-Zj-OxJ" id="1Gb-Pg-WKO">
                                <rect key="frame" x="0.0" y="0.0" width="375" height
                                    ↪ ="43.5"/>
```

```
                            <autoresizingMask key="autoresizingMask"/>
                            <subviews>
                                <label opaque="NO" userInteractionEnabled="NO"
                                    ↪ contentMode="left" horizontalHuggingPriority
                                    ↪ ="251" verticalHuggingPriority="251"
                                    ↪ fixedFrame="YES" text="Label" textAlignment="
                                    ↪ natural" lineBreakMode="tailTruncation"
                                    ↪ baselineAdjustment="alignBaselines"
                                    ↪ adjustsFontSizeToFit="NO"
                                    ↪ translatesAutoresizingMaskIntoConstraints="NO"
                                    ↪  id="0On-vA-rVx">
                                    <rect key="frame" x="16" y="11" width="343"
                                        ↪ height="22"/>
                                    <autoresizingMask key="autoresizingMask"
                                        ↪ flexibleMaxX="YES" flexibleMaxY="YES"/>
                                    <fontDescription key="fontDescription" type="
                                        ↪ system" pointSize="17"/>
                                    <nil key="textColor"/>
                                    <nil key="highlightedColor"/>
                                </label>
                            </subviews>
                        </tableViewCellContentView>
                        <connections>
                            <outlet property="schemeLabel" destination="0On-vA-rVx"
                                ↪ id="jgd-F9-I8g"/>
                            <segue destination="WOz-3c-ybz" kind="show" identifier="
                                ↪ ShowDetail" id="ssi-AZ-uI6"/>
                        </connections>
                    </tableViewCell>
                </prototypes>
                <connections>
                    <outlet property="dataSource" destination="9zg-6f-9QW" id="y5I-
                        ↪ qg-qvh"/>
                    <outlet property="delegate" destination="9zg-6f-9QW" id="azb-Fs-
                        ↪ dyn"/>
                </connections>
            </tableView>
            <navigationItem key="navigationItem" title="Schemes" id="ris-mc-cND">
                <barButtonItem key="rightBarButtonItem" systemItem="add" id="J9X-r4
                    ↪ -2QH">
                    <connections>
                        <segue destination="22q-yv-PRZ" kind="presentation"
                            ↪ identifier="AddItem" id="NoK-3e-ESs"/>
                    </connections>
                </barButtonItem>
            </navigationItem>
        </tableViewController>
        <placeholder placeholderIdentifier="IBFirstResponder" id="x7a-ff-5KO"
            ↪ userLabel="First Responder" sceneMemberID="firstResponder"/>
    </objects>
    <point key="canvasLocation" x="-186" y="705"/>
</scene>
<!--Scheme-->
<scene sceneID="0U6-EN-Vpo">
```

```
<objects>
    <viewController id="WOz-3c-ybz" customClass="SchemeViewController"
        ↪ customModule="uTune" customModuleProvider="target" sceneMemberID="
        ↪ viewController">
        <layoutGuides>
            <viewControllerLayoutGuide type="top" id="8HF-pF-cZq"/>
            <viewControllerLayoutGuide type="bottom" id="e1K-Nq-Ys9"/>
        </layoutGuides>
        <view key="view" contentMode="scaleToFill" id="hxX-gr-Ypi">
            <rect key="frame" x="0.0" y="0.0" width="375" height="667"/>
            <autoresizingMask key="autoresizingMask" widthSizable="YES"
                ↪ heightSizable="YES"/>
            <subviews>
                <stackView opaque="NO" contentMode="scaleToFill" axis="vertical"
                    ↪  alignment="center" spacing="100"
                    ↪ translatesAutoresizingMaskIntoConstraints="NO" id="u3f-Yx
                    ↪ -36n">
                    <rect key="frame" x="16" y="84" width="343" height="166"/>
                    <subviews>
                        <textField opaque="NO" contentMode="scaleToFill"
                            ↪ contentHorizontalAlignment="left"
                            ↪ contentVerticalAlignment="center" borderStyle="
                            ↪ roundedRect" placeholder="Enter scheme (e.g.
                            ↪ DADGAD)" textAlignment="natural" minimumFontSize
                            ↪ ="17" translatesAutoresizingMaskIntoConstraints="
                            ↪ NO" id="LjT-Yw-Lbe">
                            <rect key="frame" x="0.0" y="0.0" width="343" height
                                ↪ ="30"/>
                            <nil key="textColor"/>
                            <fontDescription key="fontDescription" type="system"
                                ↪ pointSize="14"/>
                            <textInputTraits key="textInputTraits" returnKeyType
                                ↪ ="done" enablesReturnKeyAutomatically="YES"/>
                        </textField>
                        <button opaque="NO" contentMode="scaleToFill"
                            ↪ contentHorizontalAlignment="center"
                            ↪ contentVerticalAlignment="center" buttonType="
                            ↪ roundedRect" lineBreakMode="middleTruncation"
                            ↪ translatesAutoresizingMaskIntoConstraints="NO" id
                            ↪ ="FaH-je-riq">
                            <rect key="frame" x="146" y="130" width="51" height
                                ↪ ="36"/>
                            <fontDescription key="fontDescription" type="system"
                                ↪ pointSize="20"/>
                            <state key="normal" title="uTune"/>
                            <connections>
                                <action selector="pressTuneButton:" destination="
                                    ↪ WOz-3c-ybz" eventType="touchUpInside" id="
                                    ↪ oAC-SL-vsW"/>
                                <segue destination="NSw-Kq-bSf" kind="show" id="
                                    ↪ Yje-UD-Ufi"/>
                            </connections>
                        </button>
                    </subviews>
```

