EDL Project Report

Persistence of Vision based display

Group D11:

Abhishek Singh (06D07037)

Prasad Thakur (06D07015)

Pravin Mote (06D07024)

Department of Electrical Engineering

Indian Institute of Technology Bombay

Aim: To create a persistence of vision (POV) based display by rotating an array of LEDs rapidly.

1. Introduction:

In deciding on a project we looked for a challenge that would have a good mix of hardware and software problems. We ended up primarily concentrating on looking at unusual display technologies and decided that a persistence of vision (POV) display would be a good balance of hardware and software. A POV display is a display created by rotating an array of LEDs rapidly. Due to the fact that human eyes can only render about 10 images per second, the fast spinning LEDS seem like a solid display.

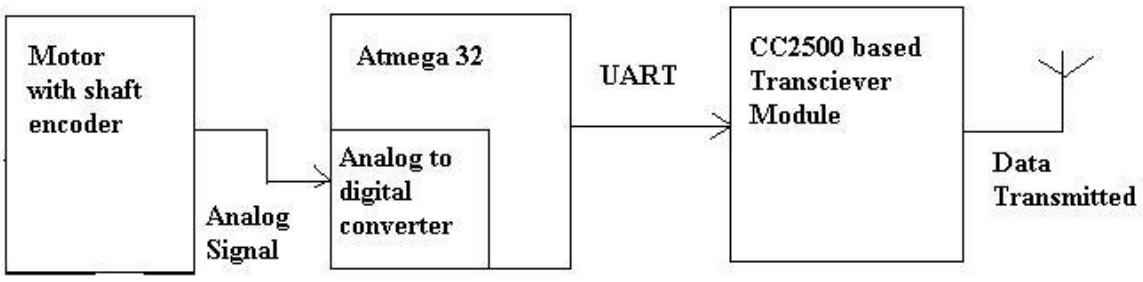
While many POV projects have been designed before, they tend to be similar, in the sense that the LED's are mounted perpendicular to the plane of the circuit. This may cause limitations to the characters and figures which can be displayed due to few number of LED.

Our original goal was to make a multi coloured LED display by using RGB (red-green-blue) LEDs. But the high cost of the RGB LEDs and the associated LED drivers prevented us from doing so. While we have made a monochrome (green) display, our design makes it easy for the user to swap the LED array for a different one and put in suitable LED drivers .Thus, one can customise the display as per the need.

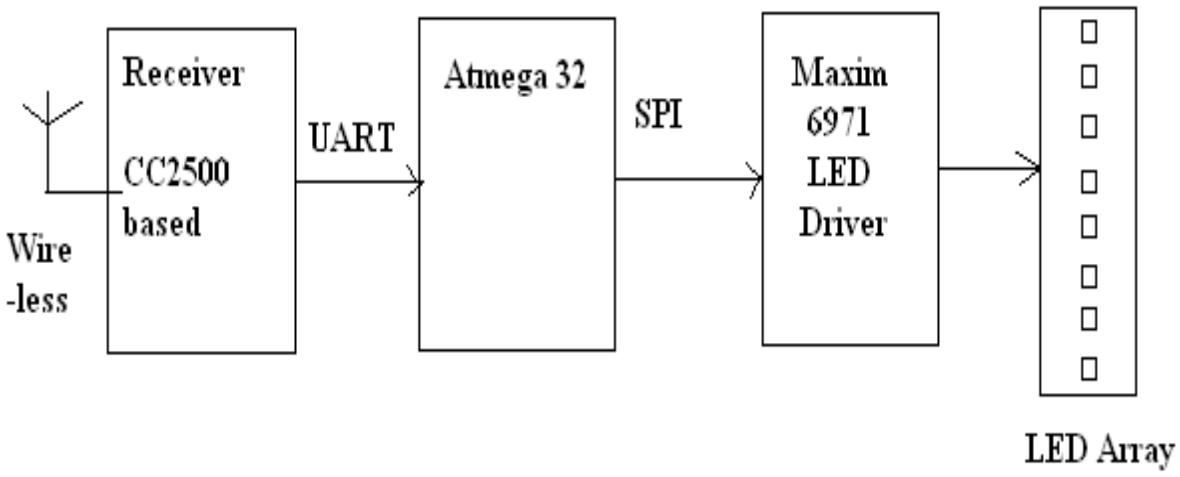
We also wanted to incorporate a feature which would transmit an image/character which is present on a wirelessly connected computer onto the rotating display. This could not be implemented as the high capacity (data-rate) transmitter and receiver units are very costly.

2. Block Diagram:

2.a.Base Circuit



b. Rotating Circuit:



3. Design:

We need two circuits for proper operation of the project.

1. Base (Non-rotating) Setup

2. Rotating Setup

3.1. Base Setup

The main feature of our design is that it is ‘position dependent’ rather than the commonly observed ‘rotating frequency dependent’ display. This kind of design is redundant to the fluctuations in the input of the power supply of the motor which may cause the speed of rotation of the motor to vary abruptly.

3.1.1. Motor

We discarded idea of using a Hall Sensor as it is inaccurate. Hence, we bought a motor with a shaft encoder circuit. The shaft encoder circuit is present at the rear of the motor. When this circuit is powered, and the motor is rotating, we can observe the output on a Cathode Ray Oscilloscope (CRO). We found out the number of divisions on our motor’s encoder by using a CRO and a Tachometer. Note that the encoder circuit can operate a maximum supply of 5 volts while the motor can operate at the maximum supply value of 12volts. Hence, we use 2 different voltage sources for the motor and encoder. However the ground connection is common to both.

The output of the encoder is in the form of pluses. Each pulse indicates that the motor shaft has rotated a specified number of degrees. In our case, the encoder has 62 divisions. This means when the motor rotates 360 degrees, we get 62 pulses. Thus, each pulse is obtained when the motor rotates 5.8 degrees from the previous position.

