EE 491-492
Spinning Display Project

Ali ALICI

Funda Kubra CİMİLLİ

Cengiz ÇOKAY

Sefa DEMİRTAŞ

Yusuf YİĞİT

Submitted to : Assist. Prof. Şenol Mutlu

Prof. Avni Morgül

Assist. Prof. Arda Deniz Yalçınkaya

ABSTRACT

A spinning display is a device that creates a stable image which is constructed by a stick turning around its centre which is filled with LEDs. By this way, using limited number of leds, we can construct an image in the form of normal displays. As far as the conventional LED displays are concerned, the revolutionary approach pursued by this type of display can be noticed remarkably. This mentioned revolutionary approach may be described as following; in conventional displays, even in LCD's, each pixel is defined as a hardware pixel. In our display approach; we have 2 coordinate axes: one of them is r; which is implemented physically and the second one is θ which is implemented virtually. In our display, the resolution principle is totally different but for comparison, a total number of 24*pi*16 virtual pixels can be created by only using 16 pixels[1]. In this project, we implemented the general idea of spinning displays using digital components and a DC motor. The finalized project consists of 48 LEDs turning around an orbit and creating a constant 24-bit colored image. The colored image is created by using three LEDs (red, green and blue) for each circle in r dimension. Developed as a graduate project, this work is performed as an experiment to see this interesting idea in work and allowed applying many tools together. In this report, we explain each step of this design in detail.

[1] The details of this calculation will be provided in subsequent chapters.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

Introduction

1.1 Introduction

"Spinning Display" is a general name for the type of displays which create images using one or small number of columns of LEDs placed on a plate. The plate is attached to a motor which will cause its turn around a circular orbit. By this way, the LEDs turning around this orbit will create an image for human eye, using the fact that the human eye is not able to clearly distinguish movements beyond a particular frequency.

There are some examples of these type of displays created by both hobbyists and commercial companies, but our project has a more complete and different approach in mind. First of all, most of these examples have no led intensity controls or sophisticated color creation. In fact, this makes them incomparable to our project, which has both intensity controls for each led and 24-bit color creation. In some sophisticated commercial products, we see sharp led positioning controls and very good color schemes, but actually they have plates turning around the orbit which is perpendicular to itself. By this approach, they create a 360 degree view in contrast to our display which has a flat view and different sized pixels.

Our "Spinning Display" consists of LEDs on a plate turning around an orbit to create 24-bit colored display. In the first semester of our project, we implemented a 16 LED grayscale version of our project, and this semester we designed a 48 LED 8-bit colored version.

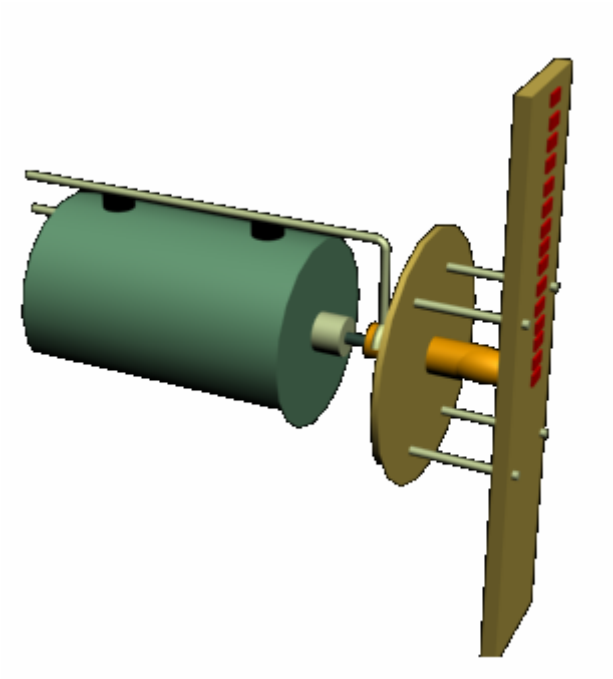


Figure 1.1: Spinning Display Model

As far as the conventional LED displays are concerned, the revolutionary approach pursued by this type of display can be noticed remarkably. This

mentioned approach may be described as following; in conventional displays, even in LCD's, each pixel is defined as a hardware pixel. If you want to design a 1024*768 display, you have to use 1024*768 LEDs. However, in our display approach we have 2 coordinate axes: one of them is r; which is implemented physically and the second one is theta which is implemented virtually. In our display, the resolution principle is totally different, but for comparison, a total number of 24*pi*16 virtual pixels can be created by only using 16 pixels[2].

The project is also multidisciplinary which gave us the chance to facilitate the experience derived from the courses we took. Actually we can present the disciplines involved in this project as follows:

- Electronics: To drive the LEDs and to provide necessary voltages
- Digital Design: To provide the interoperatibility between various digital ICs and design the digital system.
- Image processing: To convert the cartesian coordinates to polar ones.
- Digital Communication: To transfer the data from the computer to the device
- Software : To provide the software backend for the communication and processing

1.2 Objectives

Our main objective was to create a low cost display with a different approach to reach the standards from a different way. The LEDs, being versatile lighting devices are used in this design which we expect to perform equal or better in some respects compared to conventional displays. The following objectives were set at the start of the project:

1.2.1 High Resolution in a Small Area

We are using to use 16 LEDs for each color in the project. Our LEDs have 1.0mm x 0.5mm dimensions and there is 2.1mm spacing between neighboring LEDs. The LEDs are placed so that they create a screen that has a radius of approximately 33mm. It may be considered as a tiny screen but the pixels at the outermost row will have pixels only as big as a led and we will have smaller and smaller pixel sizes as we progress to the interior. In this respect, the overall screen is to be a high resolution one which we expected to see the image transmitted to it clearly.

1.2.2 Full 8 bit intensity control

8 bit intensity control objective was set since the beginning of the project. Since that time, we always progressed without changing this. Currently, we use 48 LEDs, all of which are driven by 8 bits. Their intensity will be controlled by our circuit on the plate using Pulse Width Modulation. We expect our PWM implementation not to cause any problems with intensity control because we are implementing it directly into the RAM – counter combination which is faster than conventional LED drivers.

---

[2] The details of this calculation will be provided in subsequent chapters.

1.2.3 Consistent view without flicker

Another specification that is not changed from the start is the speed of the motor, 3600 rpm, which directly effects the refresh rate, hence the flicker. So, we are setting our motor to rotate at 3600 rpm, which will give us a refresh rate of 60Hz, which is considerably higher that human flicker detection frequency.

1.2.4 An interesting and attractive model

In the attractiveness side, we wanted to have a display with an appealing design. This objective is more important for us than what it was in the previous term, so we made significant progress in this, which we explain in the next chapters. The design will give the people and also us the first impression about the project, so we are tried to create something interesting and attractive

1.2.5 Brighter Display

A brighter display is at most times desirable. We try to get the best from our displays at sunlight. Using the intrinsic brightness of LEDs, we expected that our display has a brightness higher than most of the displays. Our drawback here is we are using time division which will also divide the time that the eye is met with the light from a led. We saw the brightness of the LEDs in previous design, so we now have an idea of their intensity. Actually, it seemed pretty much enough for our needs. What we changed in our design gave us more time division, so we expect more diminishing of intensity, but also there are more LEDs now.

1.2.6 Wide Viewing Area

In everyday life, we usually share one display with many people, like we do in watching TV. In the areas where more than one people sitting close to a display are present, we expect from a display to be viewable from different viewing angles. Another design concern is, as expected viewing area. The LEDs we use have viewing angles as high as 150 degrees, so we will try to achieve wide viewing angles with these leds. The moving of LEDs may spoil a perfect experience, but we will try to reach considerable viewing areas.

1.2.7 Higher Refresh Rates – No Flicker

One of the problems of CRT displays that LCD displays solved was the refreshing problem. Physically, we seem to have a serious problem here. Our LEDs are always moving, and at every turn, we refresh all the points. Our motor will have a limited speed, so we will try hard to optimize it for a better display. We think that no flicker will be detected, but still the eye will be tired looking at our screen. Actually that was the case with 60Hz CRT screens. We plan to turn the motor at 3600rpm, which corresponds to 60Hz operating frequency.

1.2.8 Smoothness vs Pixel Based Display

We always talk about pixels when we discuss about cameras and displays. Actually, we have an advantage here because we have no distinct pixel sizes at the axis we are rotating over. We expect that it will give the eye a more continuous view,

but the results may be different. Our objective is to use this interesting fact to achieve a smoother view. We can also make the pixel changes as frequent as we want.

1.2.9 Contrast – Color Depth

Color Depth was an objective we could not achieve in the previous semester. In this semester, it entered our short term objective list. Actually, our current design includes a 24 bit color approach with red, green and blue LEDs, each driven by 8 bits. This will create a true color scheme, which will, if appropriately designed give us a perfect color. In the contrast, we have a black that can be called "pure". It is because we do not light the LEDs while creating the black color. It will give us a very high contrast ratio, so one of our aims is to use this fact as an advantage.

1.2.10 Cheapness

One of the objectives when we started was cheapness. We thought about reducing the number of pixels as a price advantage, but it seems that we have some problems here. We tried hard to keep the project cheap, but some unexpected factors seemed to block us. To give an example, due to the nature of the materials of LEDs, blue LEDs are expensive compared to others. If we add the costs of circuit elements, exterior design, PCB building and so, we think there are problems. But, as we said earlier, we always tried hard to keep the cost as low as possible.

CHAPTER 2

Led Driving – Timings – PWM

2.1 A new resolution concept

Resolution (or spatial resolution) is defined as "A measure of the accuracy or detail of a graphic display, expressed as dots per inch, pixels per line, lines per millimeter, or any other method."[3] However this concept is rather perceived as the more samples taken from a graph, the better its details are visualized. Dividing an image into more pieces and justifying each piece will give more details as the bigger number of pixels will share the task of identifying the image.

In conventional cartesian images, this can be understood easily. Here, no pixel intervenes the area that should be represented by another one and all the rectanglular image is spanned by all pixels, i.e. there is neither overlap in the "working area" of pixels nor any part of the image is left unrepresented. To give qualitative feeling for resolution, we can use the following approximation:

$$\text{Resolution} = \frac{\text{Number of pixels}}{\text{Total area}} = \frac{1}{\text{Total Area/Number of Pixels}} = \frac{1}{A} \qquad \text{Eq (2.1)}$$

where A is the unit area for a pixel (and could be the area of a LED used in a rectangular unspinning display as in Figure 2.1). As it can be easily seen, it is rather easy to to define a unit area for the rectangular image case. However the discussion of resoltion is not as easy with the spinning display because

- There are not as many pixels as the rectangular case to span all the image and the pixels are actually "in a rush" to span the rest of the image by moving very fast.
- There is not a constant unit area that can be defined as above because the areas spanned by LEDs residing on different radial distances differ as a result of rectangular motion.

However we can define another concept of resolution that is as close to the above as possible. This time resolution is not the measure of how well the details are observed but the measure of how "not-blurred" our resulting image is and how equally the radially displaced points are displayed. To put it in another way, we will accept that we have a good resolution if we can achieve a slightly blurred vision due to the persistence of the data from the previous position of the column of LEDs and slightly different areas represented by each LED. The former, at least, clearly agrees with our discussion above about the "overlap of working areas" and the latter with our desire to display everywhere with equal quality. We want to minimize

[3] answers.com

the difference of areas represented (spanned) by the near-center LEDs and near-edge LEDs (i.e. making the ratio of areas represented as close to unity as possible).

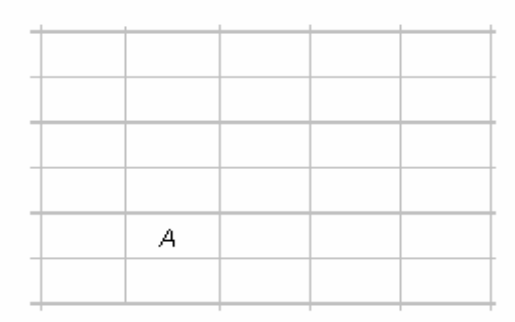


Figure 2.1 - A rectangular image with pixels and pixel unit areas well-defined

The above-developed measurement of $\frac{1}{A}$ for resolution is still valid here but as mentioned before it stands now for how "not-blurred" our resulting image is and how equal are these A's. For each of these qualifiers, we defined a criterion as below:

$$\text{Blur amount} = \frac{\text{Tangentially spanned length}}{\text{Original Tangential Length}} \qquad \text{Eq (2.2)}$$

$$\text{Area Inequality} = \frac{\text{Spanned area at one pixel}}{\text{Spanned area at another}} \qquad \text{Eq (2.3)}$$

As it can easily be seen from above that the blurring can be decreased by decreasing the radially spanned length and area inequality can be decreased by increasing the origina area. These concepts will be understood easily below.

2.1.1 Line (widthless) approximation for LEDs in radial direction

Although this approximation is unrealistic, it helps understand the nature of the spinning display and the concepts of spanned area and length with simplifying the mathematics. Please note that in this section and the next section, the LEDs will be assumed to have been placed with no gaps in between for the sake of simplicity. The ultimate result can be modified by replacing the height of a single LED with height plus the gap in between.

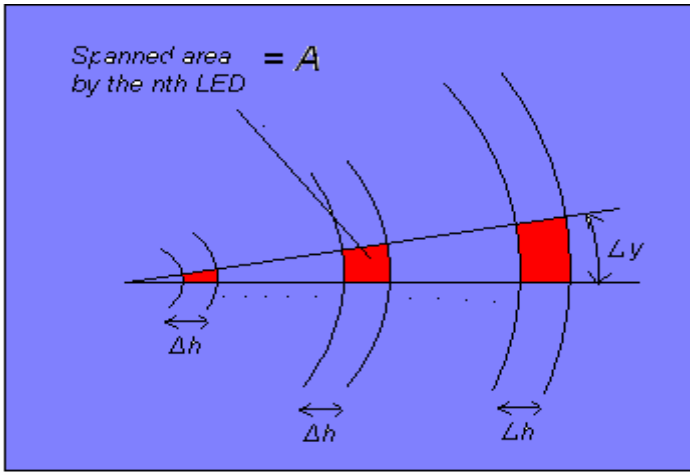


Figure 2.2 – Line approxiamtion

In Figure 2.2, the line approximation is illustrated. Δh is the height of an individual LED and the area spanned by the nth LED is,

$$A_n = r_n \Delta\theta \Delta h = n(\Delta h)^2 \Delta\theta \qquad \text{Eq (2.4)}$$

where $r_n = n\Delta h$. The Δθ here means the angle spanned during the time that the data on the LEDs did not have time to refresh. On the other hand, the tangentially spanned length is given by

$$\Delta y_n = r_n \Delta\theta = n\Delta h \Delta\theta \qquad \text{Eq (2.5)}$$

Since the widthless approximation for LEDs remain an original LED area of zero and original tangential length of zero, the equations of (2.3) and (2.4) are infinity for the worst case of comparing the outermost LED (say we have m LEDs) where $A_m = m(\Delta h)^2 \Delta\theta$ to the innermost LED $A_0 = 0$ and $\Delta y_m = r_m \Delta\theta = m\Delta h \Delta\theta$.

$$\text{Blur amount for worst case} = (m\Delta h \Delta\theta + 0)/0 = \infty$$

$$\text{Area inequality for worst case} = (m(\Delta h)^2 \Delta\theta + 0)/(0+0) = \infty$$

However, as mentioned already, this approach was unrealistic and thus we needed another approach as below.

2.1.2 Box approximation for LEDs in radial direction

In contrast to the widthless approximation, our LEDs now are considered as boxes with widths of Δb and height of Δh as before.

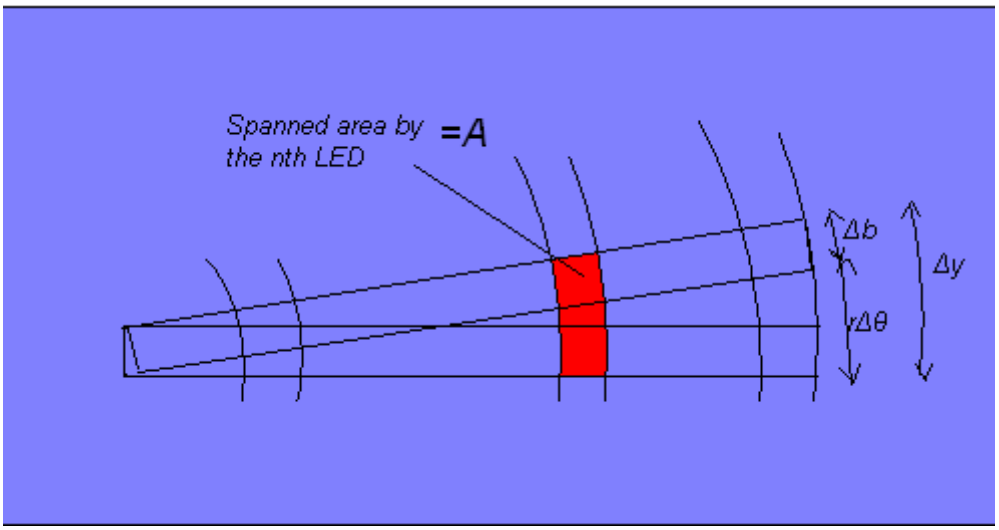


Figure 2.3 – Box aproximation

Although in Figure 2.3 the edges of the boxes used to model LEDs seem to perfectly overlap between the drawn circles, which is only valid for small angle approxiamtions, which is one reason of our selecting high rpms. For this new case:

$$\mathrm{A}_{\mathrm{n}} = r_n \Delta h \Delta\theta + \Delta b \Delta h = \mathrm{n}(\Delta h)^2 \Delta\theta + \Delta b \Delta h \quad \text{Eq (2.6)}$$

$$\Delta y_n = r_n \Delta\theta + \Delta b = n \Delta h \Delta\theta + \Delta b \quad \text{Eq (2.7)}$$

and we can easily deduce that

$$\Delta\theta = \frac{2\pi}{S} \quad \text{Eq (2.8)}$$

where S is the number of samples taken in one turn. This S will correspond to the number of columns in the polar image later.

Inserting these to the equations (2.2) and (2.3) will give more insight about the necessary precautions to be taken to prevent blurring and area inequality in the case of m LEDs:

$$\text{Blur amount for worst case} = (m\Delta h\Delta\theta + \Delta b)/\Delta b = 1 + 2\pi m \Delta h / S\Delta b \quad \text{Eq (2.8)}$$

$$\text{Area inequality for worst case} = (m(\Delta h)^2 \Delta\theta + \Delta b \Delta h)/\Delta b \Delta h = 1 + 2\pi m \Delta h / S\Delta b \quad \text{Eq (2.9)}.$$

It is really interesting to see that the worst case values of blur amount (which happens at the outermost LED) and the area inequality (which happens when we compare the outermost LED and the innermost LED assuming the latter lies at r=0) are equal. Beyond that, these two equations give us a hint about how to place our rectangular LEDs more efficiently. Since we want to pull these values as close to unity

as possible, Δh should be small and Δb should be big, dictating that the LEDs should be placed such that their short sides lie on the radial direction. We placed our LEDs accordingly.

2.2 Refresh rate and the operating rpm

Since we want to obtain a flicker-free image in our spinning display, we have to spin it faster than the human-eye response time. Normally, human-eye can distinguish the changes occuring at less than 10Hz, but after this frequency it perceives a continuous image yet with flickers until a certain frequency. This is different than the frame rate used to describe how many times the image on the screen can change. In movies, the frame rate is 24Hz but each frame is refreshed twice before getting to the next frame. This remains 48 transitions per second but since it is higher than the human response time about 5 times, the eye cannot see the transitions[4]

A good illustration of this problem was implemented in our lab. We conducted an experiment with a LED and a square wave generator. As the current in the LED was changed at a few Hz, the flickers were visible. The LED seemed to be on very brightly and then was turned off. As we increased the frequency beyond 10Hz, the transitions were even harder to perceive and there were no flickers above 20Hz. At this time, the LED did not seem to be as bright as we saw it at first because now the eye was automatically averaging the dark and bright periods. At this point we changed the duty cycle of the square wave and observed that we did not see longer bright and shorter dark (for duty cycles greater than 50%) intervals or the other way (for duty cycles less than 50%). Rather we observed a brightness change in the LED. This gave us an idea that we could use Pulse Width Modulation in order to obtain brightness thus grayscale control of our display.

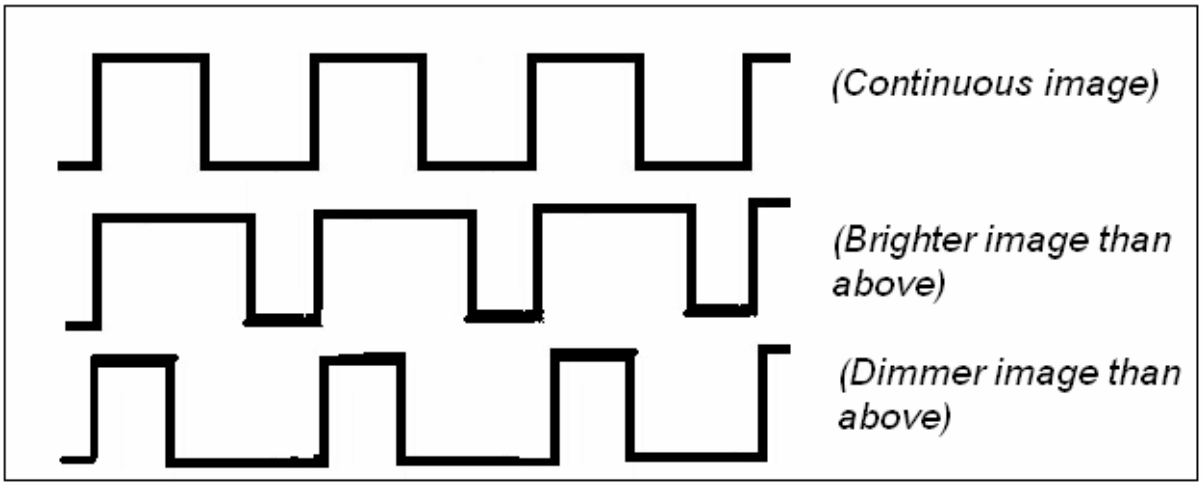


Figure 2.4 – Duty cycle vs observed brightness

[4] http://www.answers.com/topic/flicker-screen

Pulse width modulation (PWM) is a modulation technique that generates variable-width pulses to represent the amplitude of an analog input signal. In our case this analog input signal is the signal we are driving our LEDs with. This method is relatively easy to use due to its digital nature composed of on-off signals. The signal is adjusted to be high for a certain (and desired) number of clock pulses and low for the remaining pulses. When this is done fast enough as discussed above, this is perceived as a continuous brightness. For example if we want an 8 bit depth in color (or brightness if we are talking about grayscale), we have to divide one period of the square wave coming from the generator into 2^8 = 256 PWM clocks. For a moderate brightness, the duty cycle is 50% meaning that 128 "ones" and 128 "zeroes" are sent in order. For full brightness, the duty cycle should be 100% meaning that all 256 bits sent are "ones" (DC). The other duty cycles correspond to different brightnesses as shown in Figure 2.4.

To choose for the right rpm for our spinning display, we have to consider many factors as discussed in the previous section and in this section. From equations (2.8) and (2.9), we can see that the as the sampling rate in one cycle (S) increases, the blurring and area inequality decrease as desired. We decided to have as many samples as not to have one sample persist more than a maximum tangentially spanned length of Δb, the width of a pixel. In other words our samples should be taken so often that the outermost LED will persist at most as much as its width. This is a very harsh restriction on the refresh rate, thus the system clock, not to go below a certain value. For the case of m LEDs,

$$S = 2\pi(m\Delta h')/\Delta b = 2\pi m \qquad \text{Eq (2.10)}$$

where Δh' is the sum of the pixel height, Δh, and the gap between LEDs, Δg. In our case Δh' and Δb are the same (both 1mm) and cancel above. For a 16 LED case, S becomes 100, but this is the lower limit for it. We chose even a bigger number of samples, 300, for the sake of decreasing blurring and area inequality in equations (2.8) and (2.9) as the optimum case is infinity for S.

The sampling rate shoud not be confused with our refresh rate, which we should choose around 25Hz. We chose this refresh rate as 25Hz as in most movies.

Considering all these factors, we can now determine the frequency that the data on each LED are going to change. Think of a LED in a certain position displaying a certain brightness. This brightness should persist until the next sample is taken, which is 1/300 of the time it takes for one total cycle of the spinning display, i.e. 1/25 seconds. The result is that the data should change after (1/25)/300 = 1/7500 seconds (133 microseconds). This value corresponds to the period of the square waves shown in the figure for duty cycles and brightness. As we discussed before, for an 8 bit brightness control, the PWM that will implement this control should be 256 times faster than this rate. The period of PWM, thus, should be 133 us/256 = 521 nanoseconds. This corresponds to a frequency of 1.92 MHz for pulse width modulation. However it should be kept in mind that we have to keep this frequency

twice as high because we are high multiplexing the outputs of the RAMs. In the previous design for gray scale display, one RAM drove one set of eight LEDs. So, 1.92 MHz was enough for that operation. However to keep the number of RAMs low, we decided to switch the output of each RAM to the next set of 8 LEDs in the 16 LED row each time. Meanwhile the buffers will be enabled and disabled according to the predetermined order and will be in harmony with the switching RAM outputs. This results in a need for the frequency to be twice of the calculated value, namely 3.84 MHz. We are using 4 MHz clock frequency as the closest value.

Since 4 MHz is a very high frequency for most circuit components, it should be paid more attention. However the biggest problem seems inherent in the response time of buffers used to drive the LEDs with the data coming from the RAMs. But it is important to note that the PWM means a train of "ones" and then another train of "zeroes" and the frequency of 4 MHz will correspond to the clock generating these ones and zeroes. The actual frequency that the buffer will be working is much smaller than this because there will be only one transition from 1 to 0 every 133 microseconds and this effective frequency is around 1/133us = 7.5 kHz only. The other problem is that the microcontroller could not refresh the circuit that often since it took many intructions to do this task. Although we decided not to implement the circuit with a microcontroller, we have to process the image data so that it comes to the RAM in the form of train of "1"s and "0"s representimg PWM. This is done in Matlab as will be explained later in Chapter 4.