```
                        <constraints>
                            <constraint firstAttribute="trailing" secondItem="LjT-Yw-
                                ↪ Lbe" secondAttribute="trailing" id="zcp-eW-6fS"/>
                        </constraints>
                    </stackView>
                </subviews>
                <color key="backgroundColor" white="1" alpha="1" colorSpace="custom
                    ↪ " customColorSpace="genericGamma22GrayColorSpace"/>
                <constraints>
                    <constraint firstItem="u3f-Yx-36n" firstAttribute="leading"
                        ↪ secondItem="hxX-gr-Ypi" secondAttribute="leadingMargin"
                        ↪ id="9gD-7a-lUH"/>
                    <constraint firstAttribute="trailingMargin" secondItem="u3f-Yx
                        ↪ -36n" secondAttribute="trailing" id="Fg9-ic-A3Q"/>
                    <constraint firstItem="u3f-Yx-36n" firstAttribute="top"
                        ↪ secondItem="8HF-pF-cZq" secondAttribute="bottom" constant
                        ↪ ="20" id="bjh-bD-pmT"/>
                </constraints>
            </view>
            <navigationItem key="navigationItem" title="Scheme" id="kDP-if-U0E">
                <barButtonItem key="leftBarButtonItem" systemItem="cancel" id="n14
                    ↪ -4y-CJK">
                    <connections>
                        <action selector="cancel:" destination="WOz-3c-ybz" id="0GM-
                            ↪ HU-Nv2"/>
                    </connections>
                </barButtonItem>
                <barButtonItem key="rightBarButtonItem" systemItem="save" id="TqI-
                    ↪ Re-Dxo">
                    <connections>
                        <segue destination="pnx-1I-mFy" kind="unwind" unwindAction="
                            ↪ unwindToSchemeListWithSender:" id="tI6-r4-Ssd"/>
                    </connections>
                </barButtonItem>
            </navigationItem>
            <connections>
                <outlet property="saveButton" destination="TqI-Re-Dxo" id="tqi-6P-
                    ↪ MQA"/>
                <outlet property="schemeTextField" destination="LjT-Yw-Lbe" id="Gtm
                    ↪ -Pd-zWl"/>
                <outlet property="tuneButton" destination="FaH-je-riq" id="pvE-NB
                    ↪ -45T"/>
            </connections>
        </viewController>
        <placeholder placeholderIdentifier="IBFirstResponder" id="8k9-kK-x2W"
            ↪ userLabel="First Responder" sceneMemberID="firstResponder"/>
        <exit id="pnx-1I-mFy" userLabel="Exit" sceneMemberID="exit"/>
    </objects>
    <point key="canvasLocation" x="1577" y="705"/>
</scene>
<!--Tuning View Controller-->
<scene sceneID="6fh-cU-N8d">
    <objects>
        <viewController id="NSw-Kq-bSf" customClass="TuningViewController"
```

```
↪ customModule="uTune" customModuleProvider="target" sceneMemberID="
↪ viewController">
<layoutGuides>
    <viewControllerLayoutGuide type="top" id="RF8-h4-ReX"/>
    <viewControllerLayoutGuide type="bottom" id="Otp-80-rJl"/>
</layoutGuides>
<view key="view" contentMode="scaleToFill" id="7Cg-PF-N58">
    <rect key="frame" x="0.0" y="0.0" width="375" height="667"/>
    <autoresizingMask key="autoresizingMask" widthSizable="YES"
        ↪ heightSizable="YES"/>
    <subviews>
        <stackView opaque="NO" contentMode="scaleToFill" axis="vertical"
            ↪  alignment="center" spacing="42"
            ↪ translatesAutoresizingMaskIntoConstraints="NO" id="HNv-bq
            ↪ -gDK">
            <rect key="frame" x="16" y="214" width="343" height
                ↪ ="106.5"/>
            <subviews>
                <label opaque="NO" userInteractionEnabled="NO"
                    ↪ contentMode="left" horizontalHuggingPriority="251"
                    ↪  verticalHuggingPriority="251" text="Tuning..."
                    ↪ textAlignment="center" lineBreakMode="
                    ↪ tailTruncation" baselineAdjustment="alignBaselines
                    ↪ " adjustsFontSizeToFit="NO"
                    ↪ translatesAutoresizingMaskIntoConstraints="NO" id
                    ↪ ="yvn-eh-jfe">
                    <rect key="frame" x="130.5" y="0.0" width="82" height
                        ↪ ="26.5"/>
                    <fontDescription key="fontDescription" type="system"
                        ↪ pointSize="22"/>
                    <nil key="textColor"/>
                    <nil key="highlightedColor"/>
                </label>
                <button opaque="NO" contentMode="scaleToFill"
                    ↪ contentHorizontalAlignment="center"
                    ↪ contentVerticalAlignment="center" buttonType="
                    ↪ roundedRect" lineBreakMode="middleTruncation"
                    ↪ translatesAutoresizingMaskIntoConstraints="NO" id
                    ↪ ="5z4-aF-d6G">
                    <rect key="frame" x="147" y="68.5" width="49" height
                        ↪ ="38"/>
                    <fontDescription key="fontDescription" type="system"
                        ↪ pointSize="21"/>
                    <state key="normal" title="Done"/>
                    <connections>
                        <action selector="pressDoneButton:" destination="
                            ↪ NSw-Kq-bSf" eventType="touchUpInside" id="7
                            ↪ zH-0q-IAH"/>
                    </connections>
                </button>
            </subviews>
        </stackView>
    </subviews>
    <color key="backgroundColor" white="1" alpha="1" colorSpace="custom
```

```
                    ↪ " customColorSpace="genericGamma22GrayColorSpace"/>
                <constraints>
                    <constraint firstItem="HNv-bq-gDK" firstAttribute="leading"
                        ↪ secondItem="7Cg-PF-N58" secondAttribute="leadingMargin"
                        ↪ id="4Uq-Pv-Uf4"/>
                    <constraint firstItem="HNv-bq-gDK" firstAttribute="top"
                        ↪ secondItem="RF8-h4-ReX" secondAttribute="bottom" constant
                        ↪ ="150" id="COd-4p-RcT"/>
                    <constraint firstAttribute="trailingMargin" secondItem="HNv-bq-
                        ↪ gDK" secondAttribute="trailing" id="L4V-ge-cog"/>
                </constraints>
            </view>
            <connections>
                <outlet property="doneButton" destination="5z4-aF-d6G" id="BPc-3Q-5
                    ↪ jw"/>
            </connections>
        </viewController>
        <placeholder placeholderIdentifier="IBFirstResponder" id="qO4-I5-v7t"
            ↪ userLabel="First Responder" sceneMemberID="firstResponder"/>
    </objects>
    <point key="canvasLocation" x="2496.8000000000002" y="704.79760119940033"/>
</scene>
<!--Tab Bar Controller-->
<scene sceneID="FcG-wl-eND">
    <objects>
        <tabBarController storyboardIdentifier="TabBarController" id="tAr-lh-EcU"
            ↪ sceneMemberID="viewController">
            <tabBar key="tabBar" contentMode="scaleToFill"
                ↪ insetsLayoutMarginsFromSafeArea="NO" id="6eM-QC-jYJ">
                <rect key="frame" x="0.0" y="0.0" width="375" height="49"/>
                <autoresizingMask key="autoresizingMask"/>
                <color key="backgroundColor" white="0.0" alpha="0.0" colorSpace="
                    ↪ custom" customColorSpace="genericGamma22GrayColorSpace"/>
            </tabBar>
            <connections>
                <segue destination="3Yh-2X-Wy2" kind="relationship" relationship="
                    ↪ viewControllers" id="3Qk-Ba-zML"/>
                <segue destination="AVB-pk-wxK" kind="relationship" relationship="
                    ↪ viewControllers" id="Tpy-4Z-L8t"/>
            </connections>
        </tabBarController>
        <placeholder placeholderIdentifier="IBFirstResponder" id="gaS-Nf-ejZ"
            ↪ userLabel="First Responder" sceneMemberID="firstResponder"/>
    </objects>
    <point key="canvasLocation" x="-566" y="-16"/>
</scene>
<!--Rotate View Controller-->
<scene sceneID="wW0-UR-WbS">
    <objects>
        <viewController id="CYc-vb-nOR" customClass="RotateViewController"
            ↪ customModule="uTune" customModuleProvider="target" sceneMemberID="
            ↪ viewController">
            <layoutGuides>
                <viewControllerLayoutGuide type="top" id="s6o-zo-MqJ"/>
```