As we can precisely know the position of rotation, we have full control over the data which is displayed at that particular position.

3.1.2. Atmega 32 Microcontroller

When the encoder circuit is powered and the motor-shaft is rotating, the output of the encoder is fed to the Analogue-to-digital-converter (ADC) of the Atmel Atmega32 microcontroller. We then process this signal and convert it into a proportional digital value. The maximum value of the pulse obtained from the encoder is set to the hexadecimal value 'FF'. All other intermediate values are calibrated accordingly.

This use of ADC ensures that the output pulse on the encoder is due to the 5.8 degree rotation of the shaft and not due to any stray noise in the encoder circuit. Thus we can be sure that the encoder shaft has rotated 5.8 degrees when the value obtained from the ADC exceeds say FB.

We then count the number of times the ADC output exceeds FB to estimate the rotation of the encoder shaft in degrees.

We have programmed our ADC in such a way that we can detect 2.9 degrees of rotation of the shaft. This can be done by adjusting the threshold value used to compare ADC output (in the above example, we have used 'FB').

Thus we have increased our resolution from 5.8 to 2.9 degrees. Hence, we are able to control the display-data at each 2.9 degrees.

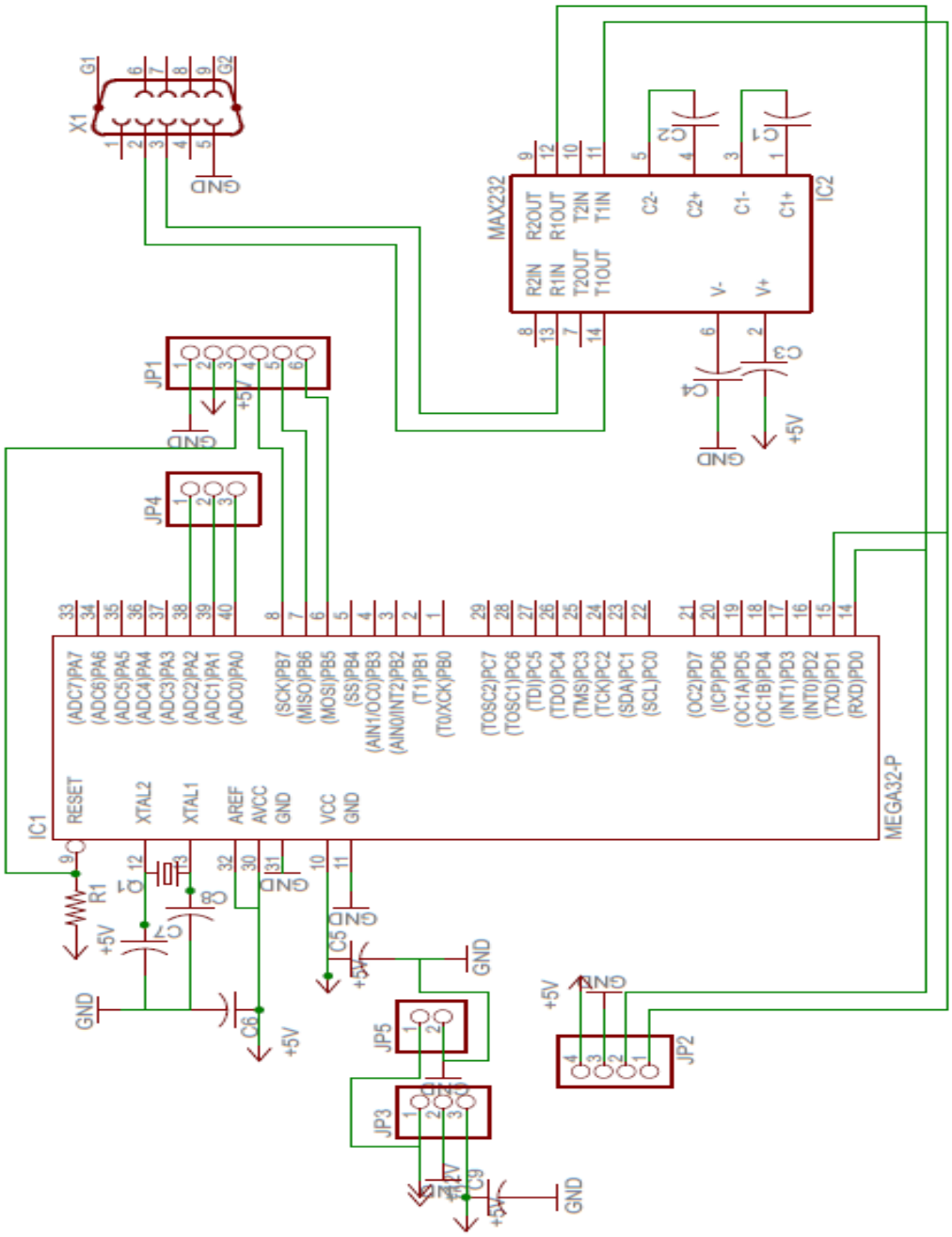
Now, keeping in mind the fact that human persistence of vision is 0.1 second, we have to display a particular data at a particular position more than 10 times in every second. Thus, we rotate the motor at about 15 turns per second. This translated to about 900 Rotations per minute (RPM). Our motor (CANON printer motor) has the maximum RPM of 5000.It also has enough torque to drive the load. Hence it is well suited for our need. Extremely good images require 20 – 30 frames per second, but since we have a spinning device, the higher rotation speed can be dangerous. As a result, we usually tried to use the lowest speed possible

As we have increased our resolution to 2.9 degrees, we can identify 124 distinct positions in one rotation (360 degrees) where data can be updated. We wirelessly transmit a signal '0x04' to the rotating receiver corresponding to each position traversed.

The receiver keeps count of this signal and can thus determine how much the shaft has rotated with respect to a reference position. This reference point is the position from which the motor starts its rotation on power-up.