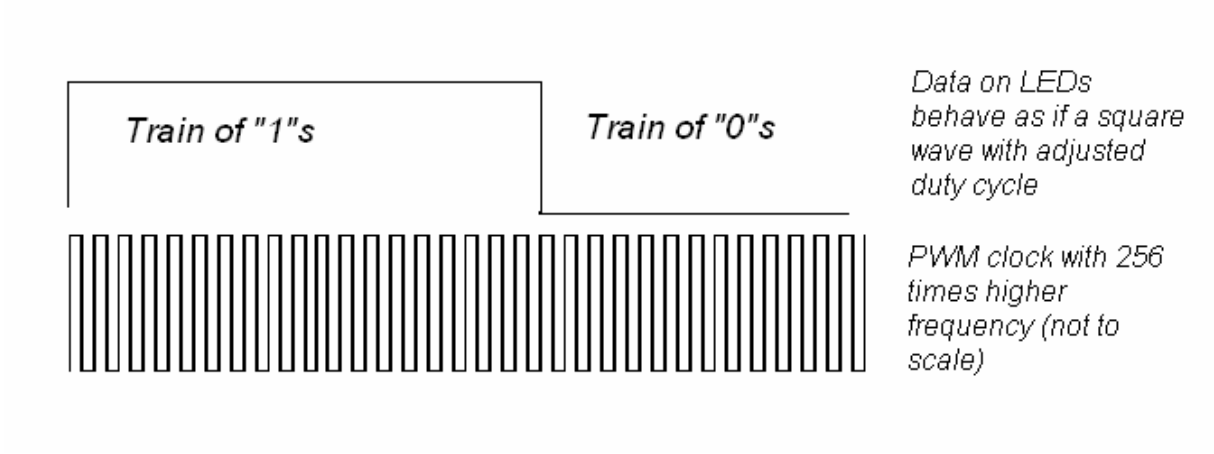


Figure 2.5 – Data on LEDS and corresponding PWM

2.3 Driving the LEDs

The LEDs have the following specifications:

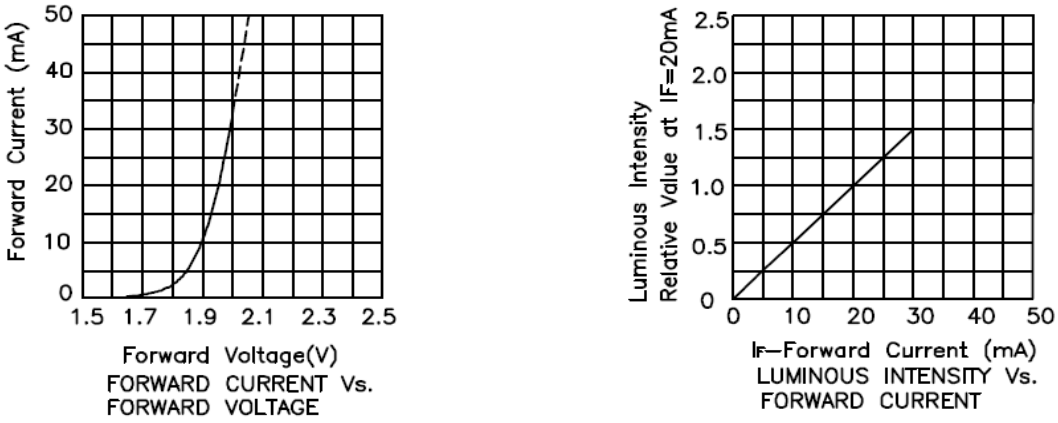


Figure 2.6 – Important LED characteristics

We are planning to operate our LEDs at 20 mA, which is also the rated current, for a reasonable luminance. This requires a 1.95 V voltage drop on the LED as seen in the first graph in Figure 2.6. We can model the LED as a resistor at this voltage under constant operation mode because the rest of the circuit sees this component as a balck box with a 1.95 V voltage under 20 mA current. The equivalent resistance is 1.95V / 20 mA = 97.5Ω ≈ 100 Ω. Considering a 5V supply voltage, the voltage divider that will remain a voltage of 1.95 V on the LED can be implemented as in Figure 2.7. Here, the 150 Ω is selected as it is the approximate solution to the following equation:

$$1.95 = 5\frac{100}{100+R} \qquad R \approx 156\ \Omega$$

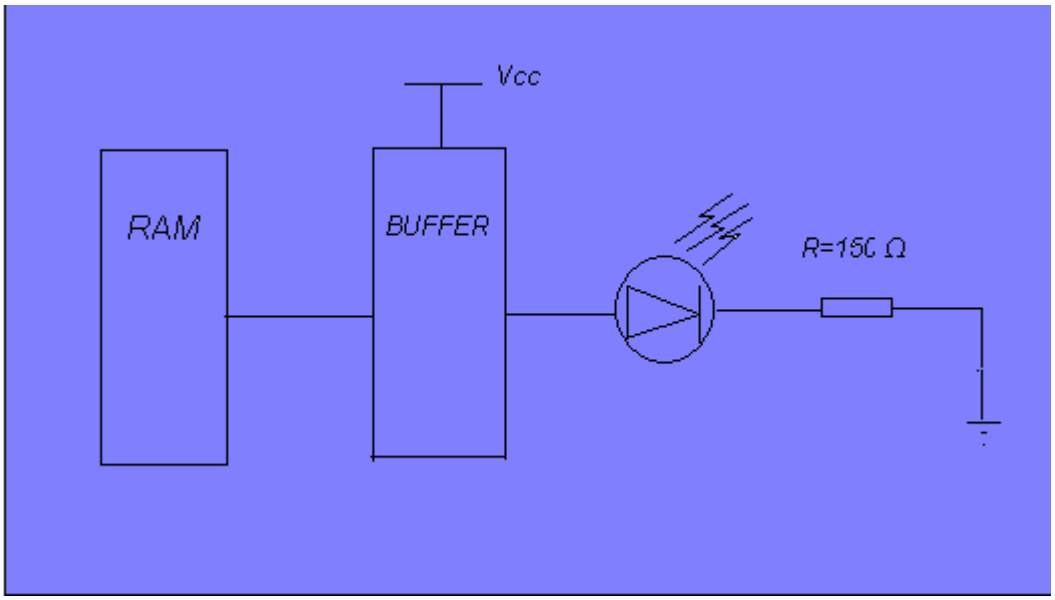


Figure 2.7 – LED driving circuit

Previous designs for LED driving circuits:

Before we unemployed the microprocessor, we decided to drive the LEDs with a transistor circuit using the BJT as a voltage controlled switch and the microprocessor output as the input voltage to the switch. The first system we tried was as follows:

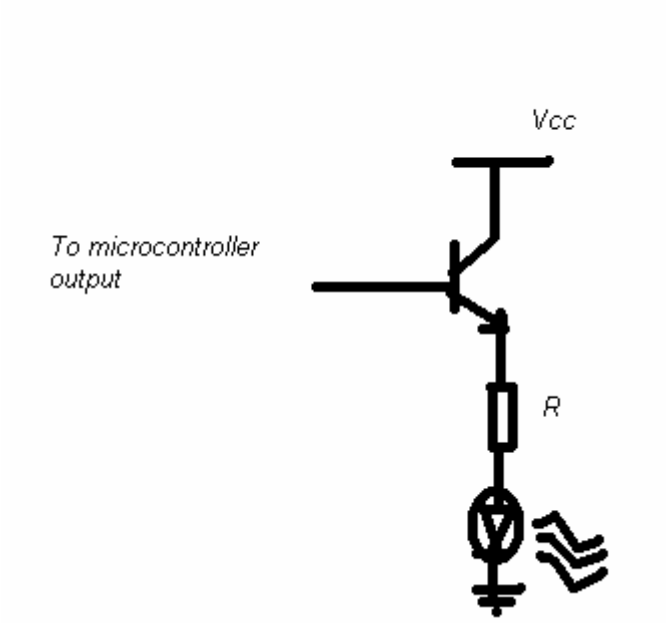


Figure 2.8 – First LED driving circuit tried in our project

The Spice simulation results with R=120Ω at 20 MHz remained the following current profile on the LED. 120 Ω was used because the LED specifications were not the same as our actual LEDs but was the necessary resistance to obtain 20 mA on the simulated LED model. This would just give a feeling about the circuit operation.

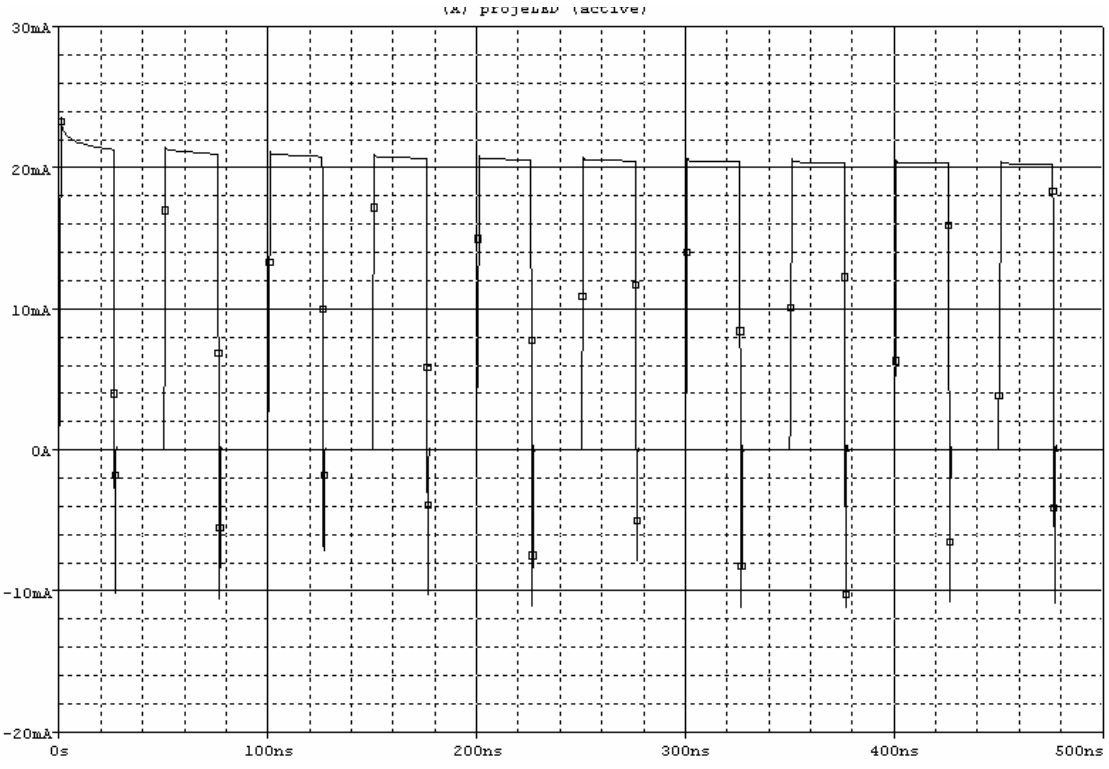


Figure 2.9 - Spice results for the first driver circuit

The above simulation was done at 20MHz because we needed very high frequencies to do Pulse Width Modulation on the LEDs turning with enormous speeds and refresh rates. This frequency is an upper bound and still gave good results. However the voltage on the LED did not seem to have time to fall to zero when the switch was open because of inner capacitances. This is not a problem since the LED would not be on without current and current successfully drops to zero when the switch is open. The main problem is that this circuit doesn't have a base resistance that will protect the base of the transistor. Also we need to operate the transistor on the edge of saturation to prevent excess charges for the quick response of switch at high frequencies. These accumulated to a new idea of LED driver, which was more successful at the lab trial:

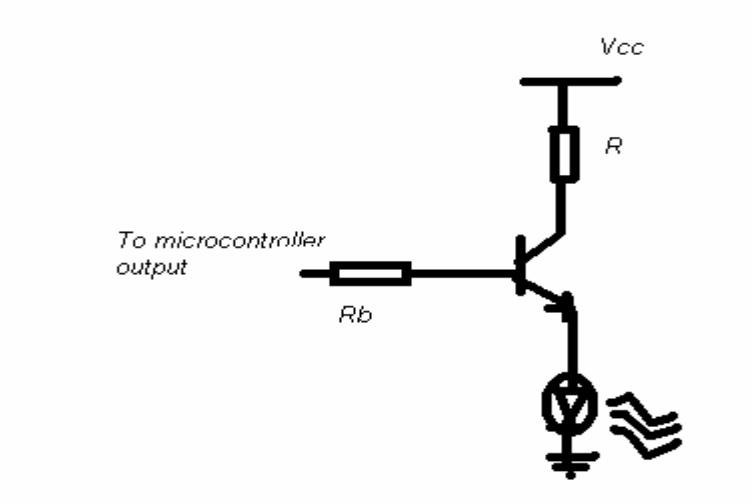


Figure 2.10 - The second configuration to drive LEDs

This second configuration provided the same good simulation results and additionally protection to the base. However, the β was measured around 25 in actual experiment and 82 in simulation, which are greatly different but all show we are in saturation region as expected. This difference may stem from different transistor parameters: We used in the lab BC547B but in simulation one of the CA3086 package parameters. However the results in the lab were not as perfect as Spice simulations because we conducted the experiment on the breadboard which was not suitable for high frequencies. In fact, the waveform was distorted unacceptably after frequencies beyond 2 MHz.

CHAPTER 3

Circuit Operation

We start with the block diagram of the circuit:

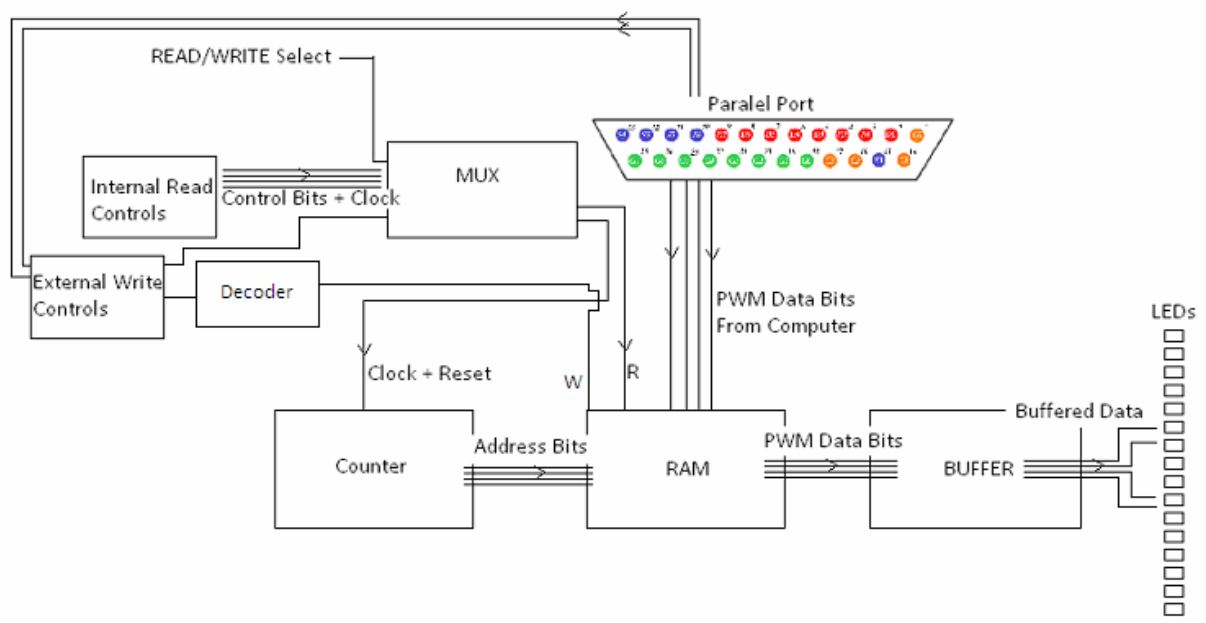


Figure 3.1: Block Diagram of the Circuit

The basic operation of the circuit is as follows:

Multiplexer selects the bits to control the counter from the internal controls and external controls. The RAMs are controlled by bits from both decoded external write controls and internal read controls. We have a Read/Write selector switch on the circuit so that when the circuit is in WRITE (high) position, the data coming from the parallel port is written to the RAM. When the switch is in READ (low) position, it means that the display will spin and the counter starts to count in the speed of internal oscillator so that the data will be taken off the RAM to create the image stored in it. Buffer is used so that the intense power needs of LEDs are served by the buffer instead of the RAM.

We should note here that the above figure and operation is a very simplified version of circuit operation. In fact, there are three groups of LEDs (red, green and blue) which are separately controlled by different RAMs and buffers. Also, one should think about the orientation of LEDs to understand the circuit operation. We have three columns of 16 LEDs which are separated 120 degrees apart. In each column, LEDs are placed with 2.1mm spacing from their centers on `r` direction. Spacing width originates from PCB and resolution specifications. Each column is placed 120 degrees apart. Lengths of LEDs are 0.5mm in `r` direction, 1mm in `θ` direction.

The following figure illustrates the design principle:

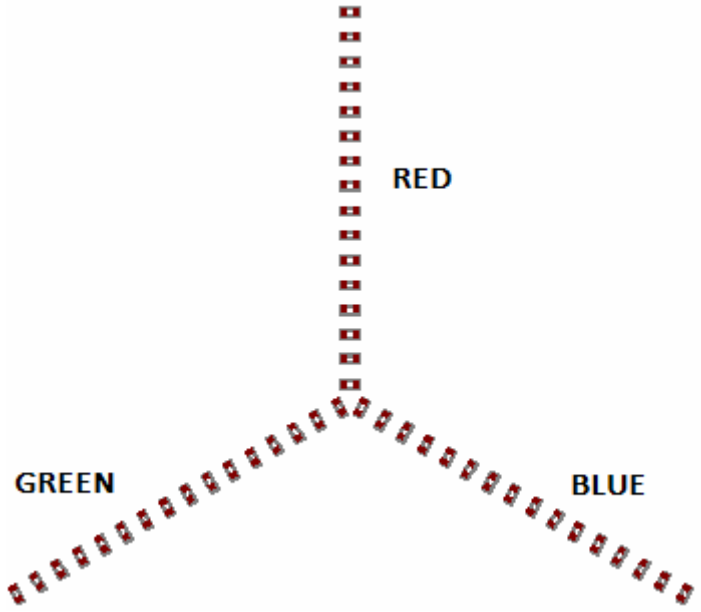


Figure 3.2: Location of RGB LED groups on the PCB.

The following sections analyze the circuit operation in detail:

3.1 Internal Read Controls

These controls are actually four controls on the circuit:

a) Internal Clock

Our device will spin at a constant angular velocity so that a constant image is displayed. Corresponding to this angular velocity, as discussed above in the *Led driving – Timings* section, we will have a pixel time. The pixel time will be divided into 256 PWM modulation cycles and this speed corresponds to our data transfer speed. This constant speed is provided by the internal oscillator on the circuit, which oscillates at a specific frequency. Counter increments using this oscillation when the device is in READ condition. This continues to next spin of the display until counters are reset by the Infrared Sensor. The internal clock is selected by the multiplexer when the circuit is in READ position.

b) Infrared Sensor

When one spin of the device is completed, the counters should be reset so that the data starts from the beginning of the image. Infrared sensor senses the infrared wave coming from the hole over the back PCB and instantly resets the counters when in read position. It will be explained in detail in the motor control chapter because same sensor is also used for motor speed control. The infrared sensor is selected by the multiplexer when the circuit is in READ position.

c) RAM 1-2-3 Chip Select, Output Enable and Write Enable bits

In read condition, the RAMs should be always on and they should have their output enabled so that the device outputs a continuous image. Using this idea, Chip Select bit is directly connected to the ground, which sets the RAM always on and Output Enable bit is connected directly to READ/WRITE selection switch, which is always low (output enable active) when the circuit is in READ position. Write Enable

bits of the RAMs are not controlled by these bits and WE overrides OE operation if it is low. We should note here that, unlike previous designs, multiplexer is not used for selecting RAM controls.

3.2 External Write Controls

As seen from their name, these four controls are external to the device; they are connected to the control bits of the parallel port. These data bits are prepared by the software interface over a computer and fed to our device. They consist of four control bits from parallel port. They are selected only when the circuit is in WRITE condition.

a) External Clock

The circuit still needs a clock to control its operation when it is in WRITE position, changing addresses of the data to be written on the RAMs. External clock is supplied by one of the parallel port bits and it is fully synchronized with the data bits coming from the parallel port. Its speed depends directly on the computer it is connected to and we achieved approximately 33 kHz clock speed using a parallel port of a typical computer[5].

b) External Counter Reset

We have two RAMs and we want to write different data to them starting from their first address. Before writing data to each of them, the counters should be reset. This job is done by the counter reset bit coming from one of the parallel port bits and this bit is selected by the MUX when the circuit is in WRITE position.

c) RAM 1-2-3 Chip Select, Output Enable and Write Enable bits

As we said before, the data will be written to the RAMs one at a time. We have three RAMs, so we should open one for writing while the others are not active. We use Write Enable bits for this job, which can control the Write cycle of the RAMs. It will be given in more detail in RAM section. The Chip Select bit is directly connected to the ground, which sets the RAM always on and Output Enable bit is connected directly to READ/WRITE selection switch which is high (not active) when the circuit is in WRITE position.

3.3 Multiplexer

The multiplexer in our circuit is a 4bit 2-to-1 multiplexer and has a part number 74HC157. It has its select input coming from a physical switch on the board which tells the multiplexer if the device is in READ or WRITE condition. As previously stated, if the circuit is in READ condition, the two internal read controls are electrically connected to the output and if it is in WRITE condition, the two external write controls are connected to the output. The multiplexer basically gives the control of the counter to the internal circuits or external computer.

3.4 Counter

In the operation of our device, 8-bit Pulse Width modulated data is used to drive the LEDs. The number of bytes to be stored in the RAMs is 225kilobytes each. So,

---

[5] The tests were conducted on a Pentium 4 – 2.8GHz computer with an MSI 865PE motherboard.

we needed an 18 bit counter. Since an 18 bit counter is not easily found, we used two 74hc4040 counters and cascaded them. This allowed us to realize a practical 18 bit counter. The cascading operation is done by connecting the most significant bit of the first counter to the clock input of the second counter.

The two counters work in both READ and WRITE states. In READ state, they are incremented by the oscillator in the circuit and reset by the infrared detector. In WRITE state, they are incremented by a control bit from parallel port and reset by another pin.

3.5 Decoder

In our latest design, we have three RAMs and they should be controlled separately during WRITE operation. This requires at least three bits, and if we add the two bits used for controlling the counter, we need five bits for external control. Unfortunately, besides 8 data bits, parallel port supports only 4 control bits. This requires a decoding operation to control the RAMs. At this point, we decided to use a decoder to decode two bits coming from the parallel port. The fact that only one RAM is active at a time during writing gave us this opportunity. So, there are basically four states of RAMs represented by two bits:

1. RAM 1 is on for writing, the others are off.
2. RAM 2 is on for writing, the others are off.
3. RAM 3 is on for writing, the others are off.
4. All RAMs are off.

The decoder (74LS139) has a functional description like the following:

TRUTH TABLE

INPUTS ENABLE SELECT			OUTPUTS			
$\overline{E}$	A1	A0	$\overline{Y3}$	$\overline{Y2}$	$\overline{Y1}$	$\overline{Y0}$
0	0	0	1	1	1	0
0	0	1	1	1	0	1
0	1	0	1	0	1	1
0	1	1	0	1	1	1
1	X	X	1	1	1	1

X = Don't Care, Logic 1 = High, Logic 0 = Low

Figure 3.3: Decoder truth table

The first three outputs of the decoder control the RAM as will be explained in the next section.

3.6 RAM

We store the PWM data to light the LEDs in three 512Kx8 bit Low Power CMOS Static RAMs with device number KM684000BL-TI. Each RAM has 8-bit output and each output directly corresponds to one LED. The problem we faced with this RAM was that its input and output pins were the same. So, we needed to make sure that the paths that fed the RAM with data in WRITE position does not interfere with the READ operation. To overcome this problem, we used a buffer for each RAM with part number 74HCT541 before the data from the parallel port enters the input of the RAM:

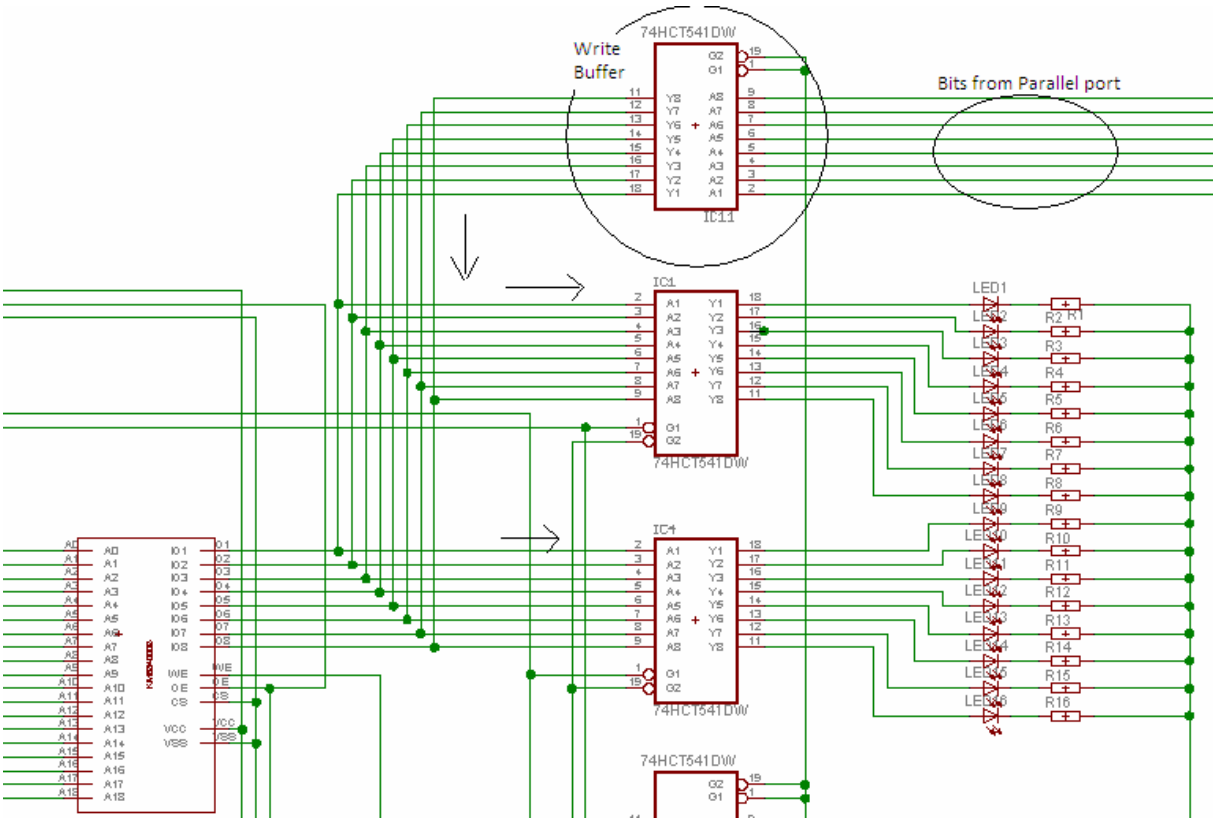


Figure 3.4: RAM and Buffer

These buffers are on when the switch is in WRITE position and when the circuit is in read position, their outputs are high impedance.