```
                <viewControllerLayoutGuide type="bottom" id="4jr-Sq-QGW"/>
            </layoutGuides>
            <view key="view" contentMode="scaleToFill" id="Rim-h3-xOU">
                <rect key="frame" x="0.0" y="0.0" width="375" height="667"/>
                <autoresizingMask key="autoresizingMask" widthSizable="YES"
                    ↪ heightSizable="YES"/>
                <subviews>
                    <stackView opaque="NO" contentMode="scaleToFill" axis="vertical"
                        ↪  translatesAutoresizingMaskIntoConstraints="NO" id="RdD-
                        ↪ rP-Muj">
                        <rect key="frame" x="100" y="220" width="175" height="175"/>
                        <subviews>
                            <imageView userInteractionEnabled="NO" contentMode="
                                ↪ scaleToFill" horizontalHuggingPriority="251"
                                ↪ verticalHuggingPriority="251" image="uTuneRotated"
                                ↪  translatesAutoresizingMaskIntoConstraints="NO" id
                                ↪ ="TNr-gj-yws">
                                <rect key="frame" x="0.0" y="0.0" width="175" height
                                    ↪ ="175"/>
                            </imageView>
                        </subviews>
                        <constraints>
                            <constraint firstAttribute="width" secondItem="RdD-rP-Muj
                                ↪ " secondAttribute="height" multiplier="1:1" id="
                                ↪ MBb-GM-7Ti"/>
                        </constraints>
                    </stackView>
                </subviews>
                <color key="backgroundColor" white="1" alpha="1" colorSpace="custom
                    ↪ " customColorSpace="genericGamma22GrayColorSpace"/>
                <constraints>
                    <constraint firstAttribute="trailing" secondItem="RdD-rP-Muj"
                        ↪ secondAttribute="trailing" constant="100" id="N28-1Z-rDF
                        ↪ "/>
                    <constraint firstItem="RdD-rP-Muj" firstAttribute="top"
                        ↪ secondItem="s6o-zo-MqJ" secondAttribute="bottom" constant
                        ↪ ="200" id="QOZ-f4-kpL"/>
                    <constraint firstItem="RdD-rP-Muj" firstAttribute="leading"
                        ↪ secondItem="Rim-h3-xOU" secondAttribute="leading"
                        ↪ constant="100" id="j29-hE-KW4"/>
                </constraints>
            </view>
            <connections>
                <outlet property="imageView" destination="TNr-gj-yws" id="1Hv-8g-
                    ↪ Rw4"/>
                <segue destination="tAr-lh-EcU" kind="presentation" identifier="
                    ↪ BeginStart" id="Ejn-5J-cbm"/>
            </connections>
        </viewController>
        <placeholder placeholderIdentifier="IBFirstResponder" id="4Cg-Cu-pMg"
            ↪ userLabel="First Responder" sceneMemberID="firstResponder"/>
    </objects>
    <point key="canvasLocation" x="-1764" y="-17"/>
</scene>
```

```
        <!--Schemes-->
        <scene sceneID="51h-8d-QaF">
            <objects>
                <navigationController automaticallyAdjustsScrollViewInsets="NO" id="AVB-pk-
                    ↪ wxK" sceneMemberID="viewController">
                    <tabBarItem key="tabBarItem" title="Schemes" image="music_library" id="
                        ↪ y73-ZQ-bvJ"/>
                    <toolbarItems/>
                    <navigationBar key="navigationBar" contentMode="scaleToFill"
                        ↪ insetsLayoutMarginsFromSafeArea="NO" id="tH8-RG-y4p">
                        <rect key="frame" x="0.0" y="20" width="375" height="44"/>
                        <autoresizingMask key="autoresizingMask"/>
                    </navigationBar>
                    <nil name="viewControllers"/>
                    <connections>
                        <segue destination="9zg-6f-9QW" kind="relationship" relationship="
                            ↪ rootViewController" id="FGZ-a8-N3b"/>
                    </connections>
                </navigationController>
                <placeholder placeholderIdentifier="IBFirstResponder" id="KTO-3j-3KM"
                    ↪ userLabel="First Responder" sceneMemberID="firstResponder"/>
            </objects>
            <point key="canvasLocation" x="-1127" y="705"/>
        </scene>
        <!--Navigation Controller-->
        <scene sceneID="tzk-7o-def">
            <objects>
                <navigationController automaticallyAdjustsScrollViewInsets="NO" id="22q-yv-
                    ↪ PRZ" sceneMemberID="viewController">
                    <toolbarItems/>
                    <navigationBar key="navigationBar" contentMode="scaleToFill"
                        ↪ insetsLayoutMarginsFromSafeArea="NO" id="bwR-Xk-qmT">
                        <rect key="frame" x="0.0" y="20" width="375" height="44"/>
                        <autoresizingMask key="autoresizingMask"/>
                    </navigationBar>
                    <nil name="viewControllers"/>
                    <connections>
                        <segue destination="WOz-3c-ybz" kind="relationship" relationship="
                            ↪ rootViewController" id="ZSQ-Mg-pRW"/>
                    </connections>
                </navigationController>
                <placeholder placeholderIdentifier="IBFirstResponder" id="ucG-ZH-5oF"
                    ↪ userLabel="First Responder" sceneMemberID="firstResponder"/>
            </objects>
            <point key="canvasLocation" x="721" y="1227"/>
        </scene>
    </scenes>
    <resources>
        <image name="icons8-bluetooth-filled" width="30" height="30"/>
        <image name="music_library" width="30" height="30"/>
        <image name="uTuneRotated" width="322" height="322"/>
    </resources>
    <inferredMetricsTieBreakers>
        <segue reference="ssi-AZ-uI6"/>
```

```
    </inferredMetricsTieBreakers>
</document>
```

C.2.12 LaunchScreen.storyboard