.

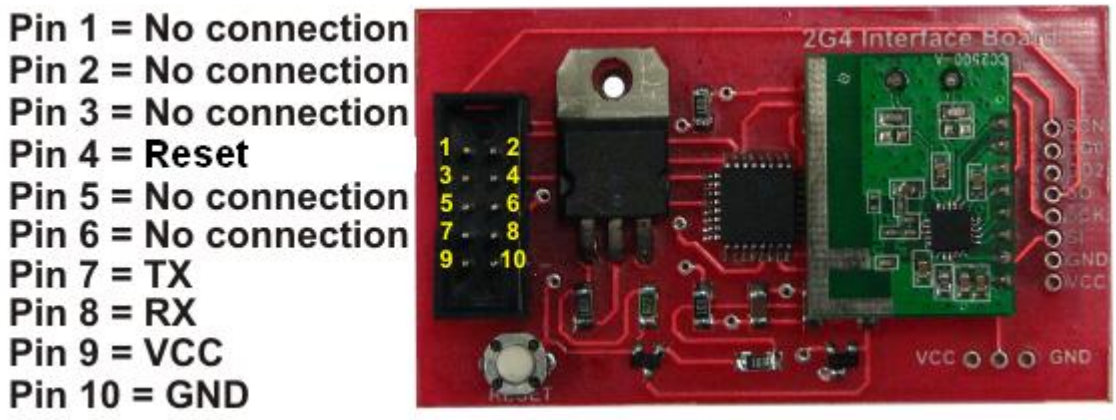
3.1.3. Base Circuit Diagram:



.

3.1.4. Transceiver Module

As mentioned earlier, the board also has a CC2500 transceiver module to communicate between the base and rotating circuits. We are using specially designed Interface Board which allows us to use this module using RS232 protocol (over a serial port at 38400 bps).



The protocol for initialization and data transfer through Interface Board is explained below:

After reset (system power on) it expects 3-configuration bytes.

Byte-1: Self address: (1-254) (0 or 255 for broadcast)

Byte-2: Remote address: (1-254) (0 or 255 for broadcast)

Byte-3: channel number: (1-254)

1. Self address:
The address used by rotating circuit to address the base circuit.

2. Remote address:
The address used by base circuit to address the rotating circuit.

3. Channel number:
The 8-bit unsigned channel number, which is multiplied by the channel spacing(199.950000 kHz) setting and added to the base frequency (2433.000000 MHz).

The self/remote address can be changed at any time by just giving a active low pulse on reset pin (pin4)and transmitting 3-bytes (self address, remote address, channel number).Here we are using only 2 transceiver modules one at the base circuit end and second on the rotating circuit.

In our program, we are initializing Rotating transceiver module as

Self address= 1,

Remote address= 2,
(address of the remote device i.e. address of transceiver connected at base)

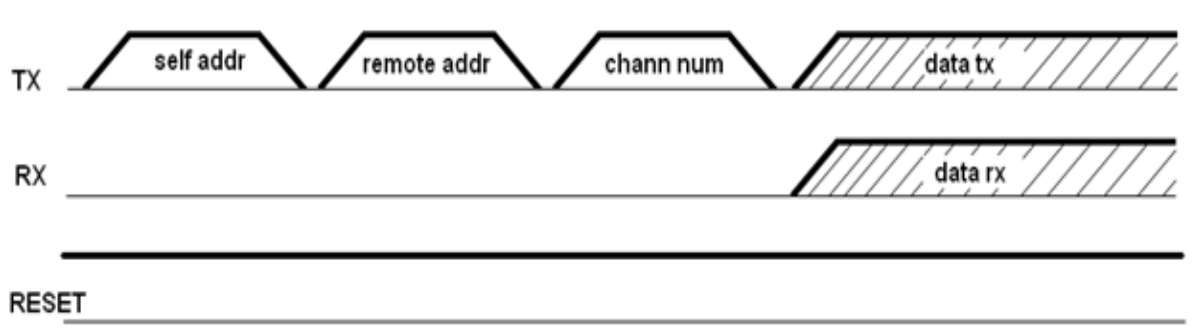
Channel as = 3 (selecting channel frequency over which device wants to communicate)

So **at the Base circuit** end we have initialization as:

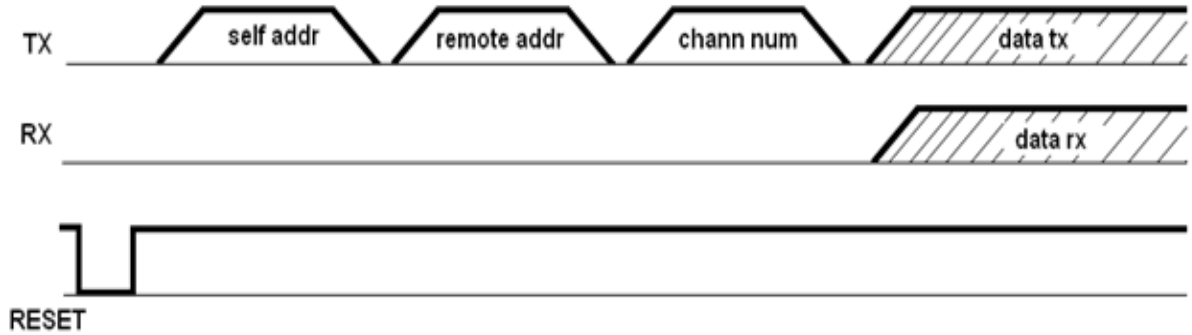
Self address= 2

Remote address= 1
(address of the remote device i.e. address of transceiver connected on rotating circuit)

Channel as= 3 (selecting channel frequency over which device wants to communicate).



module configuration diagram



reconfiguring addresses

As mentioned earlier, we wirelessly transmit the number '0x04' whenever the shaft rotates 2.9 degrees with respect to the previous position. (1 division out of 124)

RS232 to TTL Logic converter Design:

In order to test the Interface Board, we had to use the Hyper-terminal in a computer.