We can divide RAM operation into two main sections, and we will analyze them in detail:

3.6.1 Ram operation while reading

When the spinning display does its display operation, RAMs continuously feed buffers with the data for corresponding LEDs. As the device rotates, counters are incremented automatically and they change the address of the RAMs to be read. In the last version of our circuit, each RAM drives 16 LEDs and since the RAMs have only 8 outputs, a time multiplexing was required. To achieve this time multiplexing, we connected two buffers at the output of each RAM and made sure that only one of them is active at one read cycle. In this scheme, even addresses of the RAMs store data for the first 8 LEDs, and odd addresses of the RAMs store data for the second 8

LEDs. In even cycles, the second 8 LEDs are not lit, and the opposite for second 8 LEDs.

Besides time multiplexing, we had to control the RAMs so that they are open for data output in read cycles. To achieve this operation, we first needed to analyze the RAM functional description:

$\overline{CS}$	$\overline{OE}$	$\overline{WE}$	I/O Pin	Mode	Power
H	X[1)]	X[1)]	High-Z	Deselected	Standby
L	H	H	High-Z	Output disbaled	Active
L	L	H	Dout	Read	Active
L	X[1)]	L	Din	Write	Active

1. X means don't care.(Must be in low or high state.)

Figure 3.5: RAM functional description

As we see above, RAM has three control bits. In READ state, we selected to use:

CS = L, OE = L, WE = H

Our existing design used CS bits for controlling RAM operation. In that case WE and OE bits were connected as inverted version of each other and they interchanged bits as we go from the read mode to write mode. In our new design, instead of the CS bit, we use WE and OE bits. Our CS bit is directly connected to ground now. So, our RAM is always in active operation. The WE bits of the RAMs are directly connected to decoder outputs respectively. So, they are always high when the circuit is in READ position because the decoder is not active, hence all of its outputs are high. The OE bits of the RAMs are connected to the Read/Write switch on the board, i.e. they are low when the circuit is in read position and high when the circuit is in write position. So, we can achieve READ operation when the circuit is in READ operation.

With the above method, we were assured that the RAM will be in a READ operation, but the problem was to control write buffers so that their outputs are in a high-impedance state. We examined 74HCT541 function table:

FUNCTION TABLE

INPUTS			OUTPUT
$\overline{OE}_1$	$\overline{OE}_2$	A_n	Y_n
L	L	L	L
L	L	H	H
X	H	X	Z
H	X	X	Z

Notes

1. H = HIGH voltage level
   L = LOW voltage level
   X = don't care
   Z = high impedance OFF-state

Figure 3.6: 74HCT541 function table

We see that the high impedance state is achieved when one of the OE bits are high. We connected both of them together to the inverted version of READ/WRITE switch so that when the circuit is in WRITE mode, the buffers are on, else off.

3.6.2 Ram operation while writing

When the parallel port connector is connected and the device is in WRITE mode, RAMs should take the data that is coming from the parallel port. In this state, the controls bit should be in the following state:

CS = L, OE = H, WE = L

We have previously explained that the OE bits of the RAMs are connected to the Read/Write switch on the board, i.e. they are low when the circuit is in read position and high when the circuit is in write position. The CS bits were connected directly to ground. The WE bits are used to control the writing cycle.

From the parallel port, we take two bits to control the operation of the RAMs. From these two bits, we derive four states of the RAMs using the decoder as explained in the decoder section. The first three decoder outputs are connected to the WE inputs of the RAMs respectively. They are on when there happens a write operation on the corresponding RAM.

The details of this cycle will be explained in detail in the communication part of the report. But the point is, the write cycle is controlled by WE input of the RAM using parallel port bits.

3.7 Buffer

The RAMs output the data suitable for our LEDs, but the problem is they cannot drive the LEDs due to their low resistance. To achieve proper driving of LEDs at 20mA, we decided to use two buffers after the RAM. The part numbers of these two buffers are 74HCT541, the same as the above explained write buffers.

Also, the addition of extra colors, hence extra LEDs required us to make changes in the circuit. If we think about existing scheme, where one RAM drives 8 LEDs, we should have used 6 LEDs to design the new circuit. That was both expensive and required a very complex circuit if we think of the problems we faced about the RAMs in previous PCB design. So, we decided on using a total number of 3 RAMs, each driving LEDs of one color. The 8 data outputs of the RAM will be shared among the inputs of 16 LEDs in a time division multiplexing manner.

These buffers perform time division multiplexing using the principle that even addressed data of the RAM is only delivered to the first 8 LEDs by the first buffer and odd addressed data of the RAM is only delivered to the second 8 LEDs. The buffers have two output enable bits. To open the buffers, both of them must be in low state. One of them is connected to the first output bit of the counter (inverted version for the second buffers), so it changes in each cycle. The other bit is connected to a debug switch on the circuit, which opens or closes the output totally.

In each cycle, 24 of 48 LEDs will be receiving data. To achieve this time multiplexing, we decided to use buffers because:

1. We already had 3 buffers for driving the LEDs, and we would need them irrespective of multiplexing because only they could drive the current hungry LEDs.
2. They had an easy way of multiplexing; their chip select bits gave us the means of enabling / disabling them easily.

The resulting part of the schematic was like:

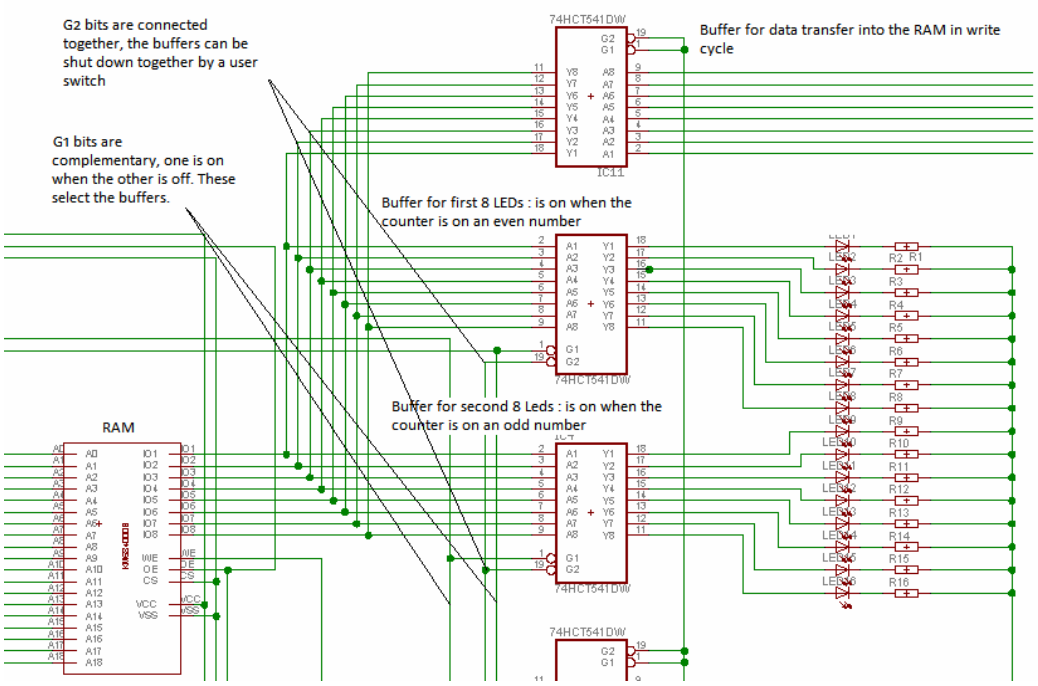


Figure 3.7: RAM and buffers for first LED column

We see another buffer at the top, which is on when the circuit is at write position; hence it loads the RAM with the data to be taken from the parallel port.

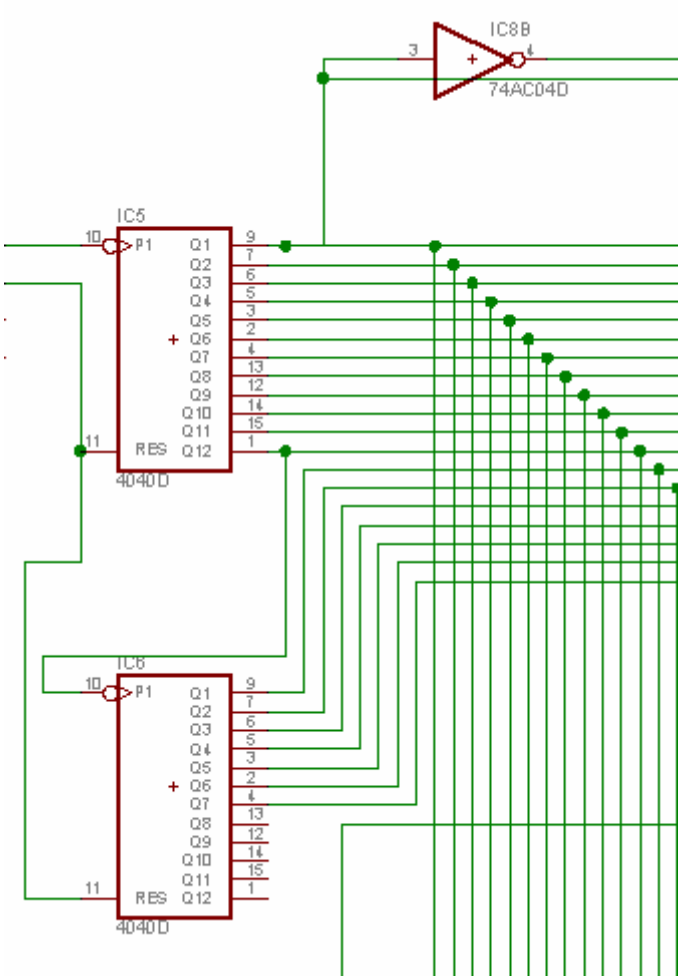


Figure 3.8: Counter first bit and its complement

We see in the above figure that the first bit of the counter and its complement are taken to the further parts of the circuit. Actually, this first bit goes directly into G1 input of first, third and fifth buffers. The other buffers' G1 bits are connected to the complement of the first bit of counters. G2 bits of buffers are connected together and they are taken to the physical switch on the circuit which is also new in this design.

3.8 Parallel Port

One of the most important parts of the spinning display project is obviously to transfer the data to the spinning part. There existed so many ways to realize this objective such as USB, wireless modules etc. We negotiated about which transfer method is most efficient and convenient for our purpose, eventually we decided to use the parallel port which is an inexpensive and yet powerful platform for implementing projects dealing with the control of real world peripherals. Brief information about parallel port is given below:

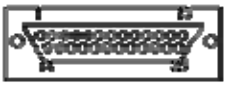
A **parallel port** is a type of socket found on PCs for interfacing with various peripherals. It is also known as a **printer port or** Centronics port. The IEEE 1284 standard defines the bi-directional version of the port. The parallel port, as

implemented on the PC, consists of a connector with 17 signal lines and 8 ground lines. The signal lines are divided into three groups:

- Control (4 lines)
- Status (5 lines)
- Data (8 lines)

As originally designed, the Control lines are used as interface control and handshaking signals from the PC to the printer. The Status lines are used for handshake signals and as status indicators for such things as paper empty, busy indication and interface or peripheral errors. The data lines are used to provide data from the PC to the printer, in that direction only. Later implementations of the parallel port allowed for data to be driven from the peripheral to the PC.[6]

Pin, port assignments and an image showed the Parallel port connectors are given below[7]:



View is looking at
Connector side of
DB-25 Male Connector.

Pin	Description	
1	Strobe	PC Output
2	Data 0	PC Output
3	Data 1	PC Output
4	Data 2	PC Output
5	Data 3	PC Output
6	Data 4	PC Output
7	Data 5	PC Output
8	Data 6	PC Output
9	Data 7	PC Output
10	ACK	PC Input
11	Busy	PC Input
12	Paper Empty	PC Input
13	Select	PC Input
14	Auto Feed	PC Output
15	Error	PC Input
16	Initialize Printer	PC Output
17	Select Input	PC Output

Pin Assignments

Note: 8 Data Outputs
4 Misc Other Outputs

5 Data Inputs

Note: Pins 18-25 are
Ground

Figure 3.9: Pin Assignments

[6] http://www.fapo.com/porthist.htm
[7] http://engr.nmsu.edu/~etti/fall96/computer/printer/printer.html

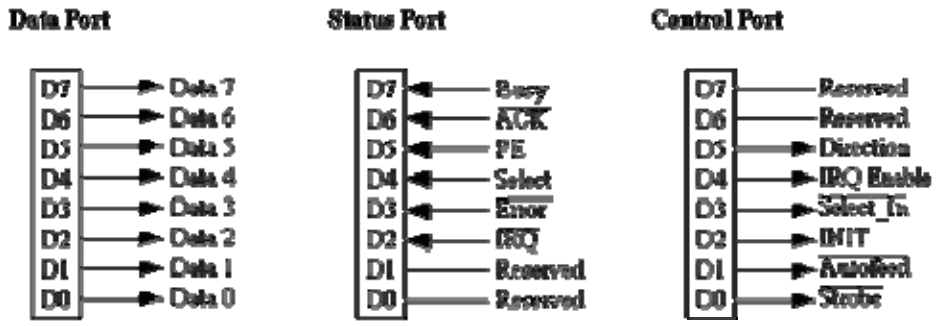


Figure 3.10: Port Assignments

Figure 3.11: Parallel port at computer side[8]

We used 8 data bits of parallel port to transmit the data which is then written on the specific address of the RAM determined by counters in the circuit. Also for Counter Increment, Counter Reset, RAM control bits and receiving data from the board we needed five another pins. For this purpose *Strobe*, *Autofeed*, *Init*, *Select Input*, *ACK* pins of the parallel port are used. Assignments of pins in our design are given below.

D0-D7 Data pins → Transmitting the data of image

Strobe → Counter Increment

Autofeed → Counter Reset

Init → RAM control data 1

Select Input → RAM control data 2

ACK → Receiving data from the board

In addition to these 8+5 pins a random ground pin is used for a referenced ground. We decided to use the original connector for the parallel port for perfect transfer.

Another important point about parallel port connector is that when connector is ejected to circuit, circuit pins which are directly connected to the parallel port pins becomes floating. That is to say that without parallel port pin connection they become ambiguous inputs. It is an unreliable case obviously. To prevent the harmful

[8] http://computer.howstuffworks.com/parallel-port1.htm

effect of this situation to the circuit, 10MOhm resistances are connected between parallel port pins at the female connector and ground of the spinning board. Thus when parallel port is connected to the circuit this situation can not affect the operation since 10MOhm is a high resistance. And when we disconnect it, these floating inputs are grounded.The schematic of the design can be found at the end of the report.

3.9 Oscillator and Infrared Receiver

The infrared detector on the spinning PCB was expected to send a reset signal through a multiplexer to the counter which stimulates the RAMs with its outputs. On the other hand, the oscillator was expected to create oscillations that would stimulate the counter. However this system had many problems in the first design, so improvements were necessary.

The first improvement was done to the oscillator circuit. We tried to run the oscillator circuit at 14 MHz, which was well beyond our needs. Before constructing the old PCB, we set the following circuit on a breadboard:

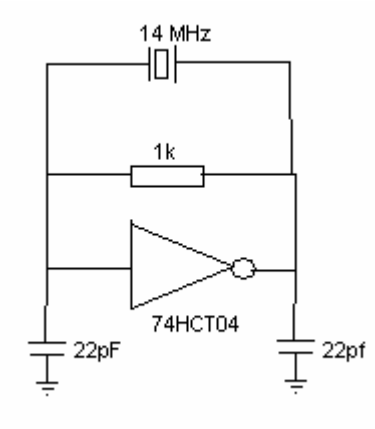


Figure 3.12: Previous Oscillator Circuit

Normally, a very high resistance is placed on the feedback path of the oscillator, which is parallel to the crystal oscillator. However since this was a very high frequency to work on a breadboard which has many parasitic capacitances, we had to decrease the equivalent impedance of the capacitive effects by placing a resistance as small as 1k to this path. This would obviously decrease our gain and the output will not be actually from 0V to 5V. This circuit worked on a breadboard with a distorted waveform as expected but still at 14 MHz.

When we set this circuit up on the PCB which has far less parasitic capacitive effects, the circuit did not give a 14 MHz waveform as expected. We first tried to change the crystal oscillator and use a 4 MHz sample. This still did not work and the output waveform was oscillating between 1.4V and 1.8V. Then we tried to increase, even open circuit the resistive feedback component because we did not need to

decrease those capacitive effects as we did on the breadboard. The new design was as follows:

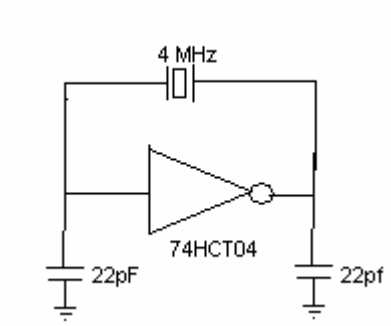


Figure 3.13: New oscillator circuit

This new configuration worked successfully on the PCB. When we understood our mistake, we tried the 14 MHz oscillator again and it was working also with the above configuration. However we decided such a high frequency was unnecessary for us now and we remained with 4 MHz for the previously discussed reasons.

The second fault in the circuit was in the counter. Even though we discarded the clock path coming from the oscillator circuit (through the multiplexer) and gave an external clean square wave as input, the counter seemed not working. We checked the reset input and it was inadvertently high. We pulled this node to ground and checked the operation again, but it was not working. We understood that the counter was erroneous and when we changed the counter with a new one, it counted successfully until we reset it manually.

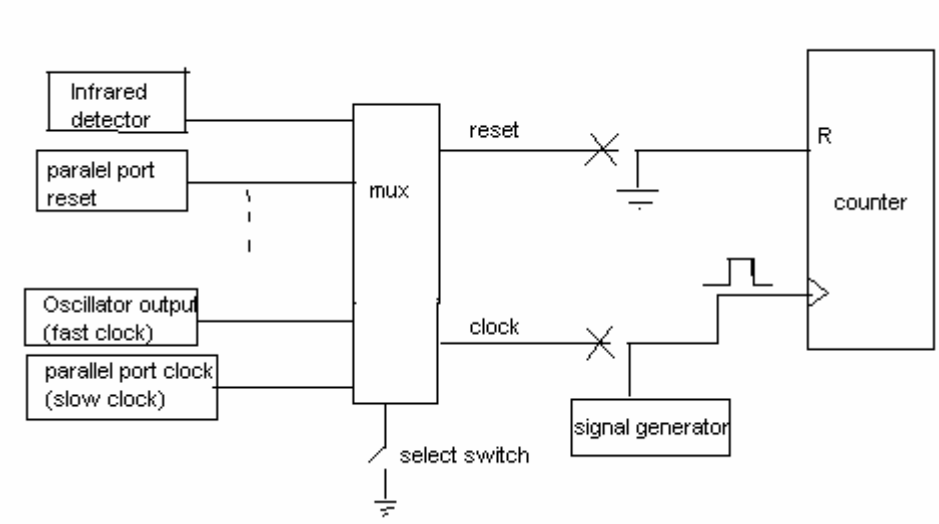


Figure 3.14: Trying the counter operation

The other improvement was done to the infrared detector. The infrared output was erroneous and we had to establish the circuit on the breadboard from scratch. The example application of our infrared detector was as follows:

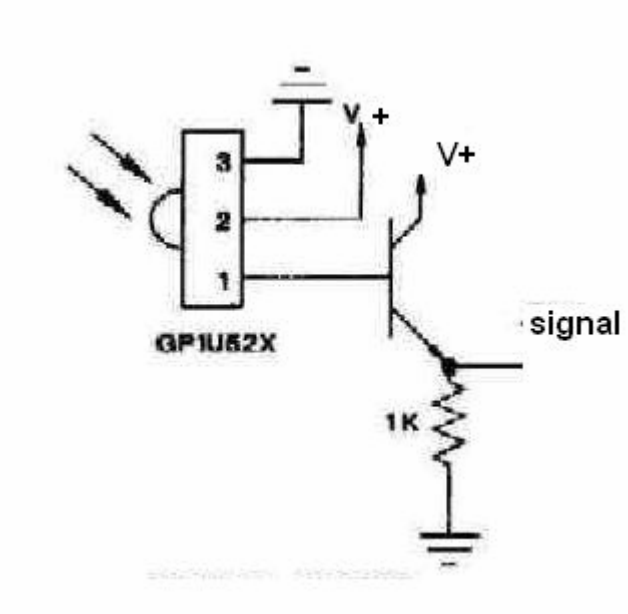


Figure 3.15: Example usage of infrared detector

When we established the above circuit, the signal at the output was as weak as 165 mV. We thought the reason for that is the infrared detector was not that powerful to compensate a base current as high as Vcc/β(1k). Thus we increased the emitter resistance step by step to 1M and we saw that every time we increased the resistance, the output signal was closer to 5V as expected. The output value at 100k was 3.6V and at 1M 4.4V. We decided that this was enough for our purposes of resetting the counter and we are using a 1M resistance now.

The infrared output level was maintained however the stability of the circuit was not maintained yet. The detector gave an output voltage of 0V when the base was exposed to 40 kHz infrared radiation. It should have remained at 5V (4.4V) when the base of the detector did not match the infrared LED. However this was not the case: The detector was very sensitive to the variations in the environment such as moving around it, turning down the lights and on again, abrupt changes in the environment enlightening, etc. Later we found out that some of these actions caused the power supply to have fluctuations and these caused undesired reset signals. We overcame this problem by placing a capacitance between the power supply of the detector and the ground. However there was not much we could do about the abrupt changes in the environment light level changes because this corresponds to an impulse at the base which includes all frequencies and of course the critical 40 kHz. This may not be a big challenge because our infrared will work mostly isolated from environment enlightening and will not be in much contact with our LEDs while spinning. The resulting working configuration of the infrared detector is as follows:

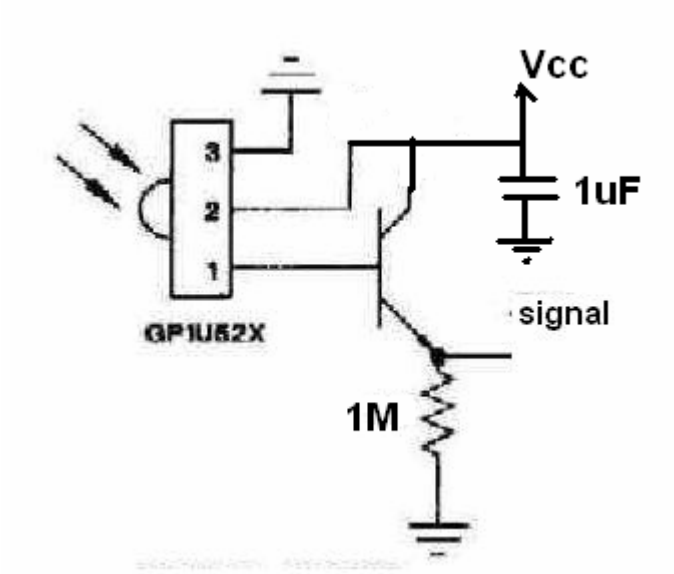


Figure 3.16: New configuration for infrared detector circuit

The infrared LED that is driving this sensor affected it not just from opposite but from a perspective. This caused a higher reset time than expected and needed. To prevent this, we put a tube between the two PCBs. This pipe would behave as if a waveguide that collects the infrared waves just when the spinning PCBs are passing opposite the stabilized infrared circuit. Another important point here was to put this hole far from the center of the PCBs because the same sized pipe would stay longer if front of the infrared LED and collect "resetting" waves longer than necessary. This is inherent in the nature of the spinning bodies that the same arc length is spanned in less time if it is further from the center. The new PCBs are ordered after considering this fact.

We made a demo of how the circuit resets via this sensor when we are spinning the display. We mounted the main PCB on the motor and then to the cage platform. We established the infrared LED circuit on the inner wall of the platform. The pipe indeed helped the circuit reset just when the sensor is passing right opposite the LED but nowhere else. This can be used successfully in the last version of the LED.

The last improvement was done on the RAM-buffer connection as discussed before. Since we are now trying to drive a column of 16 LEDs (each set for one color of RGB) with RAMs of 8 outputs, we have to drive the first 8 LEDs in the first clock cycle, then the second 8 bits in the next clock cycle and it goes on like this. When the first 8 LEDs are driven, the first buffer is enabled to drive the corresponding LEDs. The same is true for the second 8 LEDs. This configuration decreases the number of RAMs from 6 to 3, however it multiplies the necessary clock frequency by 2 because of high multiplexing. When the RAMs are being loaded data from the parallel port, now we need a demultiplexer to enable the RAM we want our data to be written to. The demux inputs are connected to ground via high resistances after the parallel port connection is removed. The new algorithm to write the data to the RAMs is described in detail in the software section.