```
<?xml version="1.0" encoding="UTF-8"?>
<document type="com.apple.InterfaceBuilder3.CocoaTouch.Storyboard.XIB" version="3.0"
    ↪ toolsVersion="14490.70" targetRuntime="iOS.CocoaTouch" propertyAccessControl="none
    ↪ " useAutolayout="YES" launchScreen="YES" useTraitCollections="YES" useSafeAreas="
    ↪ YES" colorMatched="YES" initialViewController="01J-lp-oVM">
    <device id="retina4_7" orientation="portrait">
        <adaptation id="fullscreen"/>
    </device>
    <dependencies>
        <deployment identifier="iOS"/>
        <plugIn identifier="com.apple.InterfaceBuilder.IBCocoaTouchPlugin" version
            ↪ ="14490.49"/>
        <capability name="Safe area layout guides" minToolsVersion="9.0"/>
        <capability name="documents saved in the Xcode 8 format" minToolsVersion="8.0"/>
    </dependencies>
    <scenes>
        <!--View Controller-->
        <scene sceneID="EHf-IW-A2E">
            <objects>
                <viewController id="01J-lp-oVM" sceneMemberID="viewController">
                    <view key="view" contentMode="scaleToFill" id="Ze5-6b-2t3">
                        <rect key="frame" x="0.0" y="0.0" width="375" height="667"/>
                        <autoresizingMask key="autoresizingMask" widthSizable="YES"
                            ↪ heightSizable="YES"/>
                        <subviews>
                            <stackView opaque="NO" contentMode="scaleToFill" axis="vertical"
                                ↪  translatesAutoresizingMaskIntoConstraints="NO" id="LGj-
                                ↪ Vq-4fA">
                                <rect key="frame" x="100" y="220" width="175" height="175"/>
                                <subviews>
                                    <imageView userInteractionEnabled="NO" contentMode="
                                        ↪ scaleToFill" horizontalHuggingPriority="251"
                                        ↪ verticalHuggingPriority="251" image="uTuneRotated"
                                        ↪  translatesAutoresizingMaskIntoConstraints="NO" id
                                        ↪ ="Hd8-n9-eJa">
                                        <rect key="frame" x="0.0" y="0.0" width="175" height
                                            ↪ ="175"/>
                                    </imageView>
                                </subviews>
                                <constraints>
                                    <constraint firstAttribute="width" secondItem="LGj-Vq-4fA
                                        ↪ " secondAttribute="height" multiplier="1:1" id="
                                        ↪ WSr-fh-sHR"/>
                                </constraints>
                            </stackView>
                        </subviews>
                        <color key="backgroundColor" red="1" green="1" blue="1" alpha="1"
                            ↪ colorSpace="custom" customColorSpace="sRGB"/>
                        <constraints>
                            <constraint firstAttribute="trailing" secondItem="LGj-Vq-4fA"
                                ↪ secondAttribute="trailing" constant="100" id="411-7i-cSO
                                ↪ "/>
```

```
                        <constraint firstItem="LGj-Vq-4fA" firstAttribute="leading"
                            ↪ secondItem="Bcu-3y-fUS" secondAttribute="leading"
                            ↪ constant="100" id="6Zs-W5-mgQ"/>
                        <constraint firstItem="LGj-Vq-4fA" firstAttribute="top"
                            ↪ secondItem="Bcu-3y-fUS" secondAttribute="top" constant
                            ↪ ="200" id="d8D-vX-iq0"/>
                    </constraints>
                    <viewLayoutGuide key="safeArea" id="Bcu-3y-fUS"/>
                </view>
            </viewController>
            <placeholder placeholderIdentifier="IBFirstResponder" id="iYj-Kq-Ea1"
                ↪ userLabel="First Responder" sceneMemberID="firstResponder"/>
        </objects>
        <point key="canvasLocation" x="53" y="375"/>
    </scene>
</scenes>
<resources>
    <image name="uTuneRotated" width="322" height="322"/>
</resources>
</document>
```

C.3 BM70 Programming

C.3.1 BM70_0EE0.hex

```
;Project Name : IS1870SF_102A;
;FW Information : 5505 102_BLDK3;
;FW Version : 0105;
;UI Table Version : 0101;
;UI Tool Version : IS187x_102_BLEDK3_UI_Configuration_Tool v100.132;
;Date : 19/05/08 13:54:17;
:020000040003F7
:1040000000000000000007554756E6500000000009F
:1040100000000000000000000000000000000000A0
:10402000000000000000003020026000000C0C014C
:1040300000000BAB000080018000002000002006CE
:10404000103003E000F0904022205050000000E4
:1040500000000101000000000000303000000000058
:1040600000000102000000000003C1E00000000F3
:10407000000010A040305060702090000000000110
:1040800000000000000000300000E0609755475D2
:104090006E650303F1FF020106000000000000004E
:1040A000000000000000000000000000000000000010
:1040B000000000000000000000000000000000000000
:1040C000000000000000000000003020106000E4
:1040D000000000000000000000000000000000E0
:1040E000000000000000000003030303030B0
:1040F000000000000000000000000000000000C0
:10410000000000000000000000000000000000AF
:1041100000000000000000000000000000000009F
:10412000000000AA55FFFFFFFFFFFFFFFFFFFF9B
:10413000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF8F
:10414000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF7F
:10415000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF6F
:10416000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF5F
```

```
:10417000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF4F
:10418000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF3F
:10419000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF2F
:1041A000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF1F
:1041B000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF0F
:1041C000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
:1041D000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEF
:1041E000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFDF
:1041F000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFCF
:10420000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFBE
:10421000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFAE
:10422000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF9E
:10423000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF8E
:10424000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF7E
:10425000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF6E
:10426000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF5E
:10427000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF4E
:10428000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF3E
:10429000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF2E
:1042A000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF1E
:1042B000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF0E
:1042C000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFE
:1042D000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEE
:1042E000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFDE
:1042F000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFCE
:10430000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFBD
:10431000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFAD
:10432000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF9D
:10433000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF8D
:10434000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF7D
:10435000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF6D
:10436000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF5D
:10437000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF4D
:10438000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF3D
:10439000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF2D
:1043A000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF1D
:1043B000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF0D
:1043C000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFD
:1043D000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFED
:1043E000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFDD
:1043F000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFCD
:10440000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFBC
:10441000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFAC
:10442000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF9C
:10443000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF8C
:10444000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF7C
:10445000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF6C
:10446000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF5C
:10447000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF4C
:10448000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF3C
:10449000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF2C
:1044A000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF1C
:1044B000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF0C
:1044C000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFC
```

```
:1044D000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEC
:1044E000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFDC
:1044F000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFCC
:10450000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFBB
:10451000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFAB
:10452000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF9B
:10453000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF8B
:10454000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF7B
:10455000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF6B
:10456000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF5B
:10457000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF4B
:10458000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF3B
:10459000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF2B
:1045A000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF1B
:1045B000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF0B
:1045C000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFB
:1045D000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEB
:1045E000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFDB
:1045F000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFCB
:10460000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFBA
:10461000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFAA
:10462000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF9A
:10463000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF8A
:10464000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF7A
:10465000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF6A
:10466000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF5A
:10467000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF4A
:10468000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF3A
:10469000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF2A
:1046A000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF1A
:1046B000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF0A
:1046C000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFA
:1046D000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEA
:1046E000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFDA
:1046F000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFCA
:10470000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFB9
:10471000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFA9
:10472000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF99
:10473000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF89
:10474000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF79
:10475000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF69
:10476000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF59
:10477000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF49
:10478000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF39
:10479000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF29
:1047A000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF19
:1047B000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF09
:1047C000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF9
:1047D000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFE9
:1047E000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFD9
:1047F000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFC9
:1048000000000000000007554756E65000000000097
:1048100000000000000000000000000000000098
:10482000000000000000003020026000000C0C0144
```