The logic levels for computer serial port are:

For Logic “HIGH” = -3 volts to -15 volts.

Logic “LOW”= +3 volts to +15 volts.

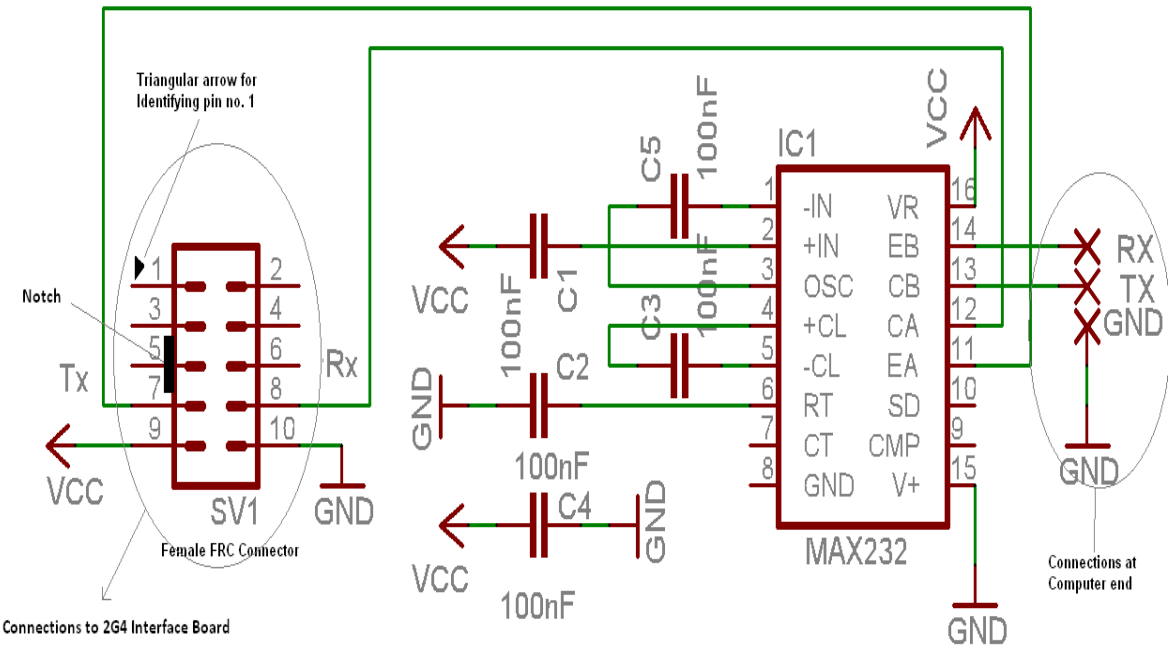
Wherein logic levels for microcontroller follows TTL/CMOS logic:

Logic “HIGH”= +5 volts

Logic “LOW”= 0 volts

Because of this if we connect these two devices directly they will not be able to recognize the data being transfer. In order to make these two devices understand each other we have RS232 to TTL logic converter which acts as a translator in between these two devices and allow them to communicate with each other.

Using MAX232 IC we build the RS232 to TTL logic converter. This IC generates the higher positive (+15 volts) and negative (-15 volts) voltages using +5volt power supply and some external capacitors. All the logic level conversion is done inside the IC when suitable external capacitors are connected. The schematic of circuit diagram is given below.



The Base Circuit also has 6 ports for programming the Atmega 32 micro-controller by an USBASP programmer. Thus one need not disturb the microcontroller from its position once fixed. This facility is quite handy when one has to test different codes while debugging the circuit/code-errors.

The base setup consists of a custom made holder (stand) for the motor. A small metal wheel is screwed to the upper-tip of the shaft. This serves as a reliable surface where the Rotating PCB is mounted and can be rotated at high speeds.

3.1.5 Base Circuit Power Source

All our devices in the base setup (except the motor) operate at 5 volts. Hence, we make use of a 7805 Voltage regulator to power the circuit.

3.2. Rotating Setup

The rotating setup consists of a PCB fitted with an Atmega 32L microcontroller, two Maxim 6971 Led drives,32 Green coloured LEDs,9 volts battery, a voltage regulator IC and a Microcontroller programming socket. One Led is fixed near the output of the battery in order to indicate if the circuit is powered up.

3.2.1 Power supply

We discarded the idea of using a brush contact system as it has contact issues.

The rotating circuit is powered by a 9 volt battery attached to it .This 9 volt is connected to a LM 317 voltage regulator. Thus, the rest of the circuit is powered by the output of the voltage regulator. The potentiometer associated with the LM 317 voltage regulator decides its output. We adjust the potentiometer in such a way that we get Vcc to the circuit to be 5volts.

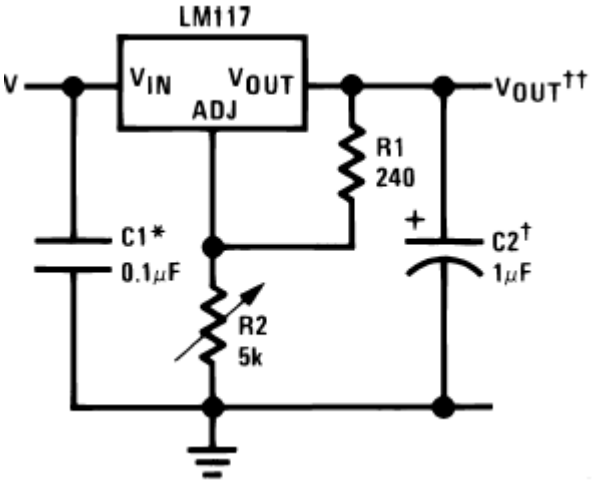


Figure.3.2.1.