CHAPTER 4

SOFTWARE USED

4.1 A different approach to the spinning image: Polar Coordinates vs Cartesian Coordinates:

Since the spinning phenomenon is closely related to polar coordinates, it would be efficient to work in polar coordinates rather than Cartesian coordinates. Our LEDs will be lit proportional to the values of the pixel values on the gray scale image. However it would be difficult to determine which values should be used to light the LEDs when it is a spinning array of LEDs that are trying to sample a Cartesian image. We should have used complex algorithms to determine the right value of the underlying pixel(s). Below is an example of the illustration of this problem. Can you or your processor decide the best value to display in any of the two LEDs shown in yellow? The unaligned LEDs contact with more than one pixel at a time and with different overlap ratios. The best value would be the weighted sum of all the pixels in contact to the LEDs, which will be considered in our proposed algorithm but difficult to implement in Cartesian coordinates:

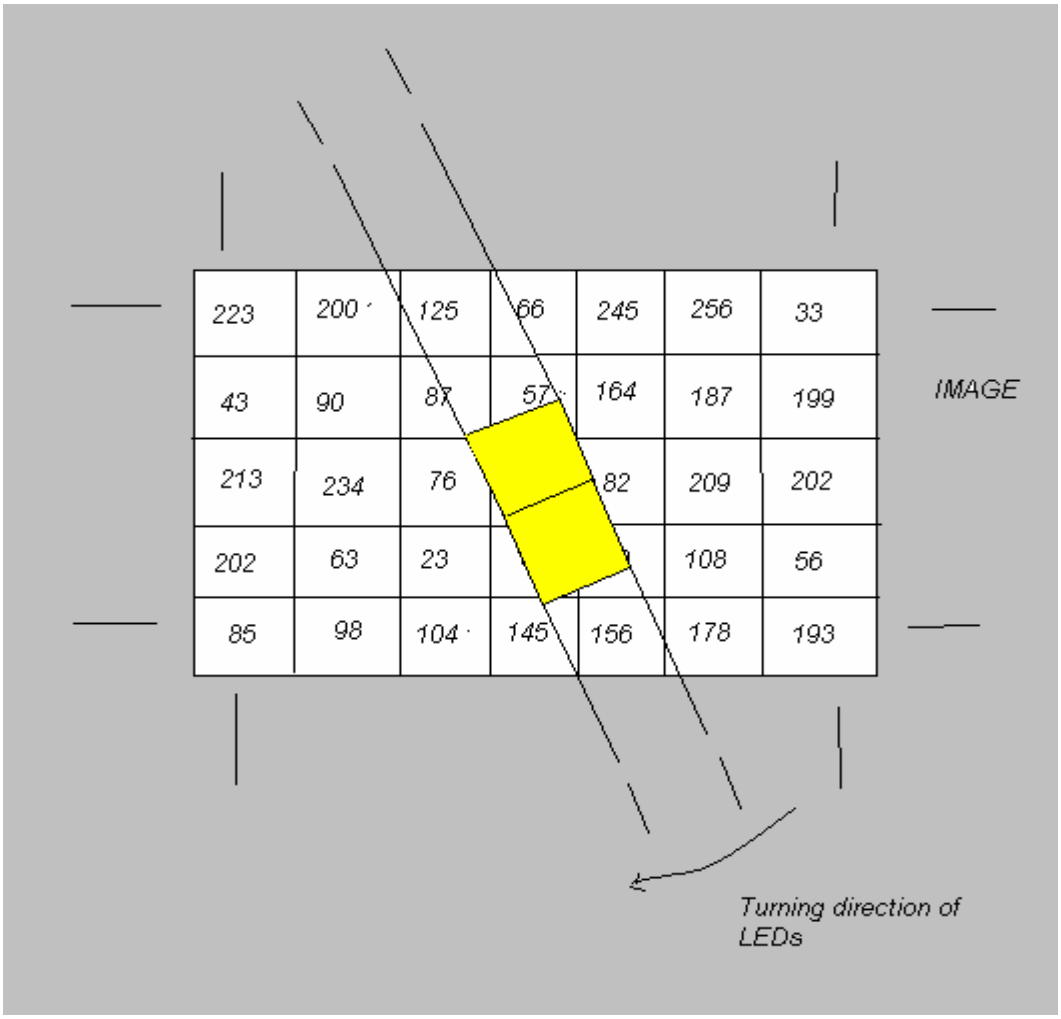


Figure 4.1- The illustration of misalignment of turning LEDs with stationary pixels

To overcome this problem, we decided to use the weighted sum of pixels idea, however we decided to live "in the same reference system" with the LEDs. Since the LEDs are moving linearly according to and their paths are easily defined in the polar coordinates, we transformed the image itself into polar coordinates doing interpolation in Matlab for the conversion. An example of a Cartesian image and its polar counterpart are below:

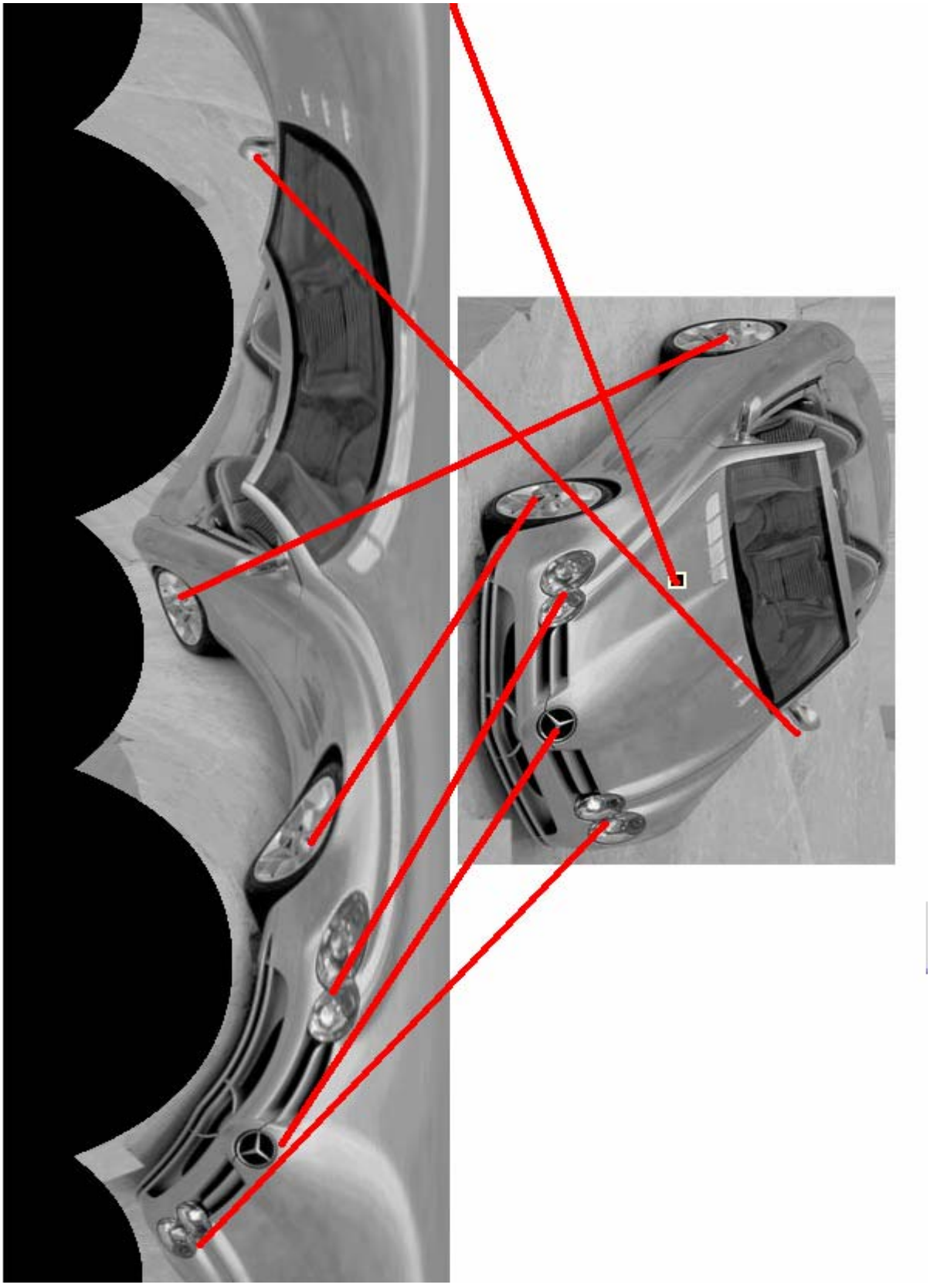


Figure 4.2- Cartesian –Polar conversion

The components of the car are shown in both images. The algorithm is like this:

- Go to the center of the Cartesian image
- Draw as many rays as you want going out of the center and spaced with equal angles. (In the above case 900 sampling rays were taken)
- For each ray, which will constitute a column of pixels in the polar image, calculate the values of each pixel for each LED by interpolation. (In the above case, linear interpolation was used)
- Now the problem of spanning the rectangular image with a spinning LED column reduced to moving the column of LEDs from left to right in the polar image (which assumes the LEDs will turn in the counterclockwise direction in the above example).

The black regions on the bottom edge are the points where our LEDs will be off because more LEDs (more rows in the polar image, each LED resembling a pixel) were used than the actual image size and those LEDs are redundant in this case.

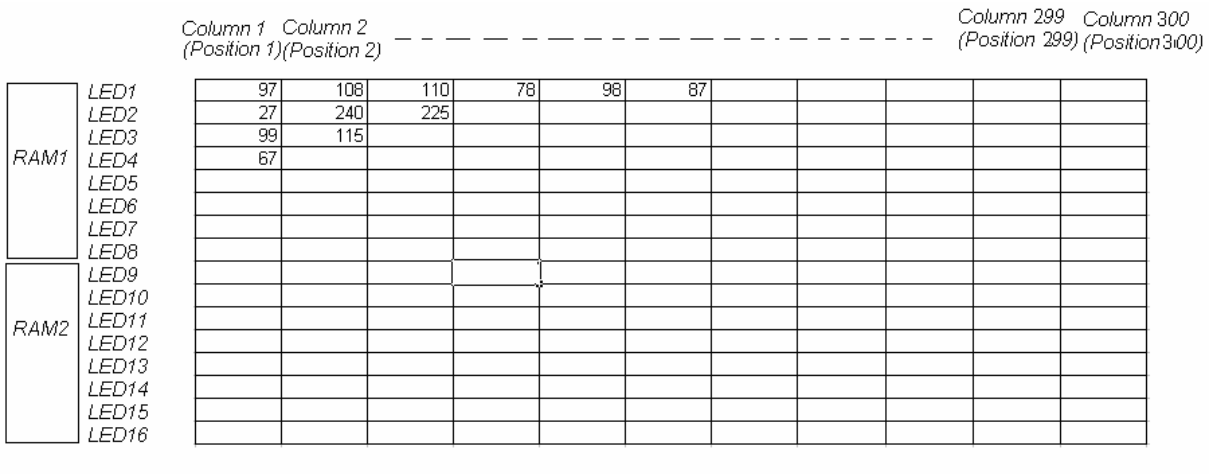
4.2 Manual PWM in Matlab

Previous algorithm for gray scale:

Matlab's task is not over with the above discussion. As already mentioned in the other chapters, PWM clock would be working at a very high frequency, so creating the necessary train of "1"s and "0"s for the LEDs would be a very difficult task for any circuit component. Thus, this task is done manually before the data are sent to the RAMs.

The data hierarchy in the RAMs for 16 LEDs and S=300 samples is shown in the next two figures. The algorithm for implementing PWM in Matlab manually will be as follows:

- Convert the image to polar coordinates so that it has 16 rows (the first 8 rows go to the first RAM and the second 8 rows go to the second RAM) and 300 columns, which is the number of samples S in our case.
- Each data value in the polar coordinates are converted to train of "1"s and "0"s and they are written in a text file that the parallel port program will take and load the RAMs accordingly.



		Column 1 (Position 1)	Column 2 (Position 2)									Column 299 (Position 299)	Column 300 (Position 300)
RAM1	LED1	97	108	110	78	98	87						
	LED2	27	240	225									
	LED3	99	115										
	LED4	67											
	LED5												
	LED6												
	LED7												
	LED8												
RAM2	LED9												
	LED10												
	LED11												
	LED12												
	LED13												
	LED14												
	LED15												
	LED16												

Figure 4.3 – Polar image example for 16 LEDs

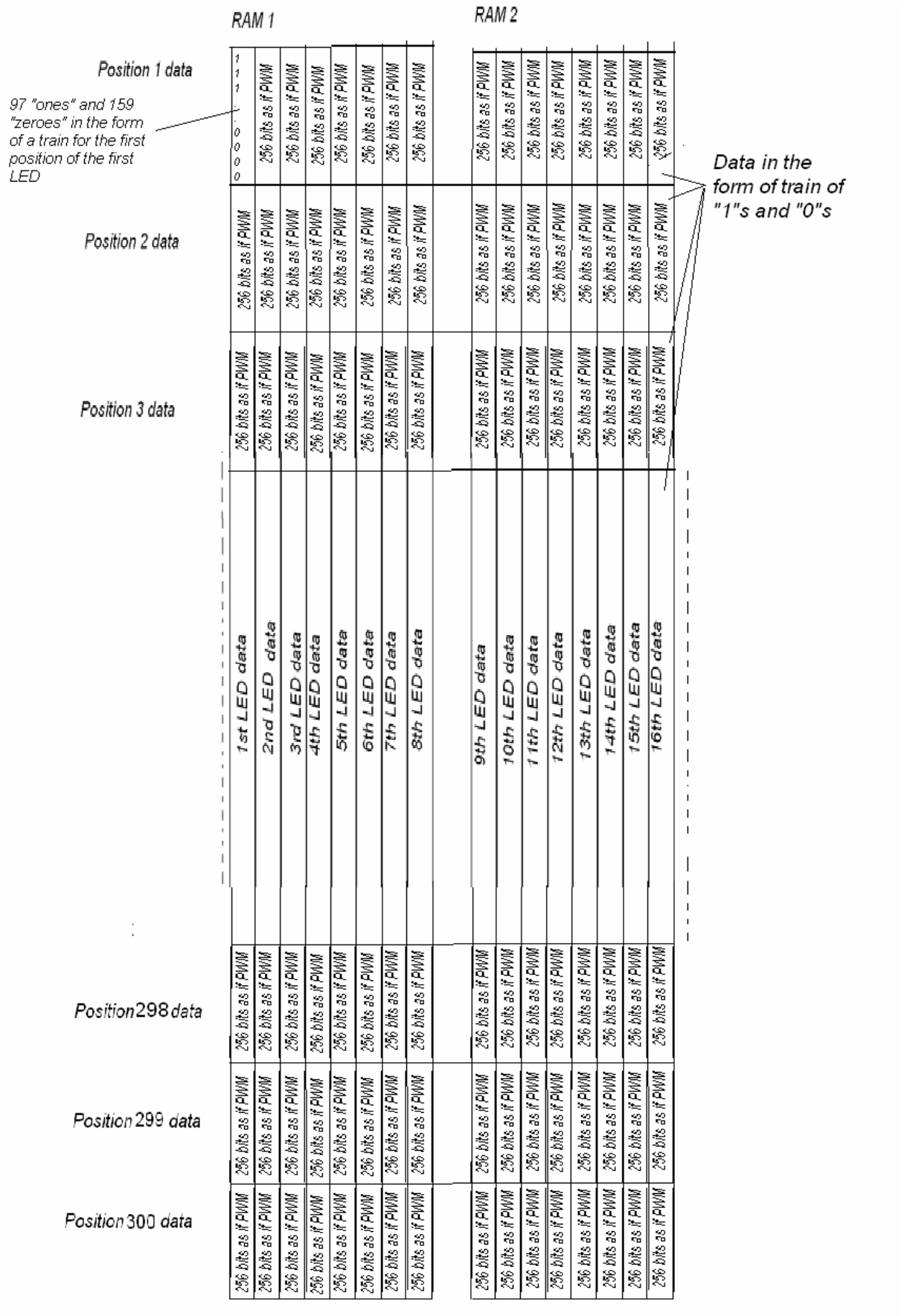


Figure 4.4 – Data as seen in the RAMs for 16 LEDs

New algorithm for RGB and high multiplexing:

Our discussion of trains of 1s and 0s remains valid however after some modifications. Since a RAM is not driving a single set of 8 LEDs but driving two of them, each at one clock cycle, the RAM data should be prepared so that in 512 consecutive addresses, it holds the two "trains" of each LED value combined. At the first clock cycle, the first values are displayed and this corresponds to the first "wagons" of the combined train, the second clock value calls the second wagons into play and this goes on. The MATLAB code is written again to accommodate this change of algorithm.

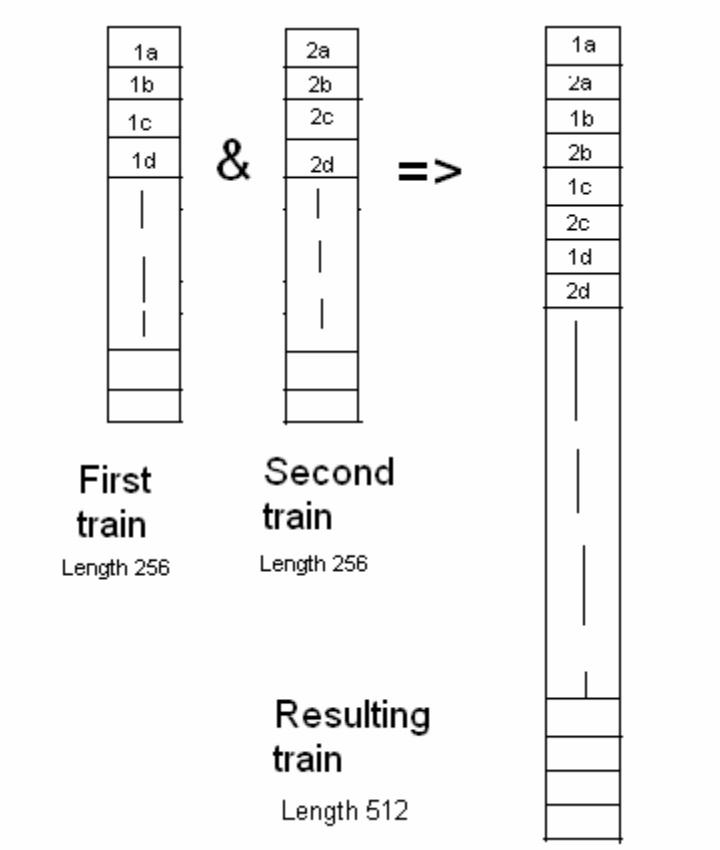


Figure 4.5-Combining trains for elements of two consecutive sets of LEDs

Another very important point here is that since the sets of 16 LEDs are oriented with 120^0, so the data in the two other RAMs should be delayed by S/3 and 2S/3 positions, where S is the number of positions sampled in one cycle and 300 for our case. The RAM columns are adjusted to accommodate this delay automatically in our MATLAB code.

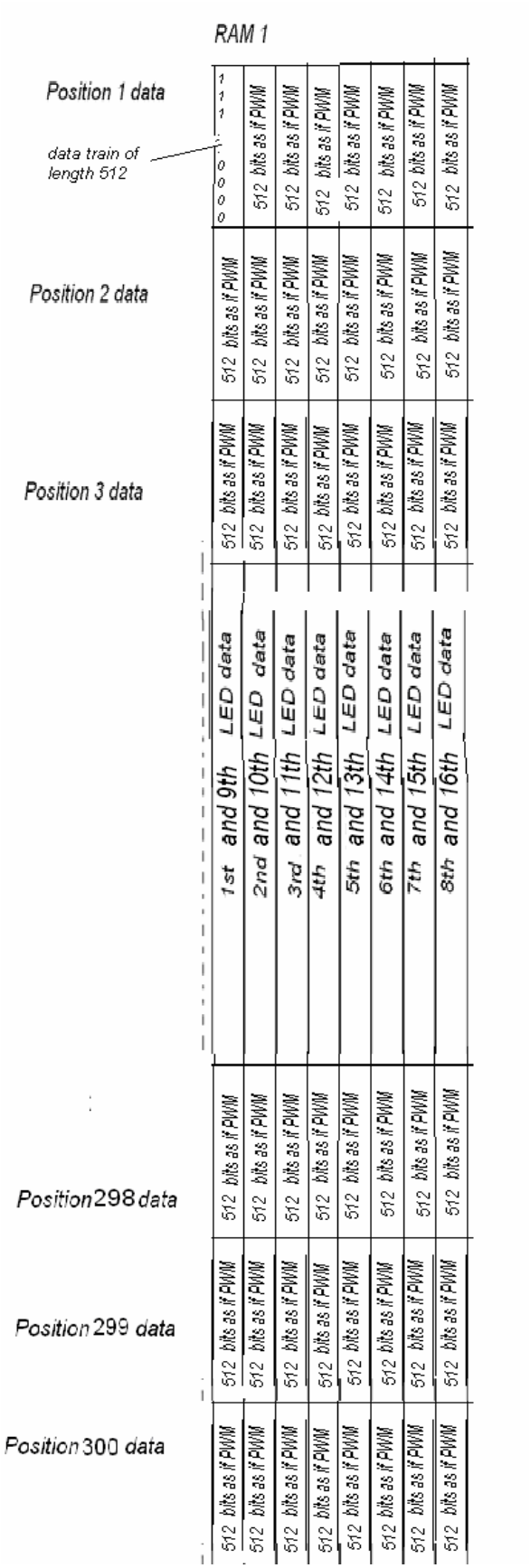


Figure 4.6- New data hierarchy in one of the three RAMs

4.3 Communication with the device – Parallel Port interface

4.3.1 Writing Controls

We implemented the coordinate conversion method using MATLAB and we are ready to transfer the resulting image to our device. We know from the previous sections that we have four bits to control the circuit using parallel port and eight bits for data. Two bits were used for controlling the RAMs. In this step, analyzing the behavior of the RAM is important:

$\overline{CS}$	$\overline{OE}$	$\overline{WE}$	I/O Pin	Mode	Power
H	X[1)]	X[1)]	High-Z	Deselected	Standby
L	H	H	High-Z	Output disbaled	Active
L	L	H	Dout	Read	Active
L	X[1)]	L	Din	Write	Active

1. X means don't care.(Must be in low or high state.)

Figure 4.7: RAM functional description

We see that a write operation occurs at the point where both WE and CS are low. But if we keep them constantly at a low position during write cycle, then there is the problem of writing to previous or next addresses.

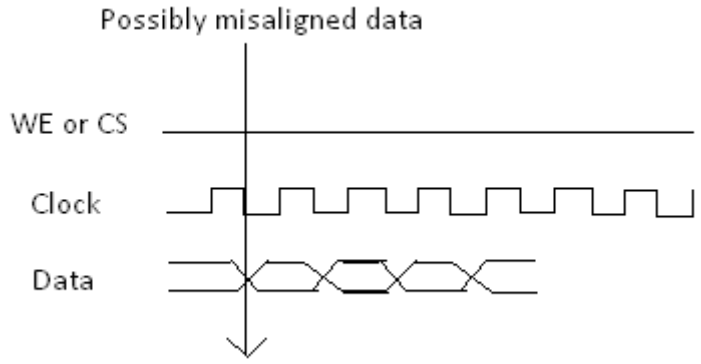


Figure 4.8: Writing without close control of WE or CS

A small delay can cause a problem using this type of writing. Misaligned data can cause problems and shifts in our image, so we must control WE or CS and close the RAM for writing during address change. According to the datasheet of our RAM, there are two ways for control of write cycle:

First one is WE controlled:

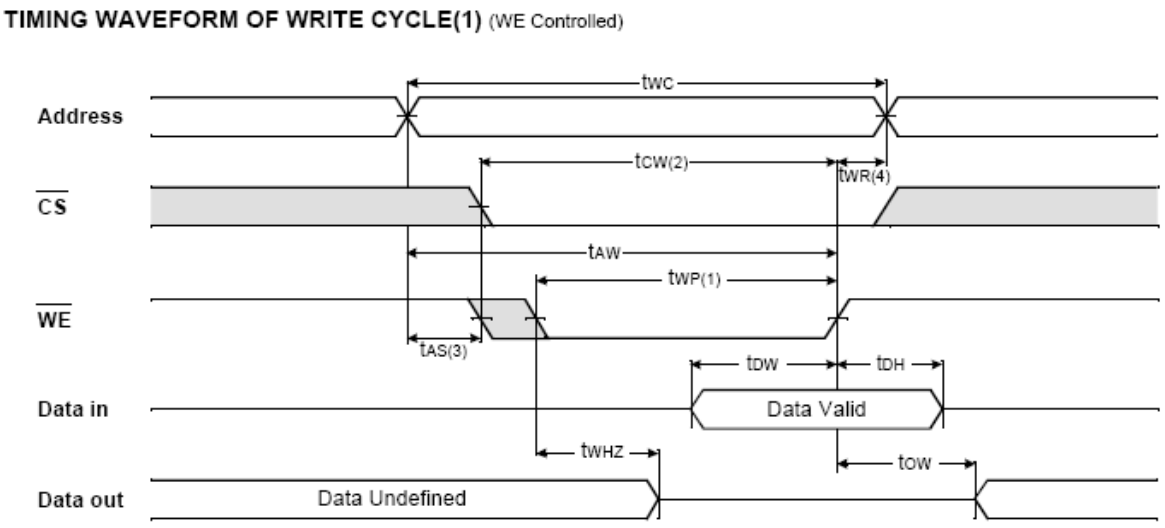


Figure 4.9: Timing Waveform of Write Cycle (WE Controlled)

And the second one CS controlled:

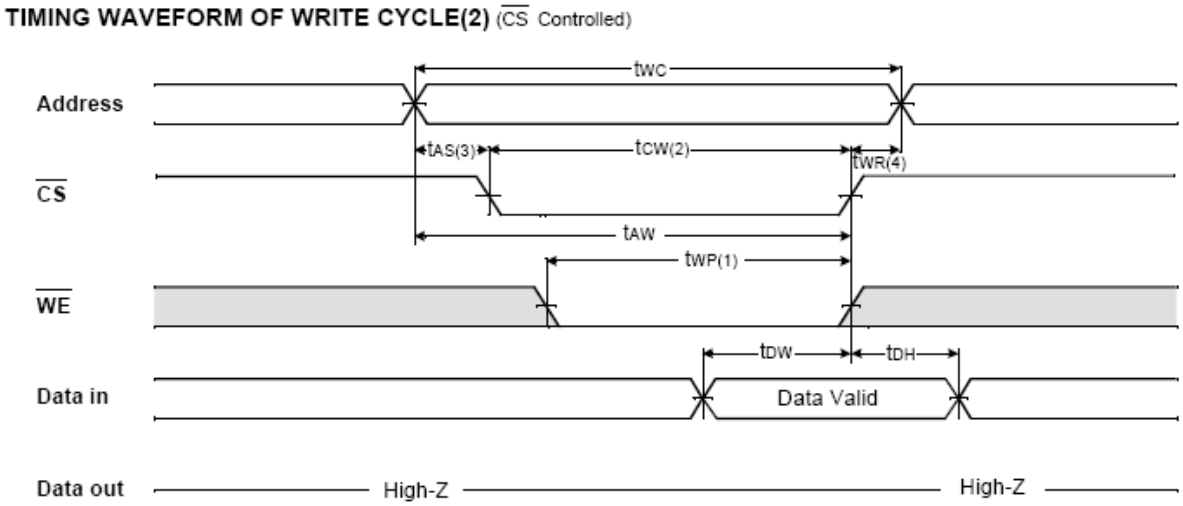


Figure 4.10: Timing Waveform of Write Cycle (CS Controlled)

We selected to connect CS to ground and use WE for control. In WRITE mode, since the OE bit is the connected to R/W switch, it is always high. In this state, setting WE bit to low enables write operation and setting WE bit to low disables RW operations and sets the output of the RAMs to high impedance so that they do not interfere with buffer outputs when the RAM is not written.