```
:10483000000000BAB0000800180000020000 2006C6
:10484000010300 3E000F0904022205050000000 0DC
:1048500000000101000000000000030300000000 0050
:1048600000000102000000000000 3C1E00000000 EB
:1048700000000 10A040305060702090000000000 108
:1048800000000000000000300000E0609755475CA
:104890006E650303F1FF0201060000000000000046
:1048A000000000000000000000000000000000 0008
:1048B000000000000000000000000000000000 00F8
:1048C000000000000000000000030201060000 DC
:1048D000000000000000000000000000000000 00D8
:1048E000000000000000000000303030303030A8
:1048F000000000000000000000000000000000 00B8
:1049000000000000000000000000000000000000A7
:1049100000000000000000000000000000000000 97
:10492000000000AA55FFFFFFFFFFFFFFFFFFFF93
:10493000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF87
:10494000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF77
:10495000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF67
:10496000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF57
:10497000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF47
:10498000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF37
:10499000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF27
:1049A000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF17
:1049B000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF07
:1049C000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF7
:1049D000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFE7
:1049E000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFD7
:1049F000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFC7
:104A0000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFB6
:104A1000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFA6
:104A2000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF96
:104A3000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF86
:104A4000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF76
:104A5000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF66
:104A6000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF56
:104A7000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF46
:104A8000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF36
:104A9000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF26
:104AA000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF16
:104AB000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF06
:104AC000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF6
:104AD000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFE6
:104AE000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFD6
:104AF000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFC6
:104B0000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFB5
:104B1000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFA5
:104B2000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF95
:104B3000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF85
:104B4000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF75
:104B5000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF65
:104B6000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF55
:104B7000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF45
:104B8000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF35
```

```
:104B9000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF25
:104BA000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF15
:104BB000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF05
:104BC000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF5
:104BD000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFE5
:104BE000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFD5
:104BF000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFC5
:104C0000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFB4
:104C1000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFA4
:104C2000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF94
:104C3000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF84
:104C4000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF74
:104C5000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF64
:104C6000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF54
:104C7000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF44
:104C8000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF34
:104C9000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF24
:104CA000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF14
:104CB000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF04
:104CC000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF4
:104CD000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFE4
:104CE000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFD4
:104CF000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFC4
:104D0000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFB3
:104D1000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFA3
:104D2000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF93
:104D3000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF83
:104D4000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF73
:104D5000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF63
:104D6000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF53
:104D7000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF43
:104D8000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF33
:104D9000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF23
:104DA000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF13
:104DB000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF03
:104DC000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF3
:104DD000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFE3
:104DE000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFD3
:104DF000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFC3
:104E0000353530352032031303235F424C444B3301323E
:104E1000010100000000000000000000000000090
:104E2000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF92
:104E3000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF82
:104E4000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF72
:104E5000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF62
:104E6000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF52
:104E7000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF42
:104E8000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF32
:104E9000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF22
:104EA000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF12
:104EB000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF02
:104EC000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF2
:104ED000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFE2
:104EE000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFD2
```

```
:104EF000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFC2
:104F0000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFB1
:104F1000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFA1
:104F2000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF91
:104F3000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF81
:104F4000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF71
:104F5000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF61
:104F6000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF51
:104F7000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF41
:104F8000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF31
:104F9000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF21
:104FA000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF11
:104FB000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF01
:104FC000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF1
:104FD000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFE1
:104FE000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFD1
:104FF000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFC1
:10500000800012800001800050000000000000052
:10501000000000000000000000000000A0002285C
:10502000030203000002A8000000000000000CE
:10503000000000000000001800030000424C4582
:105040002D535050000000000000000000000040
:10505000000000000A000428030205000012A0000E5
:10506000000000000000000000000000000000040
:10507000060005000000000000000000000025
:10508000000000000000000000000008001028E0
:10509000000A18002000000000000000000CE
:1050A000000000000000000A0011280302120006A6
:1050B000292A000000000000000000000009D
:1050C000000000000800120004953534300000094
:1050D000000000000000000000000000000D0
:1050E0000A00132803021400242A00000000000014
:1050F000000000000000000000000800140094
:1051000000424D37300000000000000000000A9
:10511000000000000000000A001528030216002D
:10512000252A000000000000000000000030
:10513000000000008001600003030303000000091
:1051400000000000000000000000000000005F
:105150000A00172803021800272A00000000000098
:10516000000000000000000000001200180015
:10517000035353035203130325F424C444B3300FE
:1051800000000000000000A00192803021A00B5
:10519000262A000000000000000000000000BF
:1051A000000000000C001A00003031303430313083
:1051B000310000000000000000000000000000BE
:1051C0000A001B2803021C00282A0000000000001F
:1051D000000000000000000000008001C00AB
:1051E000030303030000000000000000000FF
:1051F000000000000000A001D2803021E003D
:10520000232A000000000000000000000051
:1052100000000000C001E000000000000000064
:10522000000000000000000000000000007E
:105230000A001F28030220002A2A000000000A4
:10524000000000000000000000000C00200032
```

```
:10525000000000000010000000000000000004D
:10526000000000000000000160030280051BC5F64
:105270004DAC7A48939960825D43535349003300A3
:10528000000000018003128031832006EE217EF0A
:10529000DA974C959B3A6E02435353490000000045
:1052A0000500320000000000000000000000000C7
:1052B000000000000000000000000000070033298B
:1052C000020000000000000000000000000000DC
:1052D000000000000000001600402800336DD0E0
:1052E00038E26F4AA483CCD0C943535349004300EA
:1052F000000000018004128031842001803A628E7
:105300005ED8EC911C48A3AC435353490000000005
:1053100005004200000000000000000000000046
:10532000000000000000000000000070043290A
:1053300002000001000000000000000000000006A
:10534000000000000000001600502800055E40591
:10535000D2AF9FA98FE54A7DFE43535349005800C1
:1053600000000000180051280310C52001696244724
:10537000C62361BAD94B4D1E435353490100000067
:1053800005005200000000000000000000000C6
:10539000000000000000000000000070053298A
:1053A000020000020000000000000000000000F9
:1053B000000000000000018005428030C5500F5
:1053C000B39B7234BEECD4A8F44341884353534991
:1053D000010000005005500000000000000000072
:1053E000000000000000000000000000000000BD
:1053F000180056280318570007E3B07FF1C51492F01
:10540000B3398A4C435353490100000050057004B
:1054100000000000000000000000000000000008C
:1054200000000000000000007005829020000003EF
:1054300000000000000000000000000000000006C
:10544000000000FFFFFFFFFFFFFFFFFFFFFFFF68
:10545000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF5C
:10546000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF4C
:10547000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF3C
:10548000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF2C
:10549000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF1C
:1054A000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF0C
:1054B000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFC
:1054C000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEC
:1054D000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFDC
:1054E000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFCC
:1054F000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFBC
:10550000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFAB
:10551000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF9B
:10552000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF8B
:10553000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF7B
:10554000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF6B
:10555000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF5B
:10556000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF4B
:10557000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF3B
:10558000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF2B
:10559000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF1B
:1055A000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF0B
```