1.2V–25V Adjustable Regulator

3.2.2. Receiver Module

We have a transceiver module mounted on the rotating circuit. It is configured so as to act as a receiver. We wirelessly transmit a signal ’0x01’ from the base circuit to the rotating receiver corresponding to each position traversed. The data ‘0x01’ is received here.

As mentioned earlier, we are initializing Rotating transceiver module as:

Self address= 1,

Remote address= 2,
(address of the remote device i.e. address of transceiver connected at base)

Channel as = 3 (selecting channel frequency over which device wants to communicate)

This module is connected to the Amega 32L microcontroller by the USART. The connections are shown in the circuit diagram of the rotating circuit. We keep polling for the data (from the wireless module) at the Rx pin of the Atmega 32L microcontroller. Once the data is received, the program enters into function which takes care of the display (controlling of the LEDs). The 'Rx' of the wireless module is connected to the 'Rx' of the Atmega 32L.

3.2.3. LEDs

We use locally available surface mounted LEDs. They are in 1206 package. We use monochromatic led (green in our case) as they are much cheaper than the RGB (red-green-blue) LEDs. Each Led driver (Max6971) can drive 16 LEDs. We use an array of 32 LEDs in our rotating display. The Anode of each LED is connected to Vcc while the cathode end is connected to the output pins of the Led driver.

Also, the disadvantage of using RGB led is that they need 3 LED drivers for driving 16 Leds. This is three times the number required for regular monochrome LED. This increases the cost of the circuit beyond justification. Also, the current being drawn by RGB LEDs is very high as compared to the Green LED (which we are using).

The LED array is always placed radial outward direction, starting at the axis of rotation of the board. This is done in order to create a horizontal circular plane on which we can display the output.

These LEDs are soldered close to each other in order to increase resolution of the image created when the motor is rotated at suitable speeds. The LEDs are operated at 5 volts.

.

3.2.4. LED Driver

The need for a LED driver arises as we want to drive large number of LEDs from a single microcontroller. We searched a lot for a driver which would be easy to interface with the microcontroller and could control sufficient number of LEDs.

We decided to go for Maxim's 6971 driver as Maxim was ready to supply free samples of the same. We designed the remaining circuit accordingly. But later on we were dejected to know that our free samples were cancelled by the company. Hence we had to buy them.

The MAX6971 LED driver comprises a 4-wire serial interface driving 16 constant-current-sinking, open drain output ports. The constant current outputs are guaranteed for current

accuracy not only with chip-supply voltage variations (5V ±10% and 3V to 5.5V), but also over a realistic range of driver output voltage drop (0.8V to 2.5V). The 4-wire serial interface comprises a 16-bit shift register and a 16-bit transparent latch. The shift register is written through a clock input, CLK, and a data input, DIN, and the data propagates to a data output, DOUT. The data output allows multiple drivers to be cascaded and operated together. The contents of the 16-bit shift register are loaded into the transparent latch through a latch-enable input, LE. The latch is transparent to the shift register outputs when high, and latches the current state on the falling edge of LE.

Output-Enable: High forces outputs OUT0 to OUT15 high impedance, without altering the contentsof the output latches. Low enables outputs OUT0 to OUT15 to follow the state of the output latches

Each driver output is an open-drain, constant-current sink that should be connected to the cathode of either a single LED or a series string of multiple LEDs. The LED anode can be connected to a supply voltage of up to 36V, independent of the MAX6971 supply, V+. The constant - current capability is up to 55mA per output, set for all eight outputs by an external resistor, RSET.

4-Wire Serial Interface:

The serial interface on the MAX6971 is a 4-wire serial interface using four inputs (DIN, CLK, LE, Output Enable) and a data output (DOUT). This interface is used to write display data to the MAX6971. The serial-interface data word length is 16 bits, D0–D15. The functions of the five interface pins are as follows:
DIN is the serial-data input, and must be stable when it is sampled on the rising edge of CLK. Data is shifted in, MSB first. This means that data bit D15 is clocked in first, followed by 15 more data bits finishing with the LSB, D0. CLK is the serial-clock input, which shifts data at DIN into the MAX6971 16-bit shift register on its rising edge. LE is the latch load input of the MAX6971 that transfers data from the MAX6971 16-bit shift register to its 16-bit latch when LE is high (transparent latch), and latches the data on the falling edge of LE.
The fourth input provides output-enable control of the output drivers. OE is high to force outputs OUT0–OUT15 high impedance, without altering the contents of the output latches, and low to enable outputs OUT0–OUT15 to follow the state of the output latches.
OE is independent of the operation of the serial interface. Data can be shifted into the serial-interface shift register and latched, regardless of the state of OE. DOUT is the serial-data output, which shifts data out from the MAX6971's 16-bit shift register on the rising edge of CLK. Data at DIN is propagated through the shift register and appears at DOUT 16 clock cycles later.

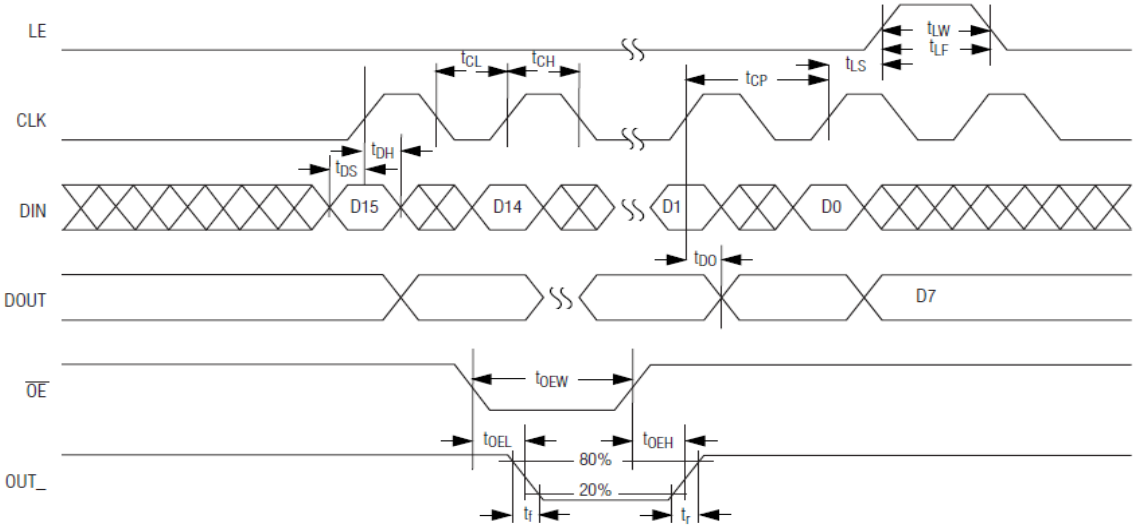
In our circuit, we make use of 2 LED drivers as we want to drive 32 LEDs.