The technique we control WE is that we set WE to high before there happens a clock transition, namely high to low transition, and to low after the transition is made. This makes us sure that we do not have problems with writing to wrong memory location with small delays. We can show our technique as follows:

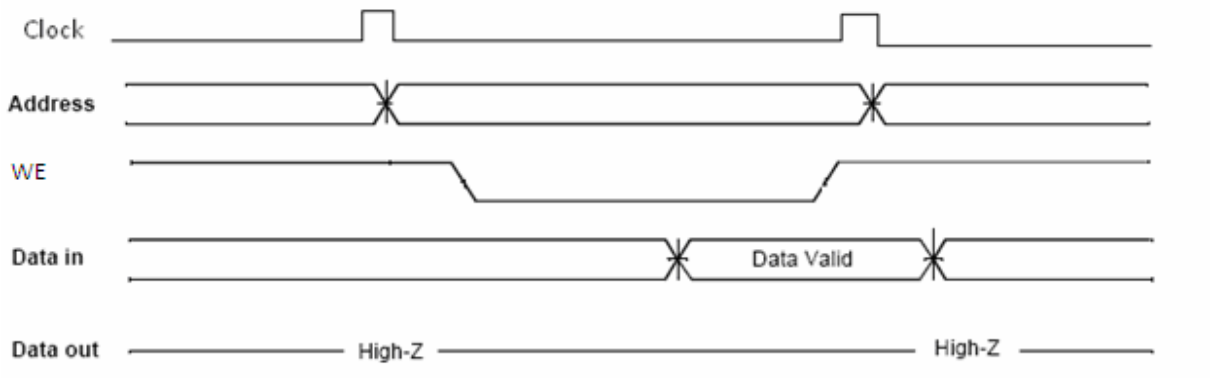


Figure 4.11: Counter Clocks and WE control implemented

It is clearly seen that after seeing the clock pulse, we wait some time before writing. The implementation of our software is to wait for time of duration T after the clock tick to WE transition, time of duration T for WE in low position and time T after switching WE to high to next clock tick.

It is also important that we should control two RAMs independently and after writing the first RAM, after resetting the counter, we should continue to the second RAM. So, there are actually 3 WE bits, one for each RAM, two being totally of at a time. Also, after operation, the RAMs should be put back to write position and the control should be given to internal controls.

We implemented a program in C# to load data to the device and it is discussed in the next section.

4.3.2 Program for Data Transfer

We built up the framework for successful communication and we got ready for the actual transmission of both the data and control bits. We decided to use 8 parallel port data pins for data and 4 parallel port control bits for controls (Clock increment, Clock reset, two RAM control bits), and finally 1 status bit for checking the state of the device. I think giving the pins that are used again will be better:

D0-D7 Data pins → Transmitting the data of image

Strobe → Counter Increment

Autofeed → Counter Reset

Init → RAM control bit 1

Select Input → RAM control bit 2

ACK → Receiving data from the board

We will continue our discussion of the program after giving the overall view of GUI:

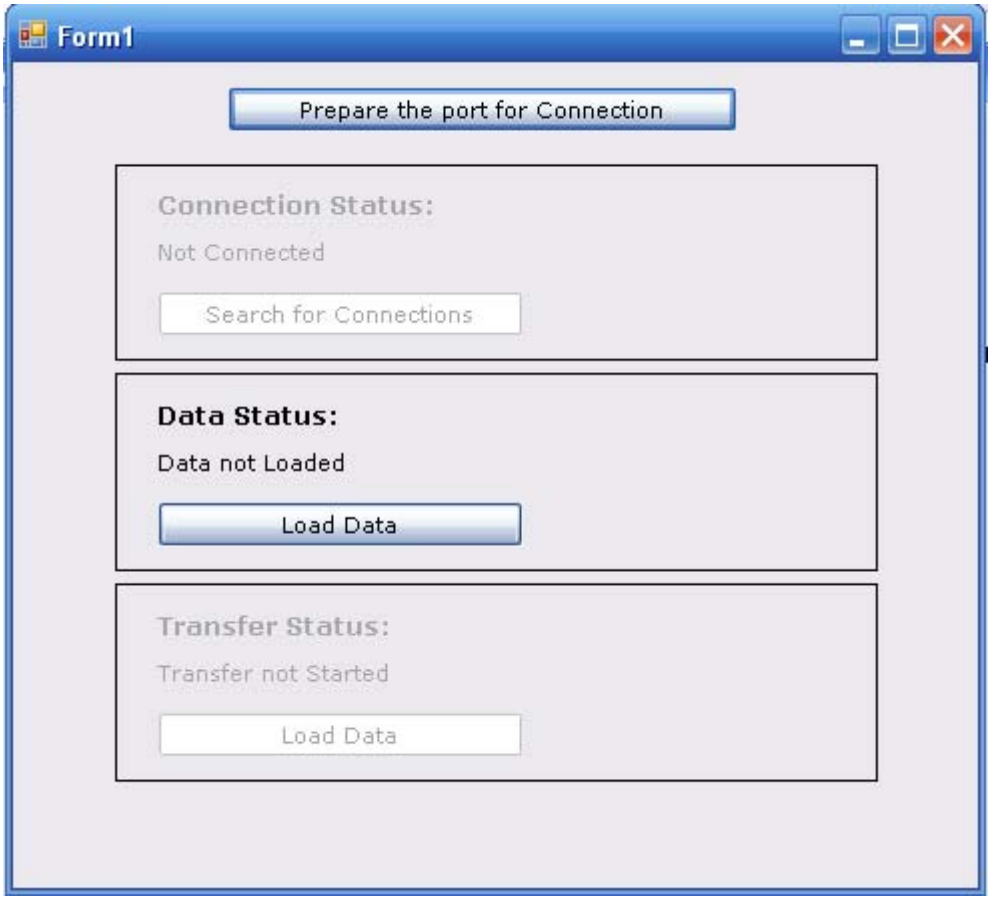


Figure 4.12: Parallel Port Data Transfer GUI

First, we see the "Prepare the port for connection" button. It prepares the port so that we start in a counter reset, RAMs closed position instead of a random state before connecting the parallel port device. After that phase, 'Search for connections' button is enabled.

Secondly, we see Connection Status part and 'Search for connections'. It actually looks to the status bit and gets the status of the device that the parallel port is connected to. It takes the switch position and controls it. If it is in a low position, then it means the device is in READ position and one should not attempt to write to the device. It only allows transferring data if that switch is found in a high position, corresponding to WRITE. We should note that Load Data option of transfer is greyed out before making that check.

Thirdly, Data Status tab is seen. It helps the user to browse and load the image data which is processed for our system beforehand. The image data is then processed into computer's RAM.

The last part is the actual transfer to our device. When all other operations are successful, this option is enabled and when the user presses this button, according to above schema, the data is transferred sequentially. After the operation, write operation is closed totally and the counters are reset. The user is then asked to change the switch position on the board.

The program is a modified version of the one designed in last semester actually, because it is now designed so that it supports three RAMs in the circuit.

Coupled with Matlab processing, these in turn gave us the means to implement 24-bit color depth.

The program is implemented in C# using Microsoft Visual Studio 2005. The program requires .NET Framework 2.0 to run. The code for this program is given in APPENDIX C.

CHAPTER 5

Main PCB Design - Power Transmission – Mechanic Platform (Test-Bench)

5.1 Power Transmission

We planned a practical design for power transmission as can be seen below. Briefly, we planned to use an additional PCB to prevent losing the central pixel of the display (without this PCB power transmission connector and our central pixel are overlapped and this case will obviously corrupt the image). In addition to this, additional PCB is benefited for Infrared circuit which is for reset the counters (detailed information about this IR circuit is given in relevant chapter) by making a hole on board.

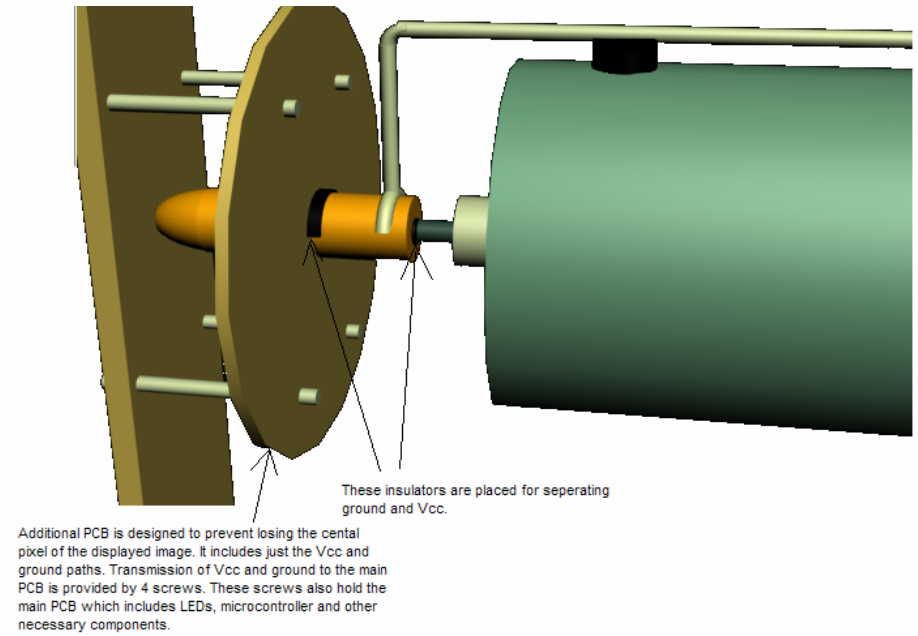


Figure 5.1: First planned Power Transmission Design (It could not be realized)

Although that design is fitness for our purpose, we experienced some problems while constructing the design. The most significant problem that we face was existed in the design of carbon brushes of which duty is transferring the Vcc to the spinning part. Since motor speed is so high (about 3600 rpm) screw that holds the rotor was shaking too much which was not able to prevented. Because of this case, connection between isolated screw and brush detached continuously, and stable power transmission was not realized. After several trials for managing to work it, we decided to change brush technique since it seemed to an unreliable solution. So we started to find a more efficient mechanism than brush technique. After several attempts, we found a special capscrew which has two cylindrical metal bodies; each can spin independently with each other. However they are connected with each other by little metal spherical balls placed between outer metal body and inner one. Through this mechanism, inner body can turn with rotor at same speed; however the outer body to which connects the power transfer cable is immobile. That is to say that, this new type of capscrew fortunately solves the problem of

unstable power transmission originated in shaking. And we decided to use it in Vcc transmission.

Another important change in the design of power transmission should have been existed in transferring the Vss (ground) to the spinning part. We planned to use a carbon brush which directly connects the rotor for ground transmission. And at the end of the rotor which was isolated (for preventing short circuit case) before coming close the special screw, rotor should have connected to the ground copper pad of the previous PCB. But then we realized that this method was impractical, since after constructed the power transmission body the rotor could not connect continuously on the copper pad due to destruction of the pad while drilling the board. We decided that this method was unreliable and so it should be changed.

We solved the problem stated above by using a special threaded nut which connects directly to the motor and also since it has a great radius, it can directly connect the semicircular copper pad which is specially designed for ground transmission. Thus this problem was overcome.

It should be stated that we used the body of the motor for ground transmission due to its direct connection to the rotor as we realized in the previous term.

Although we solved the problems stated above, unfortunately we experienced an unexpected problem at the end of this term. Since our first board (power transmission PCB) is not symmetrical with respect to the center (drilled hole into which rotor passed), our motors rotor was a little bit wrapped. So this situation caused a shaked rotation which sometimes causes disconnection in the overall transmission path. This problem could be overcome, but since we had a powerfull drill motor we decided to use it instead of our current motor.

There is no significant change existed in our design, even it makes easier our Vcc transmission fortunately since it has a long drill screw which provides us with more convenient workspace. But since the body of the drill is plastic we could not find a way to transmit the Vss at first. But we guessed that there should be a connection, which directly connects to the rotor, in the drill. For this purpose we tried to find this connection by a long conductor (this conductor connects one of the probe of multimeter) and a multimeter of which other probe directly connected to rotor. In other words we tried to find the short circuit between the rotor and a conducted point which will be the starting point of our potential Vss transmission path. A little later we found this point in the drill easily. So our design becomes similar to design of our previous motor.

To sum up, we transmitted the Vss by using the rotor which connects directly the semicircular copper path on the center of the motor through the threaded nut of which duty likes a conducted bridge between the rotor and semicircular copper path. And through our special capscrew we transmitted the Vcc. We transmit the Vcc the outer circle, which is immobile, of the capscrew. And we interposed the cable between the isolated rotor and the inner circle of the capscrew which is mobile and rotate with PCBs. Cables beneath the inner circle then soldered to copper path, which is designed for Vcc transmission on the Power transmission PCB. Thus we managed to transmit Vcc to the rotated PCBs.

Below the power transmission PCB image is given. The duty of each location is explained in detailed.

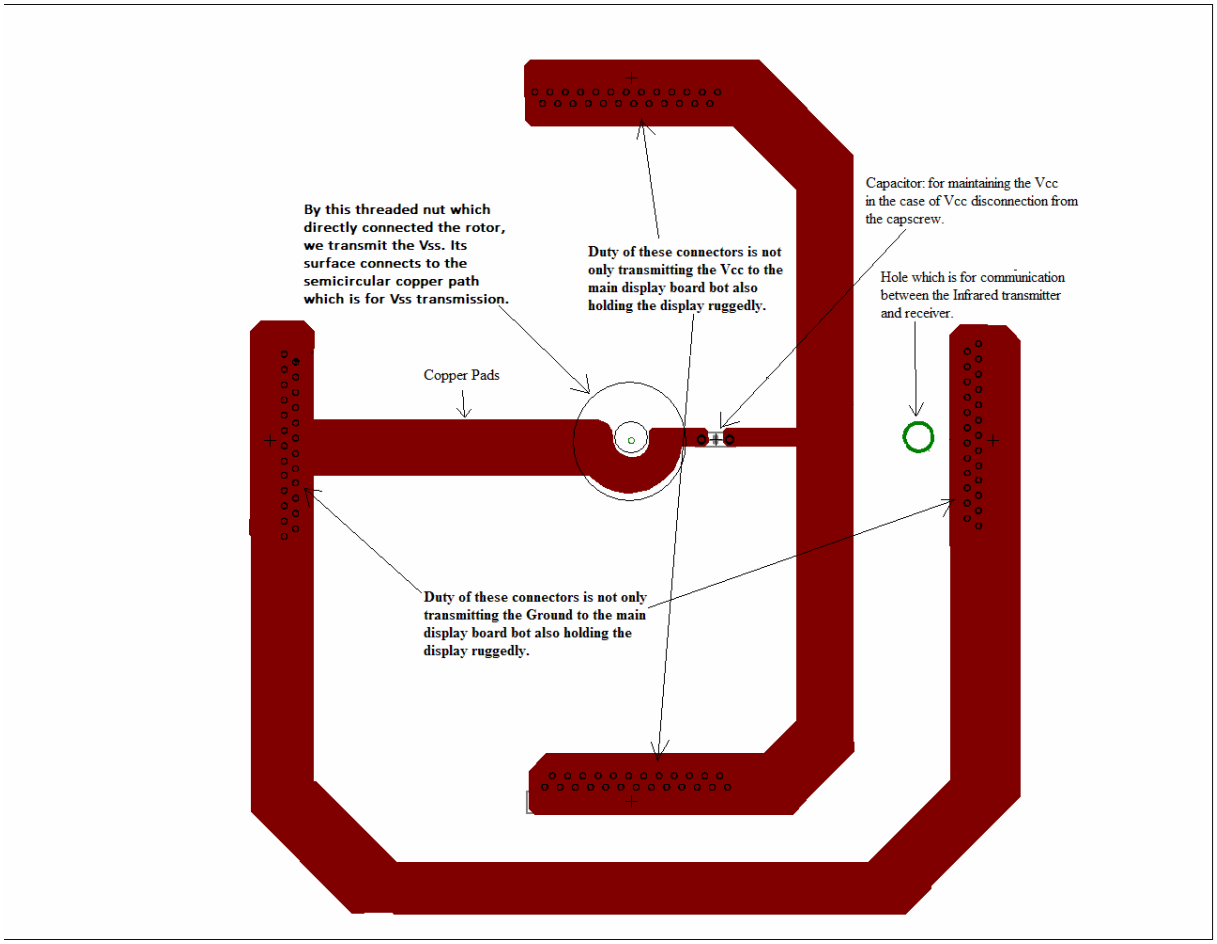


Figure 5.2: First PCB

Overall power transmission design implemented on motor is given below.

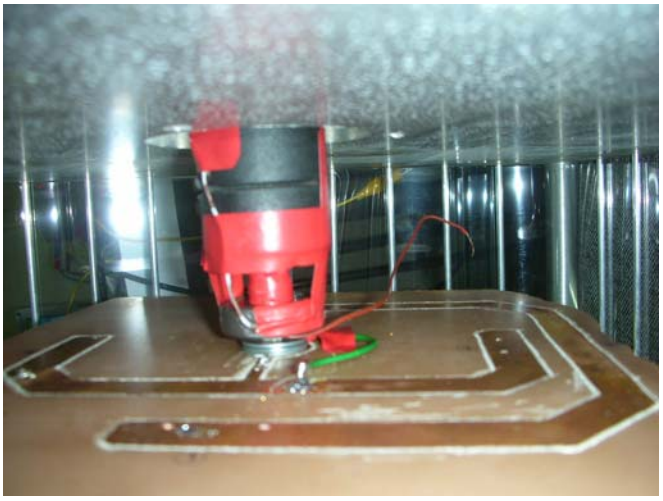


Figure 5.3: Power Transmission Design (with back PCB) Implemented on Motor

After making the necessary modifications to our first design as explained, we checked power transmission design by placing a LED on the spinning board. We tried to prevent possible impulsive disconnection by utilizing a capacitor connected between the Vcc and Vss. We saw that a circular image created by this LED, also there was no wrinkle, so it was the evidence that the power transmission design works properly. Photos demonstrated no wrinkle motions of active LED are given below.

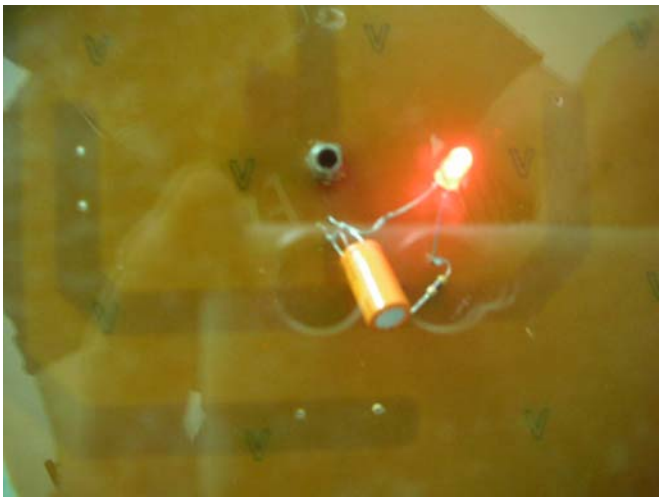


Figure 5.4: LED is turned on, but rotation has net started yet.

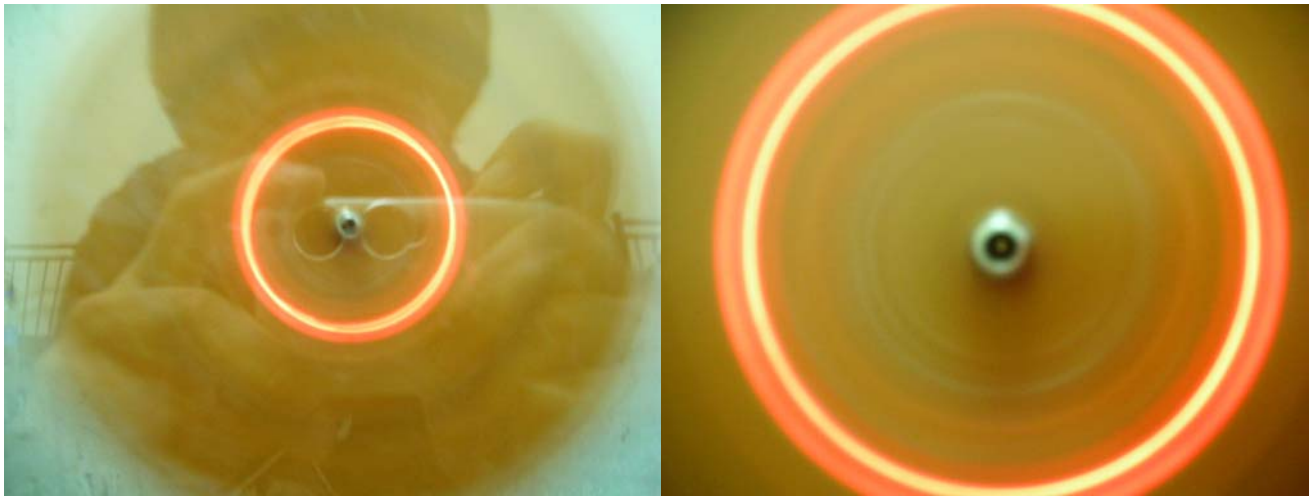


Figure 5.5: LED is turned on and rotation started.

5.2 PCB Design

After finishing the self designed schematic design, we started to design PCB. This part of the project is so important since the PCB design method can directly affect the operation of the overall circuit. That is to say that even if the connections of copper pads are consistent of schematic, circuit might not be operated as

desired. The most important PCB design rules that we strictly tried to comply with are the adjustment of the copper pad width in some special location, and placing the oscillator circuit.

Each of our LEDs sinks 20mA from the buffer and discharge this current to the same GND pad. In our current design we have three groups of LEDs, each includes 16 LEDs for RGB design. As we realized in last semester, we try to maximize the thickness of the ground paths of LED groups as possible as we can in order to prevent electromigration effect. Also we make each groups current discharge to main ground connection through different paths due to the same purpose stated. This design specification can be seen from the figures demonstrating the PCB.

That is the fact that 0.3mm (minimum thickness of ordinary copper paths of the circuit that manufacture can product) is not sufficient so much current (20mA X 16 = 320mA for each group of LEDs). If this amount of current passes through a thin pad, Electromigration can be occurred. In other words, there can be a failure of a copper conductive strip due to electromigration. So that thickened pads gone through the LEDs and common GND pad as possible as we can. In addition to this Vcc and Ground connectors' (which also hold the main Spinning Display Board) pads were thickened too much in order to reduce the resistive effects of them.

Another important rule is that the circuit elements which constructed the oscillator and directly connected the oscillator circuit (crystal oscillator, capacitors, inverters, and multiplexer in our circuit) should be placed closely with each other. We obeyed this rule also.

Another restriction that we have to allow was PCB manufacturer's minimum specifications. For most efficient design we should have use minimum clearance (distance between pads), and pad width should have been lower than 0.3mm, but since we could not manufacture the PCB ourselves with our self adjusted specifications we have to ask for help to a professional PCB manufacturer. Our chose manufacturer informed us that they can print the minimum width of 0.3mm for copper pad, and 0.3mm for clearance. So we had to allow these restrictions, and drew the PCB by using EAGLE 4.16r1 Professional Edition software. We placed the components in most efficient way, and routed the important connections (pads gone through the LEDs, common GND path, Vcc and Ground connectors' pads etc.) ourselves, and we made EAGLE's Autorouter route the rest of the board. By using autorouter's most complicated routing algorithm, we finalized routing the board.

It should be mentioned that PCB design process is almost similar to last semester's design. So, above specifications are valid in both our current PCB and last semester's design. But since the schematic of our current design is more complicated (we increased the number of RAMs, buffers, LEDs, switches etc.) routing our new PCB was so exhausting and difficult process for us.

The most important difference of our current PCB is deciding the locations of the LEDs. We decided to placed each group of LEDs like medians or bisector (each median ends at the center of gravity) of a triangle. In other words, assume Cartesian coordinates; the first group (for Red color) is located along the -y- axes. The second group (for Green color) is located along the line which makes an angle of 120° with -y- axes. And the last group (for Blue color) is located along the line which makes an

angle of -120° with -y- axes. Center of gravity obviously becomes the center of our circular motion. The figure demonstrating the locations of LED groups is given below.