```
:1055B000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFB
:1055C000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEB
:1055D000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFDB
:1055E000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFCB
:1055F000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFBB
:10560000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFAA
:10561000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF9A
:10562000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF8A
:10563000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF7A
:10564000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF6A
:10565000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF5A
:10566000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF4A
:10567000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF3A
:10568000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF2A
:10569000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF1A
:1056A000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF0A
:1056B000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFA
:1056C000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEA
:1056D000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFDA
:1056E000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFCA
:1056F000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFBA
:10570000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFA9
:10571000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF99
:10572000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF89
:10573000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF79
:10574000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF69
:10575000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF59
:10576000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF49
:10577000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF39
:10578000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF29
:10579000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF19
:1057A000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF09
:1057B000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF9
:1057C000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFE9
:1057D000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFD9
:1057E000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFC9
:1057F000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFB9
:10580000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFA8
:10581000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF98
:10582000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF88
:10583000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF78
:10584000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF68
:10585000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF58
:10586000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF48
:10587000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF38
:10588000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF28
:10589000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF18
:1058A000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF08
:1058B000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF8
:1058C000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFE8
:1058D000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFD8
:1058E000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFC8
:1058F000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFB8
:10590000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFA7
```

```
:10591000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF97
:10592000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF87
:10593000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF77
:10594000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF67
:10595000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF57
:10596000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF47
:10597000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF37
:10598000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF27
:10599000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF17
:1059A000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF07
:1059B000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF7
:1059C000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFE7
:1059D000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFD7
:1059E000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFC7
:1059F000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFB7
:105A0000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFA6
:105A1000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF96
:105A2000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF86
:105A3000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF76
:105A4000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF66
:105A5000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF56
:105A6000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF46
:105A7000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF36
:105A8000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF26
:105A9000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF16
:105AA000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF06
:105AB000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF6
:105AC000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFE6
:105AD000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFD6
:105AE000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFC6
:105AF000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFB6
:105B0000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFA5
:105B1000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF95
:105B2000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF85
:105B3000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF75
:105B4000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF65
:105B5000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF55
:105B6000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF45
:105B7000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF35
:105B8000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF25
:105B9000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF15
:105BA000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF05
:105BB000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF5
:105BC000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFE5
:105BD000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFD5
:105BE000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFC5
:105BF000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFB5
:105C0000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFA4
:105C1000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF94
:105C2000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF84
:105C3000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF74
:105C4000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF64
:105C5000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF54
:105C6000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF44
```

```
:105C7000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF34
:105C8000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF24
:105C9000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF14
:105CA000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF04
:105CB000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF4
:105CC000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFE4
:105CD000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFD4
:105CE000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFC4
:105CF000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFB4
:105D0000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFA3
:105D1000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF93
:105D2000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF83
:105D3000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF73
:105D4000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF63
:105D5000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF53
:105D6000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF43
:105D7000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF33
:105D8000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF23
:105D9000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF13
:105DA000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF03
:105DB000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF3
:105DC000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFE3
:105DD000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFD3
:105DE000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFC3
:105DF000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFB3
:105E0000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFA2
:105E1000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF92
:105E2000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF82
:105E3000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF72
:105E4000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF62
:105E5000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF52
:105E6000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF42
:105E7000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF32
:105E8000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF22
:105E9000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF12
:105EA000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF02
:105EB000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF2
:105EC000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFE2
:105ED000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFD2
:105EE000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFC2
:105EF000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFB2
:105F0000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFA1
:105F1000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF91
:105F2000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF81
:105F3000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF71
:105F4000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF61
:105F5000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF51
:105F6000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF41
:105F7000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF31
:105F8000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF21
:105F9000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF11
:105FA000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF01
:105FB000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF1
:105FC000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFE1
```

```
:105FD000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFD1
:105FE000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFC1
:105FF000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFB1
:00000001FF
```

C.3.2 BM70.txt

```
;Project Name : IS1870SF_102A;
;FW Information : 5505 102_BLDK3;
;FW Version : 0105;
;UI Table Version : 0101;
;UI Tool Version : IS187x_102_BLEDK3_UI_Configuration_Tool v100.132;
;Date : 19/05/08 13:54:17;
;CheckSum : 0x1103;

;;; 0 1 2 3 4 5 6 7 8 9 A B C D E F
0000 00 00 00 00 00 00 75 54 75 6E 65 00 00 00 00 00 ;
0010 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ;
0020 00 00 00 00 00 00 00 03 02 00 26 00 00 0C 0C 01 ;
0030 00 00 00 BA B0 00 08 00 18 00 00 02 00 00 20 06 ;
0040 01 03 00 3E 00 0F 09 04 02 22 05 05 00 00 00 00 ;
0050 00 00 01 01 00 00 00 00 00 00 03 03 00 00 00 00 ;
0060 00 00 01 02 00 00 00 00 00 00 3C 1E 00 00 00 00 ;
0070 00 00 01 0A 04 03 05 06 07 02 09 00 00 00 00 01 ;
0080 00 00 00 00 00 00 00 03 00 00 0E 06 09 75 54 75 ;
0090 6E 65 03 03 F1 FF 02 01 06 00 00 00 00 00 00 00 ;
00A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ;
00B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ;
00C0 00 00 00 00 00 00 00 00 00 00 03 02 01 06 00 00 ;
00D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ;
00E0 00 00 00 00 00 00 00 00 00 00 30 30 30 30 30 30 ;
00F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ;

;;; 0 1 2 3 4 5 6 7 8 9 A B C D E F
0100 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ;
0110 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ;
0120 00 00 00 AA 55 FF FF FF FF FF FF FF FF FF FF FF ;
0130 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0140 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0150 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0160 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0170 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0180 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0190 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
01A0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
01B0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
01C0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
01D0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
01E0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
01F0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;

;;; 0 1 2 3 4 5 6 7 8 9 A B C D E F
0200 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0210 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0220 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0230 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0240 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0250 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0260 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
```

```
0270 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0280 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0290 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
02A0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
02B0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
02C0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
02D0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
02E0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
02F0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;

;;; 0 1 2 3 4 5 6 7 8 9 A B C D E F
0300 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0310 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0320 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0330 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0340 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0350 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0360 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0370 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0380 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0390 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
03A0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
03B0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
03C0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
03D0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
03E0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
03F0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;