The LED drivers are cascaded one after the other. Thus, we can send serially send 4 bytes of data to control the 32 LEDs.

When 2 bytes of Data has been transferred, the 'serial to parallel shift register' inside the driver is full. When the remaining data is received at Din of driver, the data in the above mentioned register is clocked out into the cascaded LED driver. Now, both 'serial to parallel shift registers' are full and ready for operation

Thus, we can increase the number of LEDs by increasing the number of cascaded drivers. The MISO(master input slave output) pin of the Atmega 32L in the rotating circuit has to be disconnected from the LED drivers in order to eliminate the errors in the programming of the microcontroller.

4-Wire Serial-Interface Timing Diagram



Selecting External Component RSET to Set LED Output Current

The MAX6971 uses an external resistor, RSET, to set the LED current for outputs OUT0–OUT15. The minimum allowed value of RSET is 327.3Ω, which sets the output currents to 55mA. The maximum allowed value of RSET is 5kΩ. The reference value, 360Ω, sets the output currents to 50mA. To set a different output current, use the formula:

RSET = 18,000 / IOUT where IOUT is the desired output current in mA.

Computing Power Dissipation

The upper limit for power dissipation (PD) for the MAX6971 is determined by the following equation:

PD = (V+ x I+) + (VOUT x DUTY x IOUT x N)

Where:

V+ = supply voltage

I+ = operating supply current when sinking IOUT LED drive current into N outputs

DUTY = PWM duty cycle applied to OE

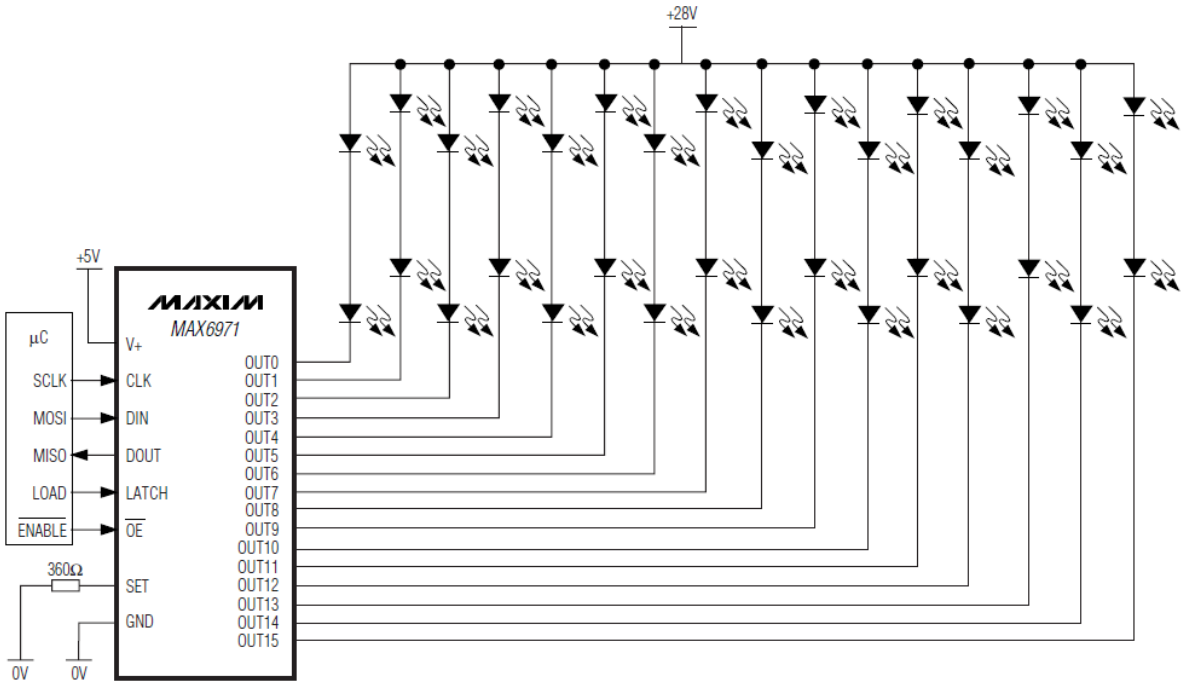
N = number of MAX6971 outputs driving LEDs at the same time (maximum is 16)

VOUT = MAX6971 port output voltage when driving load LED(s)