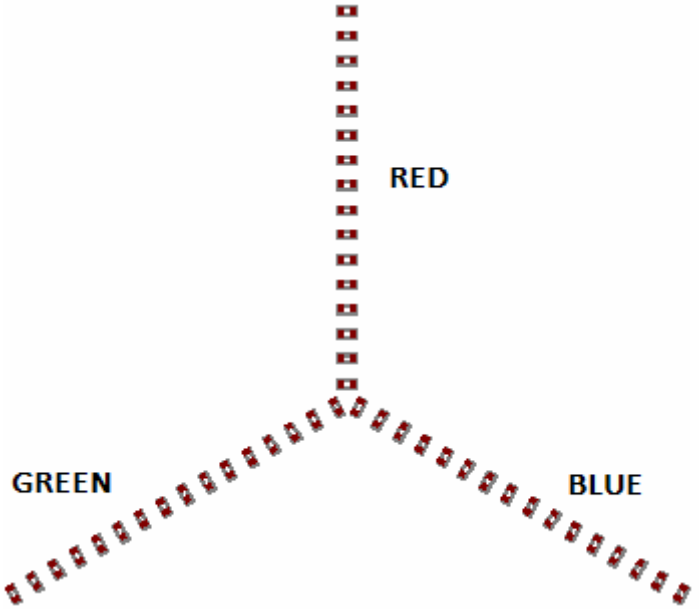


Figure 5.6: Location of RGB LED groups on the PCB.

Finally overall PCB Design is given below. Since understanding the both of top and bottom layer together in same image is too difficult, each layer is shown separately. Final figure is for overall PCB.

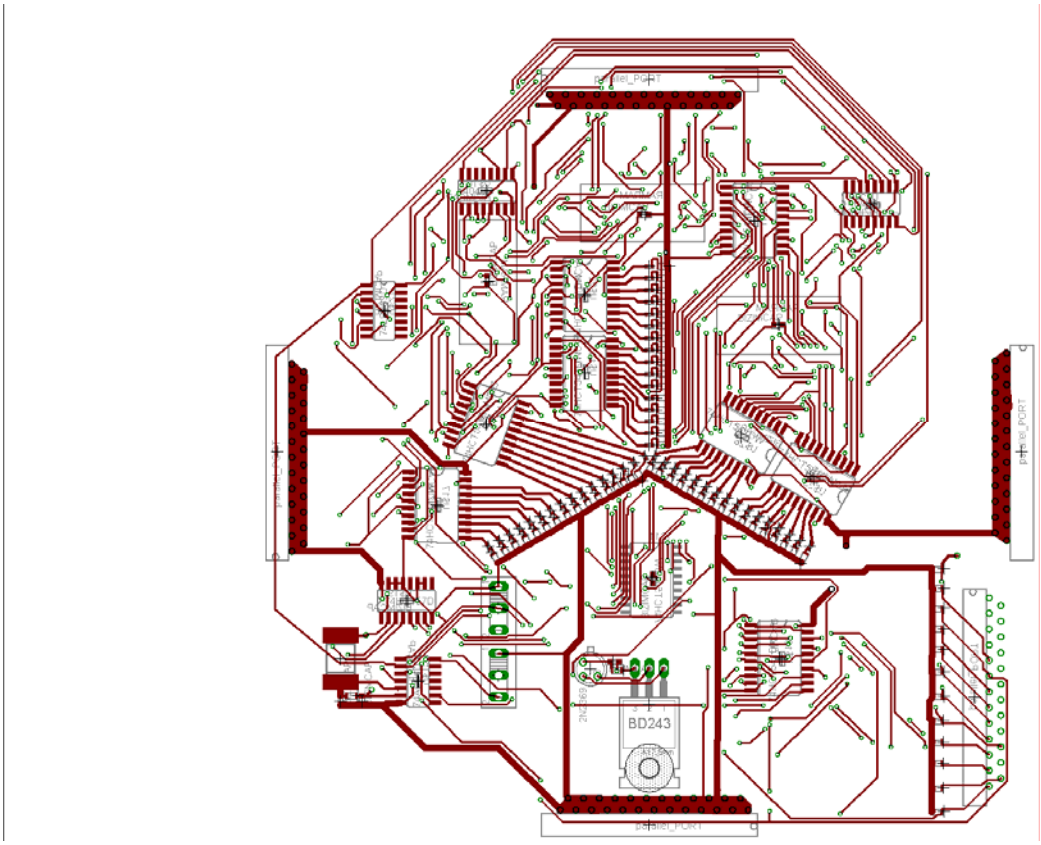


Figure 5.7: Top layer of PCB.

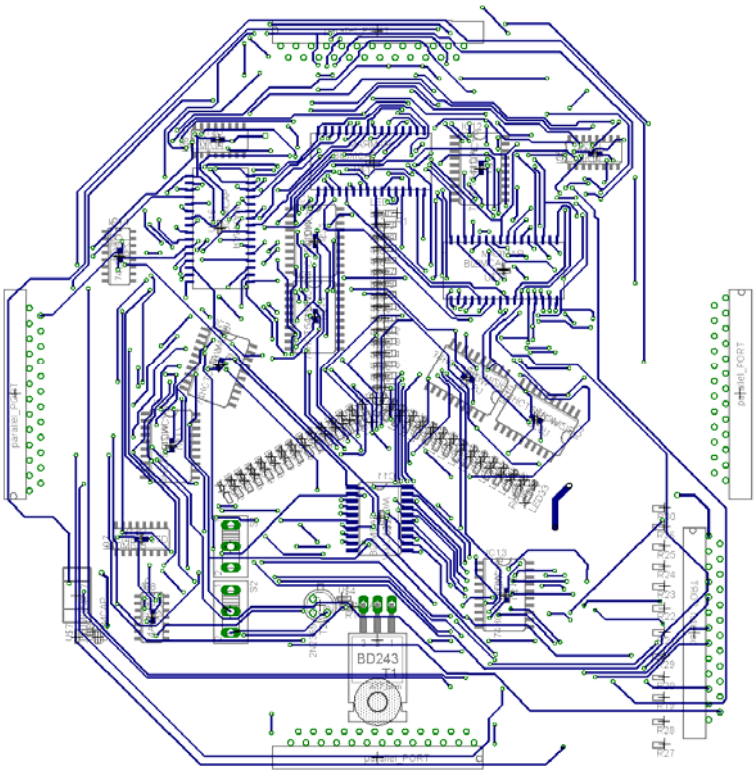


Figure 5.8: Bottom layer of PCB.

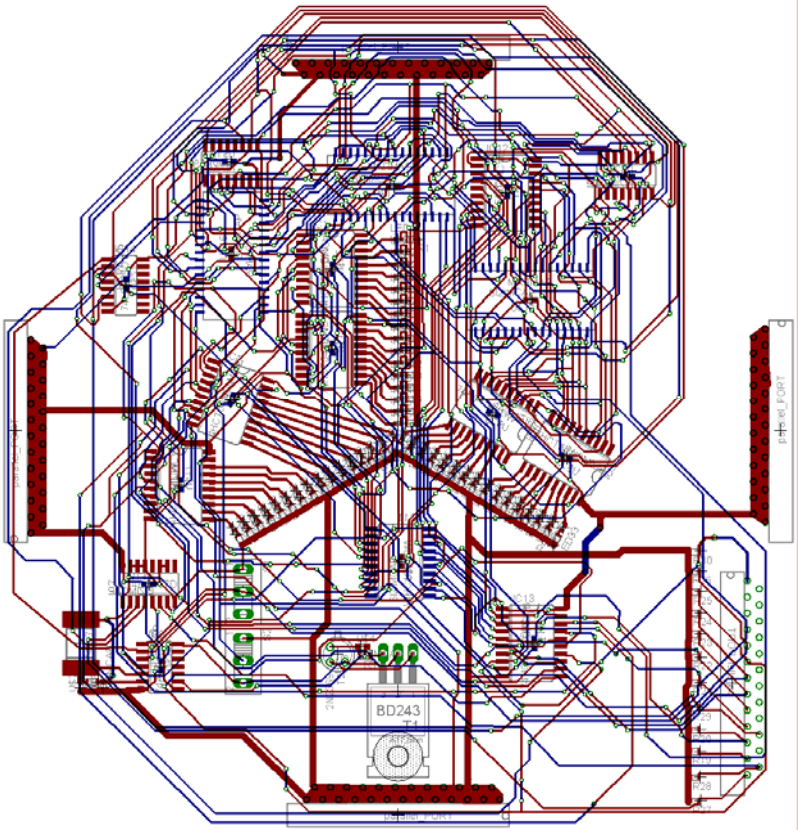


Figure 5.9: Whole PCB with both TOP and BOTTOM layers.

5.3 Mechanic Platform (Test-Bench)

In previous term we experienced a serious problem which is that we required a platform on which we placed our whole design. Lacking of a specific platform was problem for us, since our motor is rotated at a huge speed and at that speed it is impossible to control it by hand or any other weak holders. Motor should be hold by a very strength holder so that its shaking effect which affects vision negatively can be prevented. But another mechanic problem that we should overcome is protecting people from unexpected accident which is protruded small components or parts of board stemming from centrifugal force of motor due to boards huge speed.

So we decided to make a strong and efficient platform in the beginning of this term. This was an urgent necessity, since all of tests should have been done on this platform securely. For these purposes we started to make its design and tried to produce it at the beginning of this term. Below the design and production process are explained in detailed.

Our project's aim is to show an image which is from a rotated board and its "soft-pixels" stemming from the rotated LEDs on the board. The closest design to our display is a monitor or a TV. So we decided that the most suitable design is a platform which should provide us the above requirements and resemble the computer or TV casing.

Then we started to design our platform (Test-Bench) by using GoogleSketchUp 3D drawing software. After evaluating specifications that we require, our design was completed. Our design is given as follows.

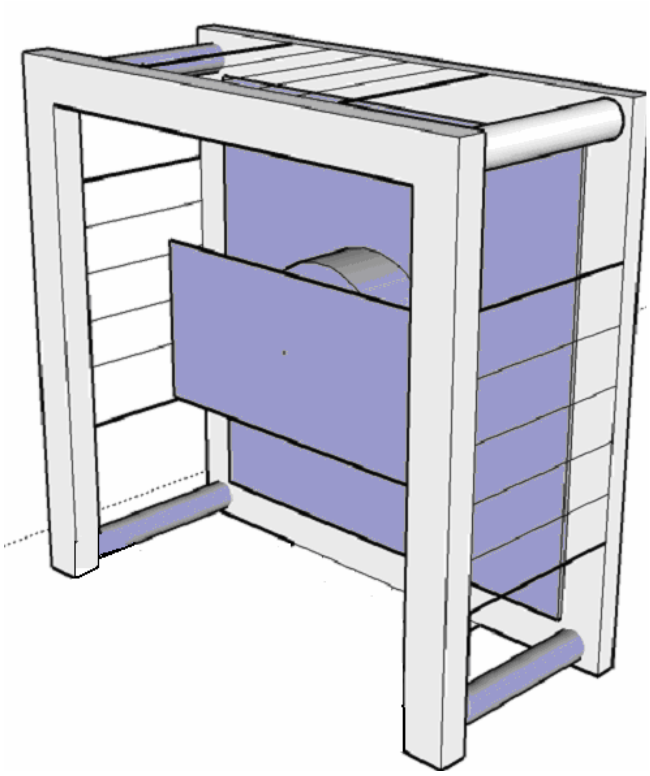


Figure 5.10: Technical Drawing of Test-Bench

After designing the platform, we need a professional help to implement our design. So we requested Machine workshop's help which is for machine engineers. Through the professional engineers directions we bought the discrete materials which will construct our Test-Bench. Our design comprised of aluminum (Aluminum 7075 which is machined so easily) for back of the platform, curtain parts and screws which connects back part to the front panel. We used bell metal to make the frame which constructs the front panel. The plexiglass material which is transparent like pure glass, but machinable in contrast to pure glass, is used for the display screen. Besides, a small plastic material is used for motor holding. We could have hold the motor just using the aluminum back part of the platform, but there was a possibility that our current motor may change in case of its lacking of capability to move our new board. So we made a hole which is diameter is bigger than our current motor diameter, at the center of back aluminum part (Both of our motor's diameter and plastic part's inner diameter is 370mm. But diameter of the hole which is on the back aluminum part is 600mm). And just behind this hole we placed a plastic platform of which inner diameter matches our motor's diameter; its duty is stabilizing the motor only. In the case of a new motor's necessity we remove this part and put another plastic part instead of it. If the fact that difficulty of machining the aluminum part which is tied to other parts of platform is more complex than machining a free plastic part is considered, the effectiveness of our solutions can be realized easily.

We changed our motor as stated in power transmission part. But reason is not its lack of capability to move the boards. Substitution is made due to wrapping on rotor mil. But fortunately the radius of our new drill motors body is lower than the first motor. So by exerting pressure with little screw embedded in plastic part, we managed to clasp it.

Images for our completed Test-Bench are given below.

Figure 5.11: Front View of our Test-Bench

Finally, dimensions of our Test-Bench are given below as figures.

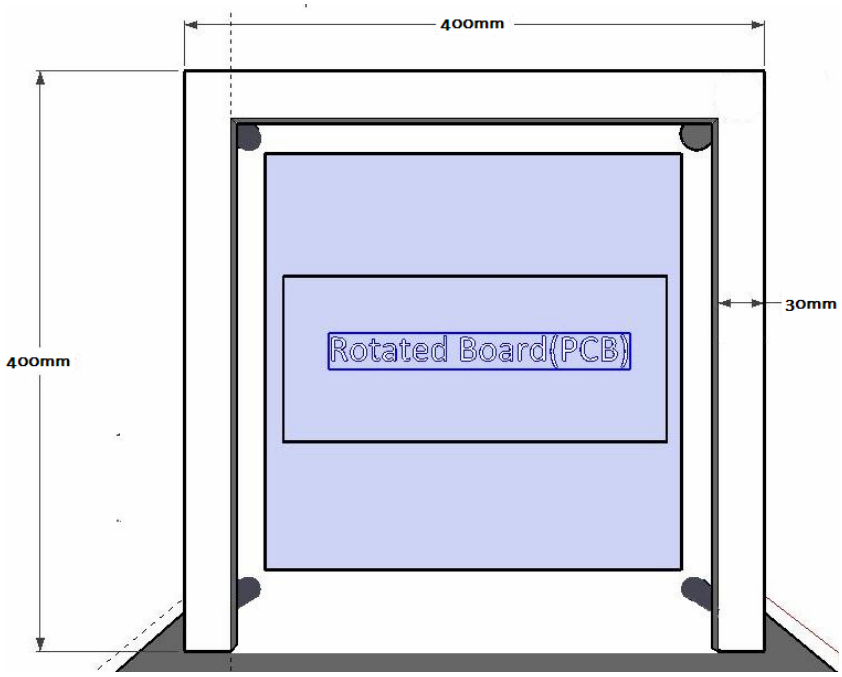


Figure 5.12: Front View and Dimensions

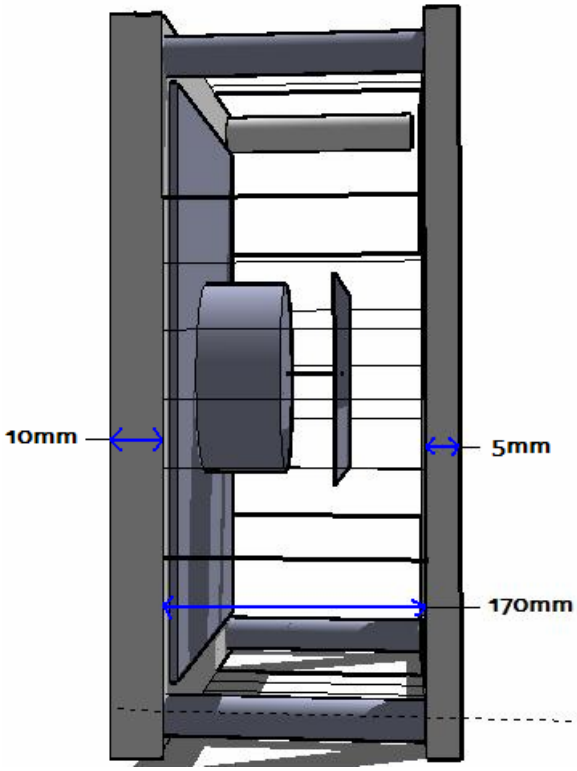


Figure 5.13: Side View and Dimensions

CHAPTER 6

Motor Control

6.1 Why is Motor Control Needed?

Motor Control is one of the most important fundamentals of the Spinning Display Project. According to our early theory, we can easily see that delay caused by motor speed leads to mismatch of data to be seen on the screen. This may result several problems such as image to be in a bad mood such as; misplaced, expanded, compressed or a rotating image. Therefore we need to stabilize the motor speed.

6.2 Linear Voltage Motor Control

Since we use a DC motor to rotate the rotational part of the project, we had mainly two alternative ways to stabilize motor speed. The first way is to use PWM. Spinning Display Project's early works showed us that using PWM (Pulse with Modulation), would be unsuccessful about lots of issues. The most important defect of PWM was the ON-OFF principle of it that disturbs continuity and causes the LED's positions to mismatch with their expected positions during lowering and increasing the speed of motor. Because, image frames are refreshed in a very high frequency and PWM is done in a very low frequency.

Moreover, the duty cycle of the PWM directly affects the amount of energy applied to the motor. The frequency of the PWM waveform will also influence the motor's operation. Since the motor is a dynamic machine, the PWM frequency can be fairly low, before the motor starts to pulsate noticeably in synchronization with the PWM frequency. This leads to an important acoustic problem. Therefore we should set the frequency of PWM as low as possible. This is again unsatisfactory for us.

Second main alternative is to use Linear Control Method to stabilize motor speed. The project's recent works showed that the most efficient way would be controlling the motor's speed by modifying the supply characteristics of the motor. Although this linear-voltage method is highly power inefficient, it is admissible for us, in case we achieve to stabilize.

6.3 Hardware Design

The speed of the DC Motor is controlled with the closed loop feedback circuit shown below, by varying the crossing current through the motor. In the first figure you can see the block diagram of the circuit, with general steps. The schematic of the circuit and also the PCB layout of it, are below too. The necessary drawings are composed by using software, *eagle*.

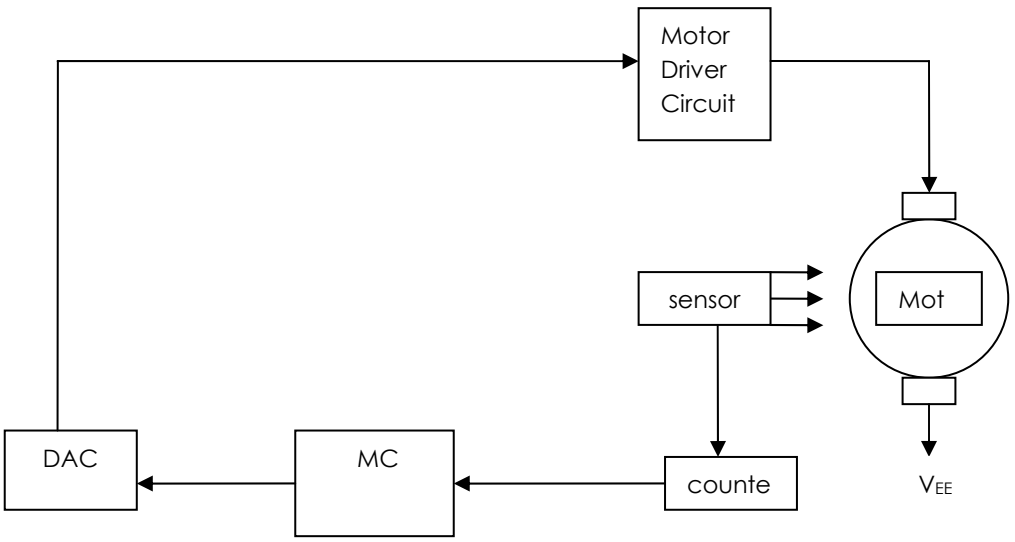


Figure 6. 1 Block Diagram of the motor control circuit

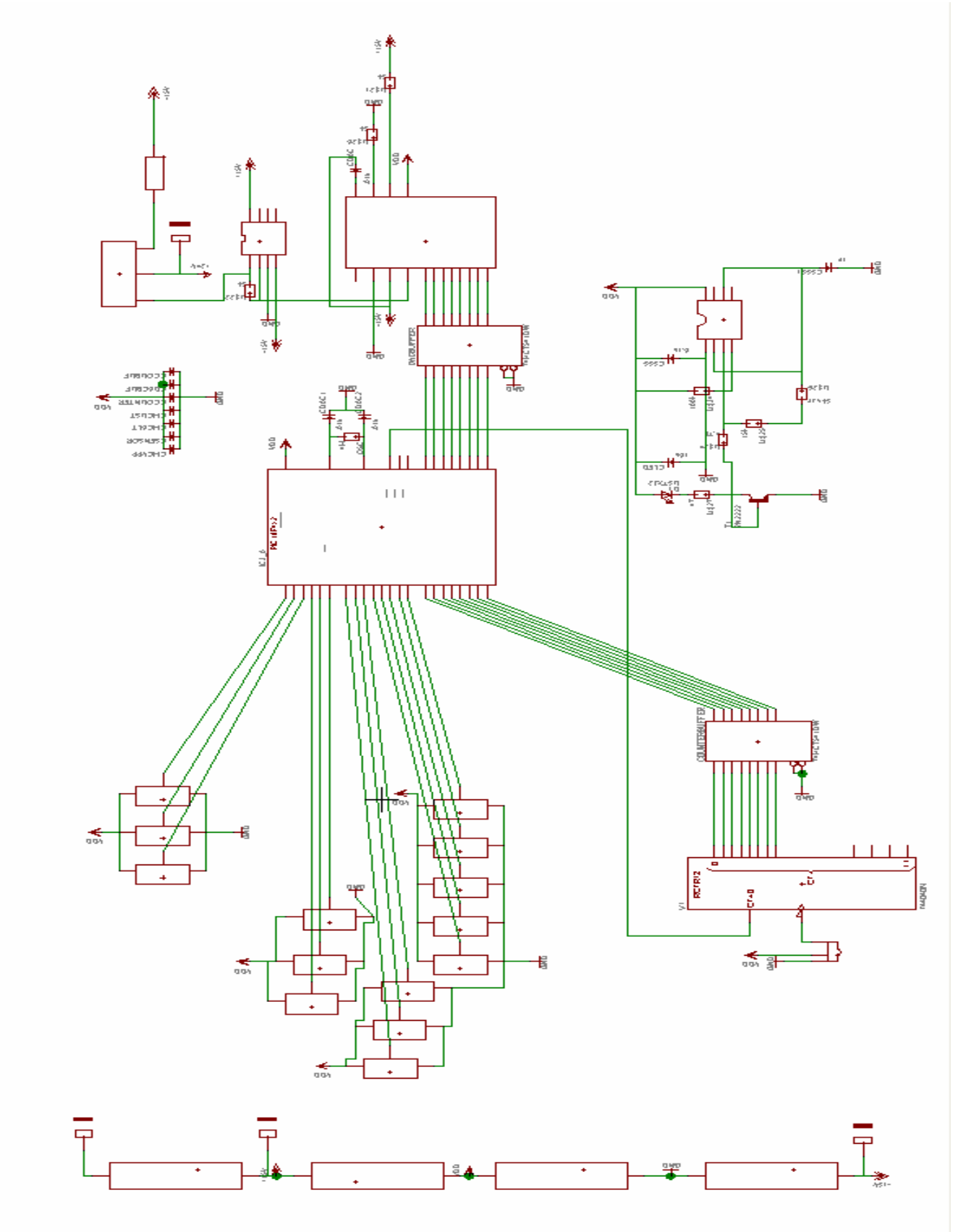


Figure 6. 2 Schematic of the motor control circuit

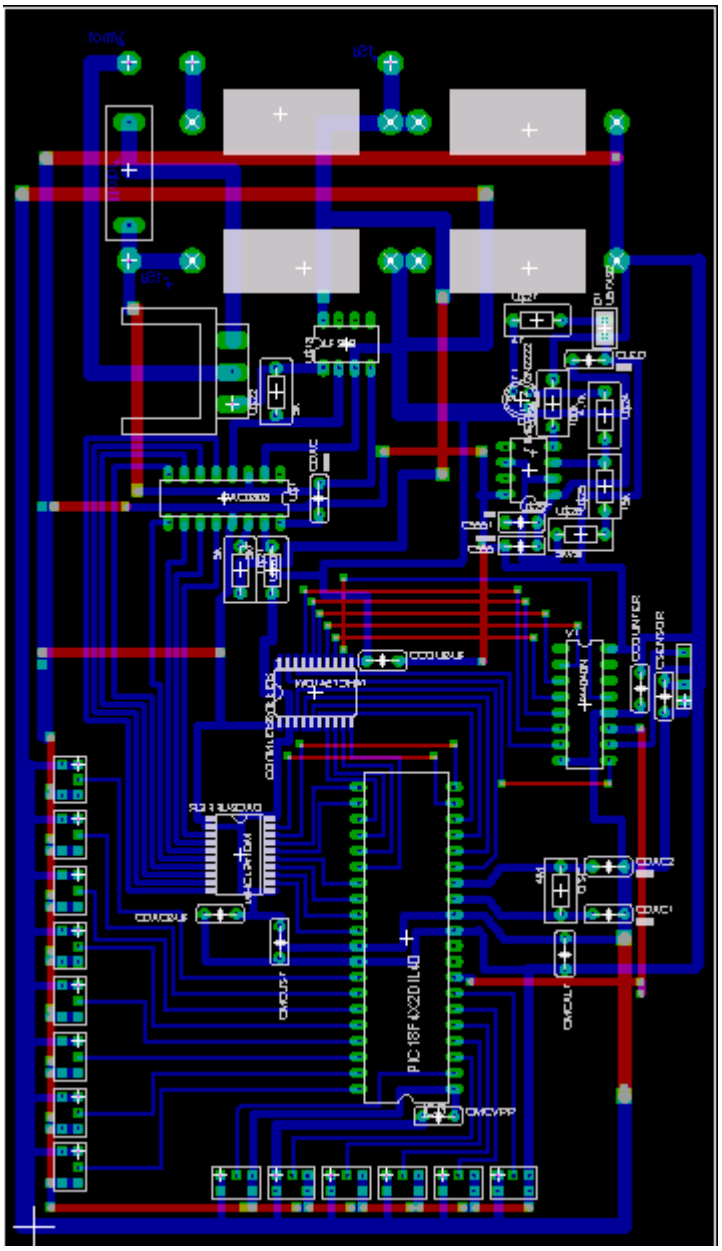


Figure 6. 3 PCB of the motor control circuit

We can examine the circuit in five parts, namely; reaching the digital speed information, determining the actual speed, determining the desired speed and applying necessary feedback, changing the digital data to the analog one as voltage variable, and the motor driver part of the circuit. The component selections and the general characteristics of the components are also included to the sections below.