;;; 0 1 2 3 4 5 6 7 8 9 A B C D E F
0400 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0410 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0420 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0430 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0440 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0450 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0460 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0470 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0480 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0490 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
04A0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
04B0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
04C0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
04D0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
04E0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
04F0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;

;;; 0 1 2 3 4 5 6 7 8 9 A B C D E F
0500 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0510 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0520 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0530 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0540 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0550 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0560 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
```

```
0570 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0580 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0590 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
05A0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
05B0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
05C0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
05D0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
05E0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
05F0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;

;;; 0 1 2 3 4 5 6 7 8 9 A B C D E F
0600 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0610 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0620 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0630 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0640 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0650 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0660 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0670 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0680 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0690 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
06A0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
06B0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
06C0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
06D0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
06E0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
06F0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;

;;; 0 1 2 3 4 5 6 7 8 9 A B C D E F
0700 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0710 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0720 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0730 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0740 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0750 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0760 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0770 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0780 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0790 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
07A0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
07B0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
07C0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
07D0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
07E0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
07F0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;

;;; 0 1 2 3 4 5 6 7 8 9 A B C D E F
0800 00 00 00 00 00 00 75 54 75 6E 65 00 00 00 00 00 ;
0810 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ;
0820 00 00 00 00 00 00 00 00 03 02 00 26 00 00 0C 0C 01 ;
0830 00 00 00 BA B0 00 08 00 18 00 00 02 00 00 20 06 ;
0840 01 03 00 3E 00 0F 09 04 02 22 05 05 00 00 00 00 ;
0850 00 00 01 01 00 00 00 00 00 00 03 03 00 00 00 00 ;
0860 00 00 01 02 00 00 00 00 00 00 3C 1E 00 00 00 00 ;
```

```
0870 00 00 01 0A 04 03 05 06 07 02 09 00 00 00 00 01 ;
0880 00 00 00 00 00 00 00 03 00 00 0E 06 09 75 54 75 ;
0890 6E 65 03 03 F1 FF 02 01 06 00 00 00 00 00 00 00 ;
08A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ;
08B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ;
08C0 00 00 00 00 00 00 00 00 00 00 00 03 02 01 06 00 00 ;
08D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ;
08E0 00 00 00 00 00 00 00 00 00 00 30 30 30 30 30 30 ;
08F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ;

;;; 0 1 2 3 4 5 6 7 8 9 A B C D E F
0900 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ;
0910 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ;
0920 00 00 00 AA 55 FF FF FF FF FF FF FF FF FF FF FF ;
0930 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0940 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0950 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0960 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0970 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0980 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0990 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
09A0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
09B0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
09C0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
09D0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
09E0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
09F0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;

;;; 0 1 2 3 4 5 6 7 8 9 A B C D E F
0A00 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0A10 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0A20 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0A30 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0A40 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0A50 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0A60 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0A70 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0A80 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0A90 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0AA0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0AB0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0AC0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0AD0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0AE0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0AF0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;

;;; 0 1 2 3 4 5 6 7 8 9 A B C D E F
0B00 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0B10 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0B20 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0B30 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0B40 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0B50 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0B60 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
```

```
0B70 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0B80 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0B90 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0BA0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0BB0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0BC0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0BD0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0BE0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0BF0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;

;;; 0 1 2 3 4 5 6 7 8 9 A B C D E F
0C00 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0C10 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0C20 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0C30 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0C40 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0C50 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0C60 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0C70 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0C80 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0C90 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0CA0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0CB0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0CC0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0CD0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0CE0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0CF0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;

;;; 0 1 2 3 4 5 6 7 8 9 A B C D E F
0D00 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0D10 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0D20 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0D30 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0D40 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0D50 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0D60 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0D70 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0D80 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0D90 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0DA0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0DB0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0DC0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0DD0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0DE0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0DF0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;

;;; 0 1 2 3 4 5 6 7 8 9 A B C D E F
0E00 35 35 30 35 20 31 30 32 5F 42 4C 44 4B 33 01 32 ;
0E10 01 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ;
0E20 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0E30 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0E40 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0E50 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0E60 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
```

```
0E70 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0E80 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0E90 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0EA0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0EB0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0EC0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0ED0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0EE0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0EF0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;

;;; 0 1 2 3 4 5 6 7 8 9 A B C D E F
0F00 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0F10 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0F20 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0F30 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0F40 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0F50 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0F60 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0F70 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0F80 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0F90 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0FA0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0FB0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0FC0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0FD0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0FE0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
0FF0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;

;;; 0 1 2 3 4 5 6 7 8 9 A B C D E F
1000 08 00 01 28 00 00 18 00 05 00 00 00 00 00 00 00 ;
1010 00 00 00 00 00 00 00 00 00 00 00 00 0A 00 02 28 ;
1020 03 02 03 00 00 2A 80 00 00 00 00 00 00 00 00 00 ;
1030 00 00 00 00 00 00 00 00 18 00 03 00 00 42 4C 45 ;
1040 2D 53 50 50 00 00 00 00 00 00 00 00 00 00 00 00 ;
1050 00 00 00 00 0A 00 04 28 03 02 05 00 01 2A 00 00 ;
1060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ;
1070 06 00 05 00 00 00 00 00 00 00 00 00 00 00 00 00 ;
1080 00 00 00 00 00 00 00 00 00 00 00 00 08 00 10 28 ;
1090 00 0A 18 00 20 00 00 00 00 00 00 00 00 00 00 00 ;
10A0 00 00 00 00 00 00 00 00 0A 00 11 28 03 02 12 00 ;
10B0 29 2A 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ;
10C0 00 00 00 00 08 00 12 00 00 49 53 53 43 00 00 00 ;
10D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ;
10E0 0A 00 13 28 03 02 14 00 24 2A 00 00 00 00 00 00 ;
10F0 00 00 00 00 00 00 00 00 00 00 00 00 08 00 14 00 ;

;;; 0 1 2 3 4 5 6 7 8 9 A B C D E F
1100 00 42 4D 37 30 00 00 00 00 00 00 00 00 00 00 00 ;
1110 00 00 00 00 00 00 00 00 0A 00 15 28 03 02 16 00 ;
1120 25 2A 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ;
1130 00 00 00 00 08 00 16 00 00 30 30 30 30 00 00 00 ;
1140 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ;
1150 0A 00 17 28 03 02 18 00 27 2A 00 00 00 00 00 00 ;
1160 00 00 00 00 00 00 00 00 00 00 00 00 12 00 18 00 ;
```

```
1170 00 35 35 30 35 20 31 30 32 5F 42 4C 44 4B 33 00 ;
1180 00 00 00 00 00 00 00 00 0A 00 19 28 03 02 1A 00 ;
1190 26 2A 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ;
11A0 00 00 00 00 0C 00 1A 00 00 30 31 30 34 30 31 30 ;
11B0 31 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ;
11C0 0A 00 1B 28 03 02 1C 00 28 2A 00 00 00 00 00 00 ;
11D0 00 00 00 00 00 00 00 00 00 00 00 00 08 00 1C 00 ;
11E0 00 30 30 30 30 00 00 00 00 00 00 00 00 00 00 00 ;
11F0 00 00 00 00 00 00 00 00 0A 00 1D 28 03 02 1E 00 ;