IOUT = LED drive current programmed by RSET

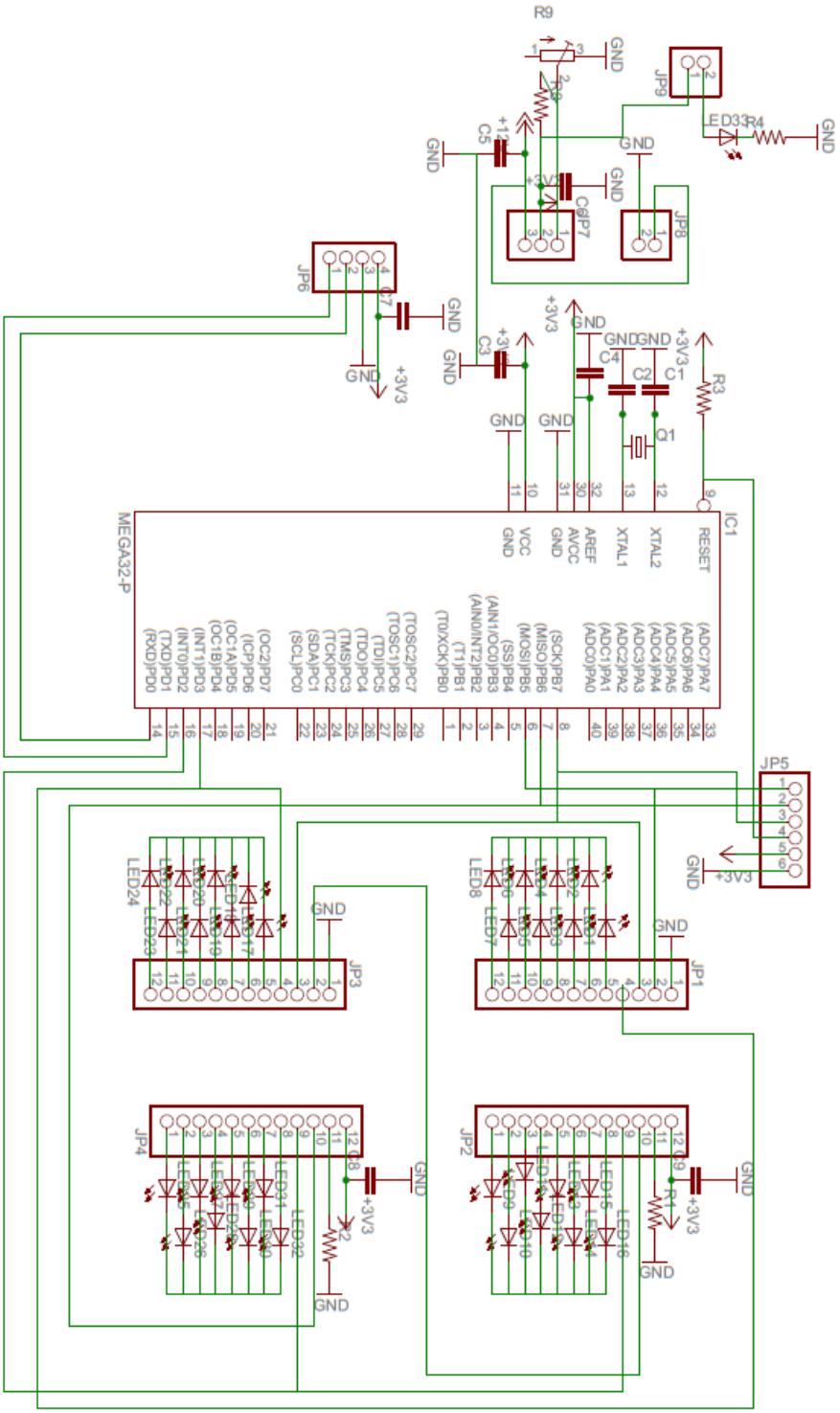
PD = power dissipation, in mW if currents are in mA

Typical Application Circuit:



These drivers are able to turn the LEDs on by sending a preset current to them which is calculated with the equation I = 18/Rset as mentioned above. We use 10K ohm value so that they are not too bright to view. This means that the LED's were getting 1.8mA each, so the final current drawn by all 32 LEDs is 57.6mA. We store the LEDs on off state in a register with each bit corresponding to an LED in their shift registers. However, the LED states are only updated from this register when the Latch enable pin is high in order to prevent LEDs from switching states in the middle of a frame.

3.2.5. Circuit Diagram of the rotating Setup



4. Major Problems Faced

1. Initially we faced lot of logistical problems. The most important being the procurement of the LED driver IC. We placed a free sample request with Maxim for the MAX6971 led driver. We also got a confirmation email regarding the same. However, the free samples were not delivered to us even after 20 days. On contacting Maxim, they informed that our free samples were cancelled.

Hence, we left with no option but to buy them online from Farnell Electronics with a special permission from Prof. Vijaykumaran. Thus, we got the LED drivers almost a month late.

2. There was a serious problem regarding the quality of Printed circuit boards supplied to us by the PCB Lab. Three units of the rather complex 'Rotating circuit' were damaged due to improper etching of the copper tracks and IC pads. None of the 'vias' were connected from when shifting from one layer to the other.

Considering the fact that this would lead to the failure of our project, we got the PCB manufactured from a professional PCB maker. This increased the cost of our circuit.

3.The manufacturers of the wireless modules were in a major confusion regarding the functionality of some input/output pins in their kit. This was demonstrated by the incomplete/incorrect information provided by them in their user manuals. Also we came to a conclusion that garbage values were transmitted at the start of the transmission of data . There was no mention of this in the user manual. We added appropriate delays to solve his problem.

> It took us nearly 2 days of work to correct their mistakes and make the wireless module functional.

4. We faced lot of trouble soldering small connections due to the unavailability of appropriate soldering iron and other tools.

5. Conclusion

Since our display is designed to work without any predetermined spin speed, upgrading our motor and display module would allow us to improve the display with minimal changes.

On the design side, we independently developed all our hardware and software. We also made sure that our project would not cause ethical concerns. While there is a slight chance of injury from the spinning board, we keep the speed low enough to prevent the chance of injury. We also used a non-default channel on our wireless receiver to avoid interfering with other group's projects.

Since the beginning of the project, we have maintained a **Google-group** for posting updates and important documents regarding the project. Our **teaching assistant is also a member** of this group.