6.3.1 The Infrared Receiver and Infrared LED

We have to measure and adjust the speed of the motor to control and stabilize it the value that we want it to rotate. To measure the rotating motor's real speed when it is loaded, we need to use a sensor. Therefore we added an Infrared Receiver (V 19 342) and an Infrared LED to the motor speed control board. The constructed Infrared receiver and Infrared LED circuit is important to count the spin of the display board. Both of the receiver and the LED are on the motor speed control board. Since both Infrared Receiver and IR LED are on the motor speed control board, we will use a mirror pasted on the behind of the display board. The test circuit was constructed for the IR sensor and LED couple as seen in the figure below:

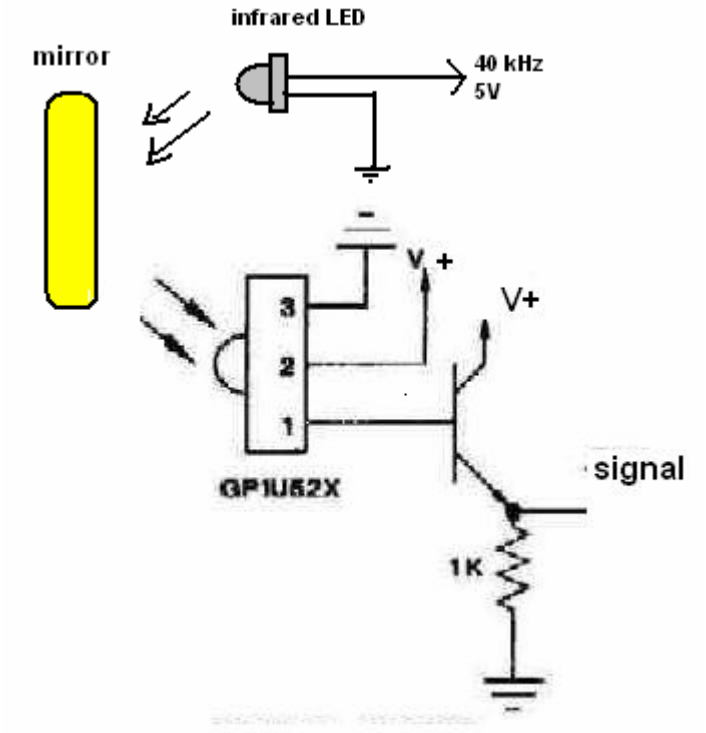


Figure 6. 4 Infrared Receiver[9]

As a result reflecting IR light reaches the receiver and every time the mirror reflects IR light, counter is incremented by one. The signal that comes out from the Infrared Receiver is connected to the clock pin of a 12-bit counter (we use only 7-bits of the counter). By the use of the microcontroller, counter counts the signals till the time period, specified by microcontroller, ends.

[9] www.archer.com

As seen from the figure above, the IR sensor has three connections as Ground, Supply and Data connections. Supply is 5V and data pin is connected to a transistor to employ the transistor as a switch to generate necessary voltage for counter.

Infrared receiver rejects all light sources that are not modulated at 40 kHz, for reliable operation. We designed and tested necessary circuits in the laboratory and added them to motor speed control circuit. We also used the sensor circuit to obtain V_S vs. rpm graph of our motor. We repeated the test with stroboscope to check the reliability of the sensor, the results were consistent. Test results are in the figure below. The test is done when the motor is unloaded.

When the motor was loaded with the main PCB on it and the mirror was behind the PCB, we again measured the speed of the motor using the oscilloscope. Applying nearly 7-8 V to motor leads it to rotate with 1500rpm speed when it is loaded. The figure below is the Speed versus Voltage graph of the motor.

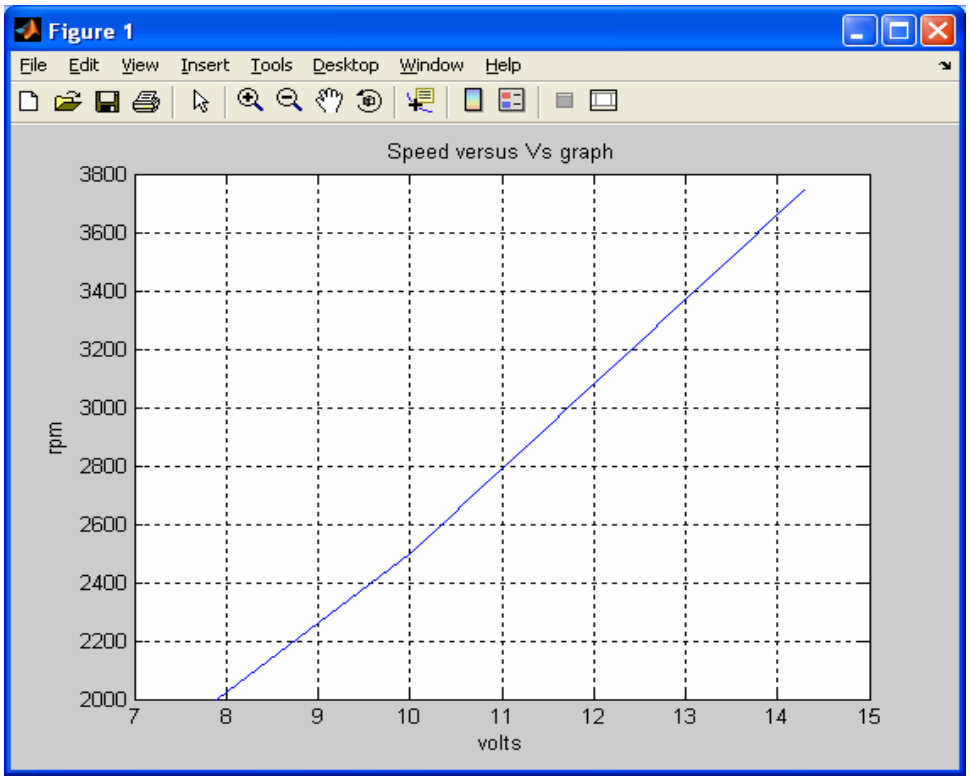


Figure 6. 5 Speed versus voltage graph (measured with sensor)

We will implement the circuit as below to supply 40 kHz 5V signal to the IR LED, we drew its PCB however in the tests we used a signal generator to obtain 40 kHz signal, since we don't have timer (555) to implement 40 kHz 5V signal yet.

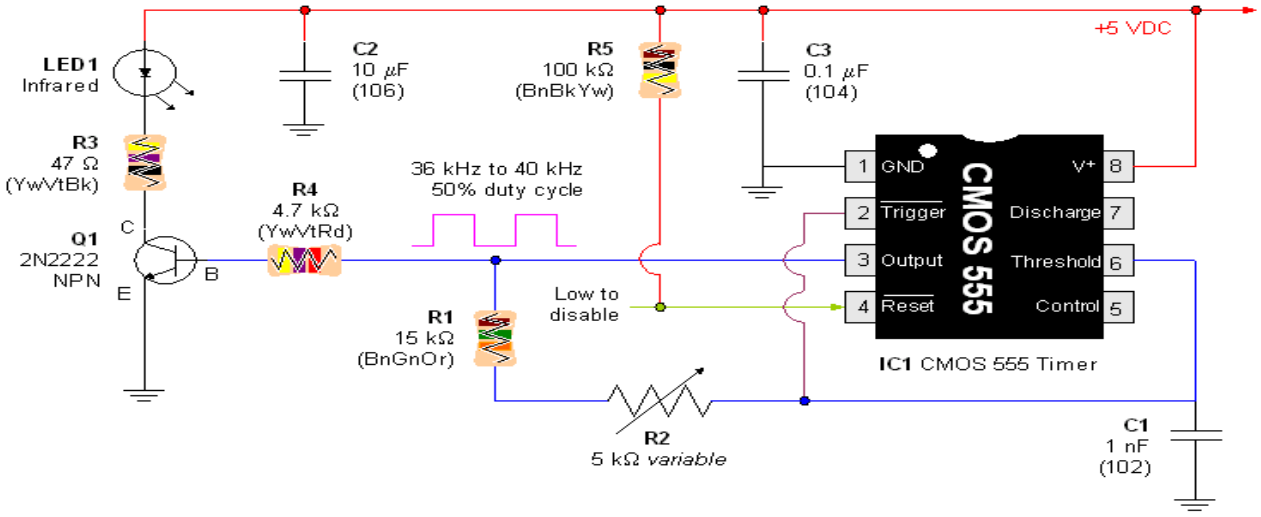


Figure 6. 6 40 kHz. Signal circuit[10]

We used another Infrared Receiver on the display board. Infrared light comes to this sensor from the Infrared LED located on the motor speed control board. We used this Infrared Receiver to reset Ram address counters. We used this reset to start displaying image from a predetermined location. Otherwise the image may be displayed by a phase difference due to original image and another possible error would be a rotating image because of the error in motor speed. If motor runs faster than you want the image starts to rotate counterclockwise, assuming the motor runs in clockwise direction. Due to the location of the Infrared LED, the image data written on RAM is again started to be sent to LED's every time and everywhere the Infrared LED resets counter of RAM.

We encountered a problem about the duration of reset time. As the Infrared Receiver sees the Infrared LED, the counter is in reset mode and don't counts. So the image is not shown in reset time. If the reset time is long, there will be a blank area over the image. Before the reset ends, no data will be sent to LED's. As a result we should decrease the reset duration.

We used the advantage of using double display board. We placed the Infrared Receiver on the first display board (the board that is behind the board which you see when observing the image), therefore Infrared Receiver is not able to see Infrared LED in case you don't open a hole to the second display board. We drilled a small hole on the second display board on the point just between the Infrared Receiver and Infrared LED. This way provides us a decrease in the reset duration.

[10] http://www.robotroom.com/Infrared555.html

6.3.2 The Counter

The Motor's actual speed information is determined nearly in this part of the circuit. The digital pulses coming from the infrared sensor to the clock of the counter is counted by the counter and a digital data is generated for the microcontroller as input. We used a 12-stage binary counter -MM74HC4040, because of its high speed, low power consumption and high noise immunity characteristics. After we tested the counter in the lab with applying pulse to its clock we reach the expected results.

6.3.3 The Microcontroller

We used a microcontroller named 18F452, an 8-bit microcontroller, produced by Microchip. We need a microcontroller because we need to implement an adjustable speed control circuit with closed-loop feedback. The microcontroller provides us to implement a wide variety of feedback control algorithm to our motor speed control board. We chose that microcontroller because of some factors such as

- Availibility to buy in Istanbul
- Higher memory capacity
- Higher number of Input/Output pins

Three ports are defined as input pins in that microcontroller. We used PortA(0-6) as input pins. We used PortA of the 18F452 to read tour count of the rotating motor using output pins of the 4040 counter. PortB is also used as input pins. We used PortB of the 18F452 to determine the period to check RPM of the motor. We used 8 pins of the PortB to determine period in other words we may change period in the order of 1 to 256. In order to apply a check period we implemented a loop count in the software loaded to the Microcontroller. PortC is also used as input. We used PotrC to read the repeat count of the motor in the given check period.

Two ports of the 18F452 are defined as outputs. Portd is used to send digital data to be converted to analog data to the DAC. It is 8-bit data, means all 8 pins of the PortD are used. PortE is also used. Only one pin of the PortE is used. We used this pin to reset counter every time we read repeat count from the counter at the end of check period.

The desired speed of the motor is between 1500 and 3600 RPM related to the oscillator value used in the display board. This value corresponds X repeat in one second. This value is to be written to the Microcontroller via switches connected to the PortA. Then everytime check period ends, current repeat count of the motor is read from counter via PortC. These values are compared. Then the signed difference will be multiplied by a value, and the result will be stored in the microcontroller. This stored value will be sent to DAC via PortD. All pins of PortD are defined as output pins and connected to a DAC. DAC converts this digital data to analog voltage. The resulting current in the output of the Dac is converted to voltae by an opamp. That volage is applied to gate of a Power Mosfet which results in change in current passing through Motor at the same time change in the speed of the motor. Then this procedure repeats itself periodicallt by the time check period ends.

We use PI control algorithm to the microcontroller as described over the figure. It is not PID because we are not considering the difference or change in the speed. Always we look at the speed value and increase or decrease the current passing through the motor to change the speed according to the desired speeed value. As we have the opportunity to change algorithm loaded to the microcontroller, we may implement another algorithm for speed control in case we are not satisfied with the performance of that method.

The assembly code that we implemented PI control to the MicroContrller is given in the Appendix D.

6.3.4 The Digital to Analog Converter

From now on, we start to convert the digital output of the microcontroller to a meaningful expression for our motor by using a digital to analog converter (DAC). Digital-to-analog converters are interfaces between the abstract digital world and analog real life.

We want to compensate the motor's actual speed with the error voltage value, which is calculated with microcontroller as an 8 bit digital data, to reach the expected speed of the motor and keep it constant.

As explained before we benefit from the voltage-speed relation of the motor to change its speed, therefore the error signal from DAC must be a voltage signal which is a physical quantity.

We decided to use the DAC0808 for this implementation of the circuit because of its fast settling time, high speed and low power consumption specifications. The data from microcontroller is used as 8 digital input bits for the DAC. Its output signal is a current signal whose magnitude can be determined also with the Vref inputs of the DAC again. Because, we want to reach a voltage signal for control operation, we stetted the circuit as below to gain the desired voltage compensation value for the motor.

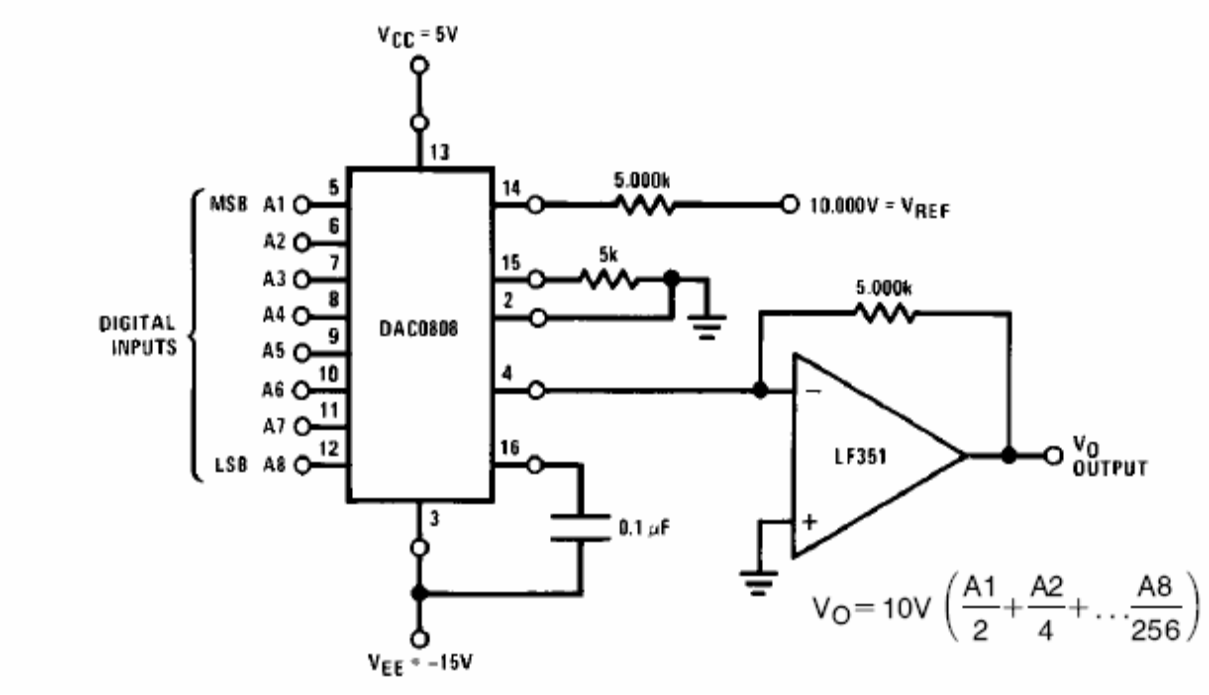


Figure 6. 7 Digital to Analog Converter[11]

While constructing this circuit we used LM353, a general purpose operational amplifier which is suitable for our specs.

6.3.5 The Power Mosfet

The last part of our motor control circuit design we have the motor drive circuit. We used a power mosfet, IRFP244.

First of all, the power mosfet have to work in its linear region for the linear-voltage motor control application and this linear region must be enough long to satisfy our specs. The graph in Figure 6.10 shows the linear transfer characteristics explicitly. Moreover, its high impedance is convenient for adapting it to such a design. According to electrical specifications we connect the motor between the IRFP244's source and -15V.

[11] www.futurlec.com

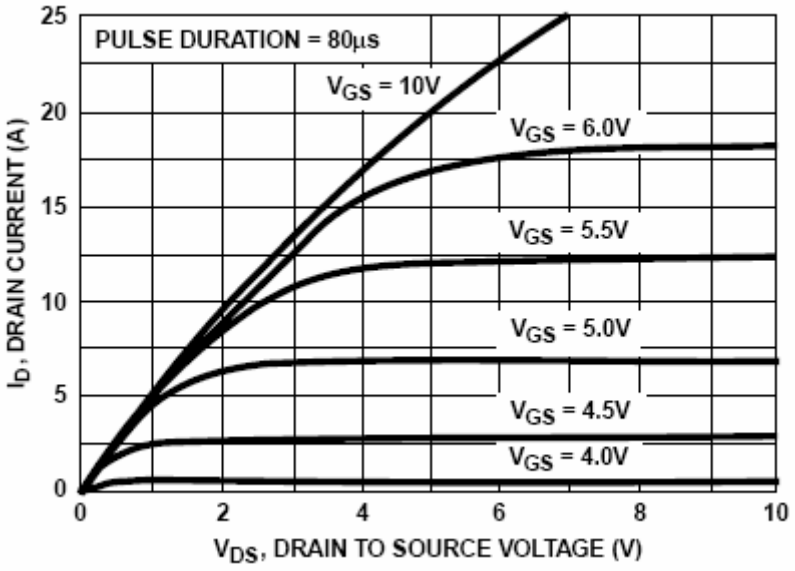


Figure 6. 8 Saturation Characteristics of the Power Mosfet[12]

Changing V_{GS} value we measured speed samples by using stroboscope. Afterwards we see he perfect linear relation between the voltage and the motor's speed. The other important result of this test is that the three motor's speed samples determined from IR sensor test is consistent with our stroboscope test results. This was very joyful for us. We can see the stroboscope test results without load from the graph below.

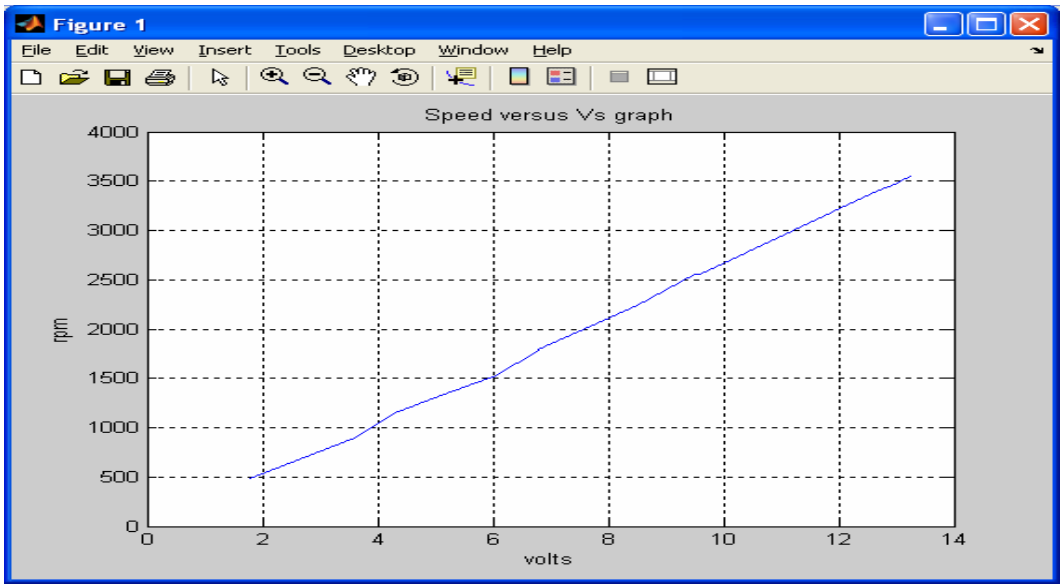


Figure 6. 9 Speed versus voltage graph(measured with stroboscope)

6.4 Conclusion

We must control and so that stabilize its speed for preventing mismatching of the images with their expected positions. One of the most important ways of the

[12] www.futurlec.com

motor control is applying a closed loop feedback control, using linear-voltage method to vary the voltage and current applied to the motor.

Our design for this kind of a circuit, that is shown below is finished. We draw the schematic and PCB of the circuit as shown in the figures 6.2 and 6.3. The figure shown below is the photograph of the Motor Control Board.

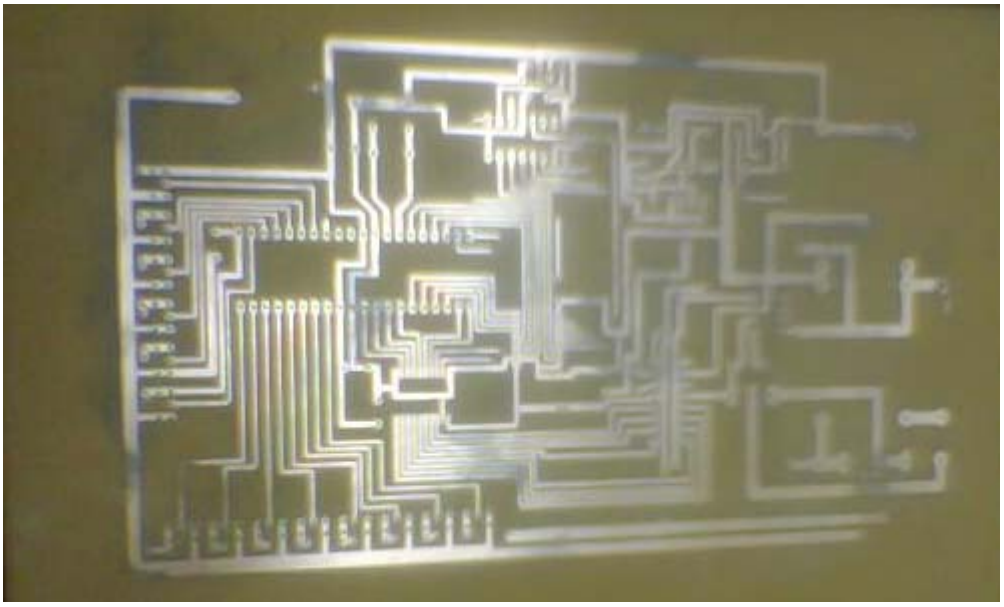


Figure 6. 10 Latest Pcb of the Motor Control Circuit

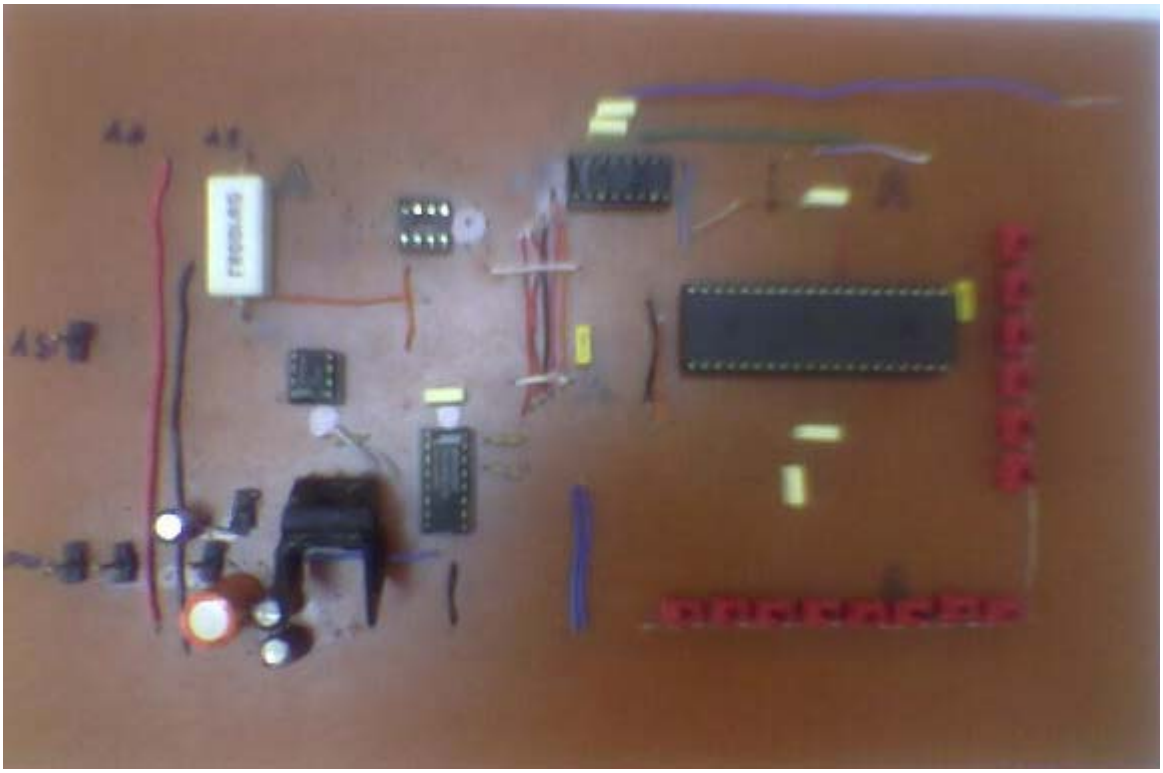


Figure 6. 11 Motor Control Board

CHAPTER 7

Cost Analysis

For this project, we worked as a team of 5 people under the authority of three instructors. All of us are senior students in Boğaziçi University Electrical and Electronics engineering.