;;; 0 1 2 3 4 5 6 7 8 9 A B C D E F
1200 23 2A 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ;
1210 00 00 00 00 0C 00 1E 00 00 00 00 00 00 00 00 00 ;
1220 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ;
1230 0A 00 1F 28 03 02 20 00 2A 2A 00 00 00 00 00 00 ;
1240 00 00 00 00 00 00 00 00 00 00 00 00 0C 00 20 00 ;
1250 00 00 00 00 00 01 00 00 00 00 00 00 00 00 00 00 ;
1260 00 00 00 00 00 00 00 00 16 00 30 28 00 51 BC 5F ;
1270 4D AC 7A 48 93 99 60 82 5D 43 53 53 49 00 33 00 ;
1280 00 00 00 00 18 00 31 28 03 18 32 00 6E E2 17 EF ;
1290 DA 97 4C 95 9B 3A 6E 02 43 53 53 49 00 00 00 00 ;
12A0 05 00 32 00 00 00 00 00 00 00 00 00 00 00 00 00 ;
12B0 00 00 00 00 00 00 00 00 00 00 00 00 07 00 33 29 ;
12C0 02 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ;
12D0 00 00 00 00 00 00 00 00 16 00 40 28 00 33 6D D0 ;
12E0 38 E2 6F 4A A4 83 CC D0 C9 43 53 53 49 00 43 00 ;
12F0 00 00 00 00 18 00 41 28 03 18 42 00 18 03 A6 28 ;

;;; 0 1 2 3 4 5 6 7 8 9 A B C D E F
1300 5E D8 EC 91 1C 48 A3 AC 43 53 53 49 00 00 00 00 ;
1310 05 00 42 00 00 00 00 00 00 00 00 00 00 00 00 00 ;
1320 00 00 00 00 00 00 00 00 00 00 00 00 07 00 43 29 ;
1330 02 00 00 01 00 00 00 00 00 00 00 00 00 00 00 00 ;
1340 00 00 00 00 00 00 00 00 16 00 50 28 00 55 E4 05 ;
1350 D2 AF 9F A9 8F E5 4A 7D FE 43 53 53 49 00 58 00 ;
1360 00 00 00 00 18 00 51 28 03 1C 52 00 16 96 24 47 ;
1370 C6 23 61 BA D9 4B 4D 1E 43 53 53 49 01 00 00 00 ;
1380 05 00 52 00 00 00 00 00 00 00 00 00 00 00 00 00 ;
1390 00 00 00 00 00 00 00 00 00 00 00 00 07 00 53 29 ;
13A0 02 00 00 02 00 00 00 00 00 00 00 00 00 00 00 00 ;
13B0 00 00 00 00 00 00 00 00 18 00 54 28 03 0C 55 00 ;
13C0 B3 9B 72 34 BE EC D4 A8 F4 43 41 88 43 53 53 49 ;
13D0 01 00 00 00 05 00 55 00 00 00 00 00 00 00 00 00 ;
13E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ;
13F0 18 00 56 28 03 18 57 00 7E 3B 07 FF 1C 51 49 2F ;

;;; 0 1 2 3 4 5 6 7 8 9 A B C D E F
1400 B3 39 8A 4C 43 53 53 49 01 00 00 00 05 00 57 00 ;
1410 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ;
1420 00 00 00 00 00 00 00 00 07 00 58 29 02 00 00 03 ;
1430 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ;
1440 00 00 00 00 FF FF FF FF FF FF FF FF FF FF FF FF ;
1450 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1460 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
```

```
1470 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1480 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1490 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
14A0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
14B0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
14C0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
14D0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
14E0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
14F0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;

;;; 0 1 2 3 4 5 6 7 8 9 A B C D E F
1500 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1510 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1520 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1530 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1540 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1550 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1560 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1570 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1580 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1590 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
15A0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
15B0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
15C0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
15D0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
15E0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
15F0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;

;;; 0 1 2 3 4 5 6 7 8 9 A B C D E F
1600 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1610 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1620 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1630 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1640 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1650 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1660 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1670 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1680 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1690 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
16A0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
16B0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
16C0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
16D0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
16E0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
16F0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;

;;; 0 1 2 3 4 5 6 7 8 9 A B C D E F
1700 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1710 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1720 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1730 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1740 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1750 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1760 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
```

```
1770 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1780 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1790 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
17A0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
17B0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
17C0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
17D0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
17E0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
17F0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;

;;; 0 1 2 3 4 5 6 7 8 9 A B C D E F
1800 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1810 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1820 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1830 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1840 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1850 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1860 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1870 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1880 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1890 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
18A0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
18B0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
18C0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
18D0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
18E0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
18F0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;

;;; 0 1 2 3 4 5 6 7 8 9 A B C D E F
1900 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1910 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1920 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1930 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1940 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1950 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1960 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1970 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1980 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1990 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
19A0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
19B0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
19C0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
19D0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
19E0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
19F0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;

;;; 0 1 2 3 4 5 6 7 8 9 A B C D E F
1A00 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1A10 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1A20 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1A30 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1A40 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1A50 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1A60 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
```

```
1A70 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1A80 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1A90 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1AA0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1AB0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1AC0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1AD0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1AE0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1AF0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;

;;; 0 1 2 3 4 5 6 7 8 9 A B C D E F
1B00 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1B10 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1B20 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1B30 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1B40 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1B50 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1B60 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1B70 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1B80 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1B90 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1BA0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1BB0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1BC0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1BD0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1BE0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1BF0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;

;;; 0 1 2 3 4 5 6 7 8 9 A B C D E F
1C00 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1C10 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1C20 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1C30 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1C40 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1C50 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1C60 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1C70 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1C80 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1C90 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1CA0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1CB0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1CC0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1CD0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1CE0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1CF0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;

;;; 0 1 2 3 4 5 6 7 8 9 A B C D E F
1D00 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1D10 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1D20 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1D30 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1D40 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1D50 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1D60 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
```

```
1D70 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1D80 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1D90 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1DA0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1DB0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1DC0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1DD0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1DE0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1DF0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;

;;; 0 1 2 3 4 5 6 7 8 9 A B C D E F
1E00 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1E10 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1E20 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1E30 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1E40 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1E50 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1E60 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1E70 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1E80 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1E90 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1EA0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1EB0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1EC0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1ED0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1EE0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1EF0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;

;;; 0 1 2 3 4 5 6 7 8 9 A B C D E F
1F00 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1F10 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1F20 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1F30 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1F40 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1F50 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1F60 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1F70 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1F80 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1F90 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1FA0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1FB0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1FC0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1FD0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1FE0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
1FF0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
```