Appendices

References

LED driver datasheet - http://datasheets.maxim-ic.com/en/ds/MAX6971.pdf

Wireless Module: http://www.thinklabs.in/resources/?p=1025.

Atmega 32 Datasheet

7805 Voltage regulator datasheet

LM317 Voltage regulator datasheet

Base circuit Partlist

Exported from base_final.sch at 02-12-2009 12:24:58

EAGLE Version 5.6.0 Copyright (c) 1988-2009 CadSoft

Part	Value	Device	Package	Library	Sheet
7805_REGULATOR		PINHD-1X3	1X03	pinhead	1
ADC-PORT-PINS		PINHD-1X3	1X03	pinhead	1
C1		C-US025-025X050	C025-025X050	rcl	1
C2		C-US025-025X050	C025-025X050	rcl	1
C3		C-US025-025X050	C025-025X050	rcl	1
C4		C-US025-025X050	C025-025X050	rcl	1
C5		C-US025-025X050	C025-025X050	rcl	1
C6		C-US025-025X050	C025-025X050	rcl	1
C7		C-US025-025X050	C025-025X050	rcl	1
C8		C-US025-025X050	C025-025X050	rcl	1
C9		C-US025-025X050	C025-025X050	rcl	1
IC1	MEGA32-P	MEGA32-P	DIL40	atmel	1
IC2	MAX232	MAX232	DIL16	maxim	1
JP5		PINHD-1X2	1X02	pinhead	1
PROGRAMMING-PINS		PINHD-1X6	1X06	pinhead	1
Q1		CRYSTALHC49S	HC49/S	crystal	1
R1		R-US_0207/7	0207/7	rcl	1
RS232-CONNECTOR		F09HP	F09HP	con-subd	1

WIRELESS-MODULE PINHD-1X4 1X04 pinhead 1

Top Partlist

Exported from final_done_new.sch at 02-12-2009 12:16:01

EAGLE Version 5.6.0 Copyright (c) 1988-2009 CadSoft

Part	Value	Device	Package	Library	Sheet
C1	0.1uF	C-EU025-025X050	C025-025X050	rcl	1
C2		C-EU025-025X050	C025-025X050	rcl	1
C3		C-EU025-025X050	C025-025X050	rcl	1
C4		C-EU025-025X050	C025-025X050	rcl	1
C5		C-EU025-025X050	C025-025X050	rcl	1
C6		C-EU025-025X050	C025-025X050	rcl	1
C7		C-EU025-025X050	C025-025X050	rcl	1
C8		C-EU025-025X050	C025-025X050	rcl	1
C9		C-EU025-025X050	C025-025X050	rcl	1
IC1	MEGA32-P	MEGA32-P	DIL40	atmel	1
JP2		PINHD-1X12	1X12	pinhead	1
JP4		PINHD-1X12	1X12	pinhead	1
JUMPER		PINHD-1X2	1X02	pinhead	1
LED1		LEDSMT1206	1206	led	1
LED2		LEDSMT1206	1206	led	1

LED3	**LEDSMT1206**	**1206**	**led**	**1**
LED4	**LEDSMT1206**	**1206**	**led**	**1**
LED5	**LEDSMT1206**	**1206**	**led**	**1**
LED6	**LEDSMT1206**	**1206**	**led**	**1**
LED7	**LEDSMT1206**	**1206**	**led**	**1**
LED8	**LEDSMT1206**	**1206**	**led**	**1**
LED9	**LEDSMT1206**	**1206**	**led**	**1**
LED10	**LEDSMT1206**	**1206**	**led**	**1**
LED11	**LEDSMT1206**	**1206**	**led**	**1**
LED12	**LEDSMT1206**	**1206**	**led**	**1**
LED13	**LEDSMT1206**	**1206**	**led**	**1**
LED14	**LEDSMT1206**	**1206**	**led**	**1**
LED15	**LEDSMT1206**	**1206**	**led**	**1**
LED16	**LEDSMT1206**	**1206**	**led**	**1**
LED17	**LEDSMT1206**	**1206**	**led**	**1**
LED18	**LEDSMT1206**	**1206**	**led**	**1**
LED19	**LEDSMT1206**	**1206**	**led**	**1**
LED20	**LEDSMT1206**	**1206**	**led**	**1**
LED21	**LEDSMT1206**	**1206**	**led**	**1**
LED22	**LEDSMT1206**	**1206**	**led**	**1**
LED23	**LEDSMT1206**	**1206**	**led**	**1**
LED24	**LEDSMT1206**	**1206**	**led**	**1**
LED25	**LEDSMT1206**	**1206**	**led**	**1**
LED26	**LEDSMT1206**	**1206**	**led**	**1**
LED27	**LEDSMT1206**	**1206**	**led**	**1**
LED28	**LEDSMT1206**	**1206**	**led**	**1**

LED29	LEDSMT1206	1206	led	1
LED30	LEDSMT1206	1206	led	1
LED31	LEDSMT1206	1206	led	1
LED32	LEDSMT1206	1206	led	1
LED33	LED3MM	LED3MM	led	1
LM317_REGULATOR	PINHD-1X3	1X03	pinhead	1
MAX6971(ONE)	PINHD-1X12	1X12	pinhead	1
MAX6971(TWO)	PINHD-1X12	1X12	pinhead	1
PROGRAMMING_PINS	PINHD-1X6	1X06	pinhead	1
Q1	CRYSTALHC49S	HC49/S	crystal	1
R1	R-US_0207/7	0207/7	rcl	1
R2	R-US_0207/7	0207/7	rcl	1
R3	R-US_0207/7	0207/7	rcl	1
R4	R-US_0207/7	0207/7	rcl	1
R8	R-US_0207/7	0207/7	rcl	1
R9	R-TRIMM64P	RTRIM64P	rcl	1
VOLTAGE_SOURCE	PINHD-1X2	1X02	pinhead	1
WIRELESS-MODULE	PINHD-1X4	1X04	pinhead	1