For the first semester, each person worked for this project nearly 5 hours per week (for 15 weeks), this means that totally we spared 7 hours per week individually which is totally 105 hours. It can be considered as a reference that the salary of a graduate engineer working for such a project is nearly 12.5YTL per hour. Therefore, it can be calculated that if we worked for such a project as an engineer, we would be able to earn 1312.5YTL individually. Which results in an amount of 1312.5*5 = 6562.5Ytl.

For the second semester, each person worked for this project nearly 6 hours per week, which is totally 15*6 = 90 hours up to now. It can be considered as a reference that the salary of a graduate engineer working for such a project is nearly 12.5YTL per hour and if we consider we are getting experienced we may increase our salary to 15Ytl. Therefore, it can be calculated that if we worked for such a project as an engineer for 15 weeks, we would be able to earn 1350 YTL individually. Which results in an amount of 1350*5 = 6750Ytl.

We have also paid some money for the necessary equipments that will be used in our project.

Test bench	200
300 Smd LEDs(KPHHS-1005SURCK)	33
200 Smd LEDs KPHHS-1005CGCK(V5)	60
150 Smd LEDs KPHHS-1005PBC-A(V4	20
Microcontroller (18F452)	38
Motor (Brushless DC) and PCB stabilizer	40
PCB cost (Spinnıng Board and MCB)	300
Necessary IC's	250
Necessary Equipments	90
First Semester Engineering Work	6562.5
Second Semester Engineering Work	6750
Total	**14343.5**

Moreover we have submitted an application to the EMO's (Elektrik Mühendisleri Odası) project competition with this group and Spinning Display project. We won the support of 400Ytl for our project for the time being.

APPENDICES

APPENDIX A

Electromigration

Electromigration is the transport of material caused by the gradual movement of the ions in a conductor due to the momentum transfer between conducting electrons and diffusing metal atoms. The effect is only important in applications where high direct current densities are used, such as in microelectronics and related structures. As the structure size in electronics such as integrated circuits (ICs) decreases, the practical significance of this effect increases.

Electromigration decreases the reliability of ICs. In the worst case it leads to the eventual loss of one or more connections and intermittent failure of the entire circuit. Since the reliability of interconnects is not only of great interest in the field of space travel and for military purposes but also with civilian applications like for example the anti-lock braking system of cars, high technological and economic values are attached to this effect.

Due to the relatively high life span of interconnects and the short product lifecycle of most consumer ICs, it is not practical to characterize a product's electromigration under real operating conditions. A mathematical equation, the Black's equation, is commonly used to predict the life span of interconnects in integrated circuits tested under "stress", that is external heating and increased current density, and the model's results can be extrapolated to the device's expected life span under real conditions. Such testing is known as High temperature over life (HTOL) testing.

Although electromigration damage ultimately results in failure of the affected IC, the first symptoms are intermittent glitches, and are quite challenging to diagnose. As some interconnects fail before others, the circuit exhibits seemingly random errors, which may be indistinguishable from other failure mechanisms (such as ESD damage.) In a laboratory setting, electromigration failure is readily imaged with an electron microscope, as interconnect erosion leaves telltale visual markers on the metal layers of the IC.

With increasing miniaturization the probability of failure due to electromigration increases in VLSI and ULSI circuits because both the power density and the current density increase. In advanced semiconductor manufacturing processes, copper has replaced aluminium as the interconnect material of choice. Despite its greater fragility in the fabrication process, copper is preferred for its superior conductivity. It is also intrinsically less susceptible to electromigration. However, electromigration continues to be an everpresent challenge to device fabrication, and therefore the EM research for copper interconnects is ongoing (albeit being a relatively new field.)

A reduction of the structure (scaling) by a factor k increases the power density proportional to k and the current density increases by k^2 whereby EM is clearly strengthened.

In modern consumer electronic devices, ICs rarely fail due to electromigration effects. This is because proper semiconductor design practices incorporate the effects of electromigration into the IC's layout. Nearly all IC design houses use automated EDA tools to check and correct electromigration problems at the transistor layout-level. When operated within the manufacturer's specified temperature and voltage range, a properly designed IC-device is more likely to fail from other (environmental) causes, such as cumulative damage from gamma-ray bombardment.

Nevertheless, there have been documented cases of product failures due to electromigration. In the late 1980s, one line of Western Digital's desktop drives suffered widespread, predictable failure 12-18 months after field usage. Using forensic analysis of the returned bad units, engineers identified improper design-rules in a third-party supplier's IC controller. By replacing the bad component with

that of a different supplier, WD was able to correct the flaw, but not before significant damage to the company's reputation.

Overclocking of processors, especially when using higher than nominal voltage, causes electromigration between their transistors and significantly shortens the chips' lifetime.

Electromigration can be a cause of degradation in some power semiconductor devices such as low voltage power MOSFETs, in which the lateral current flow through the source contact metallisation (often aluminium) can reach the critical current densities during overload conditions. The degradation of the aluminium layer causes an increase in on-state resistance, and can eventually lead to complete failure.[13]

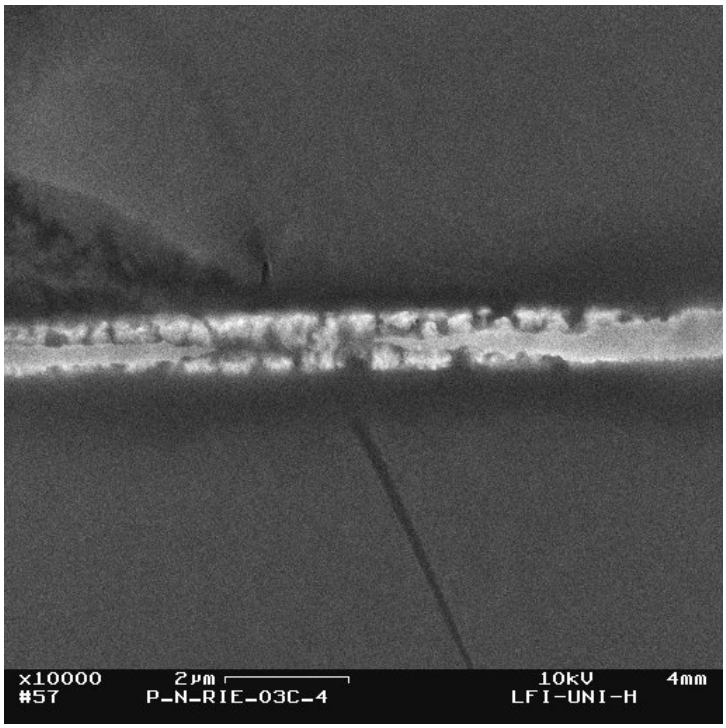


Figure A1: Failure of a copper conductive strip due to electromigration, viewed with a scanning electron microscope.[14]

[13] *http://en.wikipedia.org/wiki/Electromigration*
[14] *[http://en.wikipedia.org/wiki/Image:Leiterbahn_ausfallort_elektromigration.jpg]*

APPENDIX B

Program Codes

The Code for the program for transferring data to the device:

PortInterop.cs Contents:

```
using System;
using System.Runtime.InteropServices;

public class PortAccess
{
		[DllImport("inpout32.dll", EntryPoint="Out32")]
		public static extern void Output(int adress, int value);
		[DllImport("inpout32.dll", EntryPoint = "Inp32")]
		public static extern int Input(int adress);
}
```

Program.cs Contents:

```
using System;
using System.Collections.Generic;
using System.Windows.Forms;

namespace SpinningDisplayP
{
		static class Program
		{
	/// <summary>
	/// The main entry point for the application.
	/// </summary>
	[STAThread]
		static void Main()
		{
			Application.EnableVisualStyles();
		Application.SetCompatibleTextRenderingDefault(false);
			Application.Run(new Form1());
		}
		}
}
```

Form1.cs Contents:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.Threading;
using System.IO;

namespace SpinningDisplayP
{

    public partial class Form1 : Form
    {
```

```
        Display Spinning;

        public Form1()
        {
            Spinning = new Display();
            InitializeComponent();
            Form1.CheckForIllegalCrossThreadCalls = false;
            //Random ali = new Random();
            //FileStream yusuf = new FileStream("c:\\yusuf.txt",
FileMode.Create);
            //StreamWriter armut = new StreamWriter(yusuf);

            //armut.WriteLine("Ram1 Data");
            //armut.WriteLine("512000");
            //for (int i = 0; i < 512000; i++)
            //{
            //    armut.WriteLine(ali.Next(255));
            //}
            //armut.WriteLine("Ram2 Data");
            //armut.WriteLine("512000");
            //for (int i = 0; i < 512000; i++)
            //{
            //    armut.WriteLine(ali.Next(255));
            //}

            //armut.Close();
            //yusuf.Close();

        }

        private void SearchButton_Click(object sender, EventArgs e)
        {
            PortSearch.RunWorkerAsync("");
        }

        private void PortSearch_DoWork(object sender, DoWorkEventArgs
e)
        {
            SearchButton.Enabled = false;
            Spinning.SearchForConnections(ConnectionStatus);
            SearchButton.Enabled = true;

            if (Spinning.DataStatus == Status.DataLoaded &&
Spinning.ConnectionStatus == Status.Connected)
            {
                TransferPanel.Enabled = true;
            }

        }

        private void Prepare_Click(object sender, EventArgs e)
        {
            Spinning.PreparePort();
            ConnectionPanel.Enabled = true;
        }

        private void LoadData_Click(object sender, EventArgs e)
        {
            Spinning.GetDataByDialog(openFileDialog1);
        }
```

```
            private void openFileDialog1_FileOk(object sender,
CancelEventArgs e)
            {
                Spinning.GetData(openFileDialog1);
                if (Spinning.DataStatus == Status.DataLoaded)
                {
                    DataLoadedLabel.Text = "Data Loaded";
                }
                if (Spinning.DataStatus == Status.DataLoaded &&
Spinning.ConnectionStatus == Status.Connected)
                {
                    TransferPanel.Enabled = true;
                }
            }

            private void TransferButton_Click(object sender, EventArgs e)
            {
                ConnectionPanel.Enabled = false;
                DataPanel.Enabled = false;
                TransferPanel.Enabled = false;
                TransLabel.Visible = true;
                DataTransfer.RunWorkerAsync();
            }

            private void DataTransfer_DoWork(object sender,
DoWorkEventArgs e)
            {
                Spinning.TransferData();
            }

            private void DataTransfer_RunWorkerCompleted(object sender,
RunWorkerCompletedEventArgs e)
            {
                ConnectionPanel.Enabled = true;
                DataPanel.Enabled = true;
                TransferPanel.Enabled = true;
                TransLabel.Visible = false;
            }
        }

        public enum Status
        {
            Connected, NotConnected,TryingToConnect, Transferring,
DataNotLoaded, DataLoaded
        }

        public class Display
        {
            public Display()
            {
                ConnectionStatus = Status.NotConnected;
                DataStatus = Status.DataNotLoaded;
                WriteAddress = 0x378;
                ReadAddress = 0x379;
                ControlAddress = 0x37A;
                //WriteAddress = 0x03bc;
                //ReadAddress = WriteAddress + 1;
                //ControlAddress = WriteAddress + 2;
                SearchTimeOut = 1000;
                SearchStep = 10;
```

```
                OneThirdCycle = 1;


            }

            public int PreparePort()
            {
                PortAccess.Output(ControlAddress, 14 ^ 0x0b); // clock
reset 1110
                return 1;
            }

            public int GetDataByDialog(OpenFileDialog openFileDialog1)
            {
                if (DataStatus == Status.DataNotLoaded)
                {
                    openFileDialog1.ShowDialog();
                }
                return 1;
            }

            public int TransferData()
            {
                int i;

                DateTime Current = DateTime.Now;

                PortAccess.Output(ControlAddress, 14 ^ 0x0b); // clock
reset 1110

                for (i = 0; i < Ram1Data.Length; i++)
                {
                    PortAccess.Output(ControlAddress, 12 ^ 0x0b); //
clock sonu 1100
                    PortAccess.Output(WriteAddress,Ram1Data[i]); // Data
hazır
                    //Thread.Sleep(OneThirdCycle); // Az bekle az sonra
ram açılacak
                    PortAccess.Output(ControlAddress, 0 ^ 0x0b); // 1.
rami aç controllere 0000 ver
                    //Thread.Sleep(OneThirdCycle); // az bekle yazmasını
                    PortAccess.Output(ControlAddress, 12 ^ 0x0b); //
kapat şimdi rami 1100
                    //Thread.Sleep(OneThirdCycle); //biraz bekle
                    PortAccess.Output(ControlAddress, 13 ^ 0x0b); // ram
kapalıyken bi clock ver de bi sonraki adrese geçelim
                    // yani 1101
                }

                //Counterı resetle bakalım
                PortAccess.Output(ControlAddress, 14 ^ 0x0b); // clock
reset 1110
                //Thread.Sleep(OneThirdCycle);

                for (i = 0; i < Ram2Data.Length; i++)
                {
                    PortAccess.Output(ControlAddress, 12 ^ 0x0b); //
clock sonu 1100
                    PortAccess.Output(WriteAddress, Ram2Data[i]); // Data
hazır
```

```
                    //Thread.Sleep(OneThirdCycle); // Az bekle az sonra
ram açılacak
                    PortAccess.Output(ControlAddress, 4 ^ 0x0b); // 2.
rami aç controllere 0100 ver
                    //Thread.Sleep(OneThirdCycle); // az bekle yazmasını
                    PortAccess.Output(ControlAddress, 12 ^ 0x0b); //
kapat şimdi rami 1100
                    //Thread.Sleep(OneThirdCycle); //biraz bekle
                    PortAccess.Output(ControlAddress, 13 ^ 0x0b); // ram
kapalıyken bi clock ver de bi sonraki adrese geçelim
                    // yani 1101
                }

                PortAccess.Output(ControlAddress, 14 ^ 0x0b); // clock
reset 1110

                for (i = 0; i < Ram3Data.Length; i++)
                {
                    PortAccess.Output(ControlAddress, 12 ^ 0x0b); //
clock sonu 1100
                    PortAccess.Output(WriteAddress, Ram3Data[i]); // Data
hazır
                    //Thread.Sleep(OneThirdCycle); // Az bekle az sonra
ram açılacak
                    PortAccess.Output(ControlAddress, 8 ^ 0x0b); // 3.
rami aç controllere 1000 ver
                    //Thread.Sleep(OneThirdCycle); // az bekle yazmasını
                    PortAccess.Output(ControlAddress, 12 ^ 0x0b); //
kapat şimdi rami 1100
                    //Thread.Sleep(OneThirdCycle); //biraz bekle
                    PortAccess.Output(ControlAddress, 13 ^ 0x0b); // ram
kapalıyken bi clock ver de bi sonraki adrese geçelim
                    // yani 1101
                }

                PortAccess.Output(ControlAddress, 14 ^ 0x0b); // clock
reset 1110

                TimeSpan Passed = DateTime.Now - Current;

                MessageBox.Show("The operation completed in " +
Passed.Seconds.ToString() + " seconds.");
                return 1;

            }

            public int GetData(OpenFileDialog openFileDialog1)
            {
                if (File.Exists(openFileDialog1.FileName))
                {
                    FileStream FileHandle = new
FileStream(openFileDialog1.FileName, FileMode.Open);
                    StreamReader FileReader = new
StreamReader(FileHandle);

                    String Current = "";
                    Current = FileReader.ReadLine();
                    if (Current == "Ram1 Data")
                    {
                        int NofData = 0;
```

```
                        try
                        {
                            NofData =
Convert.ToInt32(FileReader.ReadLine());
                            if (NofData == 0)
                            {
                                DataError();
                            }
                            Ram1Data = new int[NofData];
                            for (int i = 0; i < NofData; i++)
                            {
                                Ram1Data[i] =
Convert.ToInt32(FileReader.ReadLine());
                            }
                            Current = FileReader.ReadLine();
                            NofData =
Convert.ToInt32(FileReader.ReadLine());
                            if (NofData == 0)
                            {
                                DataError();
                            }
                            Ram2Data = new int[NofData];
                            for (int i = 0; i < NofData; i++)
                            {
                                Ram2Data[i] =
Convert.ToInt32(FileReader.ReadLine());
                            }
                            NofData =
Convert.ToInt32(FileReader.ReadLine());
                            if (NofData == 0)
                            {
                                DataError();
                            }
                            Ram3Data = new int[NofData];
                            for (int i = 0; i < NofData; i++)
                            {
                                Ram3Data[i] =
Convert.ToInt32(FileReader.ReadLine());
                            }
                            DataStatus = Status.DataLoaded;
                            FileHandle.Close();
                            FileReader.Close();
                            return 1;

                        }
                        catch (Exception)
                        {
                            Ram1Data = null;
                            Ram2Data = null;
                            Ram3Data = null;

                            DataError();
                            FileHandle.Close();
                            FileReader.Close();
                            return 0;
                        }
                    }
                    else
                    {
                        DataError();
                        return 0;
```

```
                    }
                }
                else return 0;
            }

            public void DataError()
            {
                MessageBox.Show("There was an error reading data!");
            }

            public int SearchForConnections(Label ConnectionStatusLabel)
            {
                if (ConnectionStatus == Status.NotConnected)
                {
                    int CurrentTime = 0;
                    int Data;
                    int i = 1;
                    while (ConnectionStatus == Status.NotConnected &&
CurrentTime < SearchTimeOut)
                    {
                        ConnectionStatusLabel.Text = "Connecting... Try "
+ i.ToString();

                        Data = PortAccess.Input(ReadAddress);

                        //MessageBox.Show(Data.ToString());
                        if (((Data ^ 0x80)&0x40) == 0x40) //D6 bit is 0
                        {
                            ConnectionStatus = Status.Connected;
                        }
                        else
                        {
                            Thread.Sleep(SearchStep);
                            CurrentTime += SearchStep;
                        }
                        i++;
                    }

                    if (ConnectionStatus == Status.NotConnected)
                    {
                        ConnectionStatusLabel.Text = "Could not connect
to the device, check the connection";
                        return 0;
                    }
                    else
                    {
                        ConnectionStatusLabel.Text = "Connected";
                        return 1;
                    }
                }
                else return -1;
            }

            public Status ConnectionStatus;
            public Status DataStatus;

            public int WriteAddress;
            public int ReadAddress;
            public int ControlAddress;

            public int SearchTimeOut;
```

```
        public int SearchStep;

        public int[] Ram1Data;
        public int[] Ram2Data;
        public int[] Ram3Data;

        int OneThirdCycle;

    }
}
```

APPENDIX C

Brushless DC Motor

BLDC motors are a type of synchronous motor. This means the magnetic field generated by the stator, and the magnetic field generated by the rotor rotate at the same frequency.

The stator of a BLDC motor consists of stacked steel laminations with windings placed in the slots that are axially cut along the inner periphery. Depending upon the control power supply capability, the motor with the correct voltage rating of the stator can be chosen.

The rotor is made of permanent magnet and can vary from two to eight pole pairs with ternate North (N) and South (S) poles.

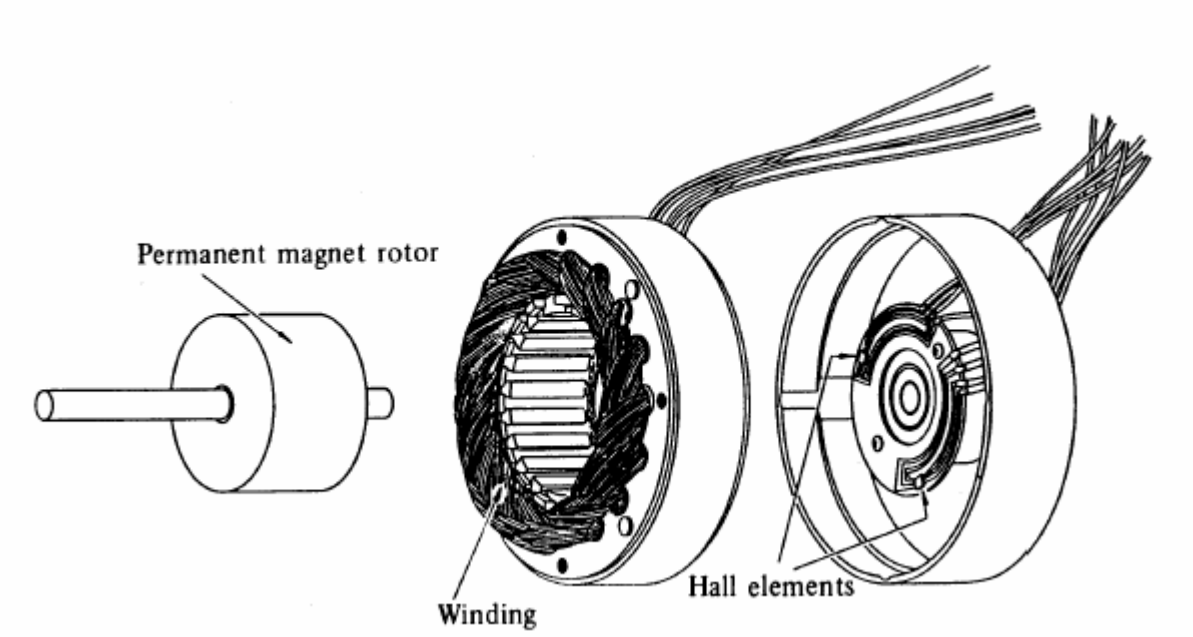


Figure C.1: a sample of BLDC[15]

Unlike a brushed DC motor, the commutation of a BLDC motor is controlled electronically. To rotate the BLDC motor, the stator windings should be energized in a sequence. It is important to know the rotor position in order to understand which winding will be energized following the energizing sequence. Rotor position is sensed using Hall effect sensors embedded into the stator.

The speed can be controlled in a closed loop by measuring the actual speed of the motor. The error in the set speed and actual speed is calculated. A Proportional plus Integral plus Derivative (P.I.) controller can be used to amplify the speed error. When a BLDC motor rotates, each winding generates a voltage known as back Electromotive Force or back EMF, which opposes the main voltage supplied to the windings according to Lenz's Law. The polarity of this back EMF is in opposite direction of the energized voltage.

---

[15] T. Kenjo, "Permanent magnet and brushless dc motors", Oxford, 1985

The potential difference across a winding can be calculated by subtracting the back EMF value from the supply voltage. The motors are designed with a back EMF constant in such a way that when the motor is running at the rated speed, the potential difference between the back EMF and the supply voltage will be sufficient for the motor to draw the rated current and deliver the rated torque. If the motor is driven beyond the rated speed, back EMF may increase substantially, thus decreasing the potential difference across the winding, reducing the current drawn which results in a drooping torque curve. The last point on the speed curve would be when the supply voltage is equal to the sum of the back EMF and the losses in the motor, where the current and torque are equal to zero[16].

[16] www.microchip.com/downloads/en/AppNotes

APPENDIX D

MicroController Program Codes

```
;              Assembly Code for PI Motor Control

    list p=18F452, n=48, t=ON, st=OFF
    #include "p18F452.inc"



;-----------------BIT DEFINITIONS-----------------------------------
    cblock  0x20            ;start of general purpose registers
        TUR                 ;is the Register that stores repeat count of the motor in check period
        REFTUR              ;is the Register that stores desired repeat count for check period determined by PortA
        ZAMAN               ;is the Register that stores switch data to determine check period determined by PortB
        SPEED               ;is the Register that stores the data to determine the current to be applied to the motor
        LPC                 ;is the Register2 to determine check period
    endc
;-----------------VECTORS-------------------------------------------

    org 0x000000        ; reset vector
    bra START

;--------------------PROGRAM-----------------------------------
START

    CALL    INIT            ;go to initilization
    MOVLW   0x30
    MOVWF   SPEED


MLOOP
    MOVFF   PORTA,REFTUR
    MOVFF   PORTB,ZAMAN
    CLRF    PORTE           ;enables counter to start counting

;           Loop to wait till check period
LOOP1
    MOVLW   0xFF
    MOVWF   LPC
LOOP2
    DECFSZ  LPC,1,0
    GOTO    LOOP2
```

```
;           End of check period

    CLRF    PORTC
    MOVFF   PORTC,TUR       ;takes input to get repeat count information

    MOVF    TUR,0,0
    SUBWF   REFTUR,0,0
    BTFSS   STATUS,C
    CALL    EKSI,0

    MOVF    TUR,0,0
    SUBWF   REFTUR,0,0
    BTFSC   STATUS,C
    CALL    ARTI,0

    CLRF    PORTD
    MOVFF   SPEED,PORTD     ;gives output to Dac to drive Motor
    MOVLW   0x01
    MOVWF   PORTE           ;resets counter for new Loop


    goto    MLOOP
;                           Stars the process again
```

```
;                                   Initialization of Ports
INIT
     CLRF     TRISD                 ;Set Ports as outputs
     CLRF     TRISE
     MOVLW    0xFF                  ;Set Ports as inputs
     MOVWF    TRISA
     MOVWF    TRISB
     MOVWF    TRISC
     return   1

;                                   Subroutine to increase current
ARTI
     INCF     SPEED,1,0
     BTFSC    STATUS,Z
     DECF     SPEED,1,0
     RETURN   1

;                                   Subroutine to decrease current
EKSI
     DECF     SPEED,1,0
     BTFSC    STATUS,Z
     INCF     SPEED,1,0
     RETURN   1


     END
;                                   End of Program
```