EAST WEST UNIVERSITY

"Microcontroller Based Color Analyzer With Computer Interfacing"

By

Sabbir Masqur Alam

Md. Nasim Ahmed Khan

Md. Abul Hasan

In partial fulfillment of the requirements for the degree of Bachelor of Science in Electrical and Electronic Engineering

Fall, 2012

Department of Electrical and Electronic Engineering

Faculty of science and engineering

East West University

East West University

Department of EEE

Microcontroller Based Color Analyzer with Computer Interfacing

By

Sabbir Masqur Alam: 2008-2-80-036

Md. Nasim Ahmed Khan: 2008-2-80-023

Md. Abul Hasan: 2008-2-80-027

Submitted to the

Department of Electrical and Electronic Engineering
Faculty of Science and Engineering

East West University
Dhaka, Bangladesh

In partial fulfillment of the requirement for the degree of Bachelor of Science in
Electrical and Electronic Engineering (B.Sc. in EEE)

Fall 2012

Approved by

……………………….	………………………..
Project Advisor	Department Chairperson
Dr. Muhammed Mazharul Islam	Dr. Mohammad Mojammel Al-Hakim

Approval

The project work titled "Microcontroller Based Color Analyzer with Computer Interfacing" submitted by **Sabbir Masqur Alam (2008-2-80-036), Md. Nasim Ahmed Khan (2008-2-80-023)** and **Md. Abul Hasan (2008-2-80-027)**, session Fall, 2012, has been accepted as satisfactory in partial fulfillment of the requirement for the degree of Bachelor of Science in Electrical and Electronic Engineering.

……………………………………….

Dr. Mohammad Mojammel Al-Hakim
Professor and Chairperson
Department of Electrical and Electronic Engineering
East West University
Dhaka, Bangladesh

Acknowledgment

First of all, we are grateful to the almighty Allah for giving us this opportunity to complete the project.

We would like to thank Dr. Muhammed Mazharul Islam, Assistant Professor, Department of Electrical and Electronic Engineering (EEE), East West University (EWU), Dhaka, our supervisor, for his constant guidance, supervision, constructive suggestions and constant support during this project work.

We are grateful to Dr. Mohammad Mojammel Al-Hakim, Associate Professor and Chairperson, Dr. Anisul Haque, Professor, and Dr. Halima Begum, Assistant Professor, Department of Electrical and Electronic Engineering (EEE), East West University (EWU), for their valuable suggestions and help. We would also like to thank our advisors Mr. Md. Niazul Islam Khan and Mr. Fakir Mashuque Alamgir, other faculty members, office staffs and EWU in general.

At last we want to thank our parents and all our friends for their moral support and helpful discussion during this work.

Authorization

We hereby declare that we are the sole authors of this project. We authorize East West University to lend this project to other institutions or individuals for the purpose of scholarly research.

………………………	……………………………	…………………
Sabbir Masqur Alam	**Md. Nasim Ahmed Khan**	**Md. Abul Hasan**

We further authorize East West University to reproduce this project report by photocopy or other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

………………………	……………………………	…………………
Sabbir Masqur Alam	**Md. Nasim Ahmed Khan**	**Md. Abul Hasan**

East West University
Dhaka, Bangladesh
December, 2012

Table of Contents

List of Figures

List of Tables

Abstract

In textiles and garment related industries, it is often needed to dye fabrics with a particular color. The dye color is usually composed using some base or primary colors (such as red, green and blue). However, it is difficult to identify the exact ratio of primary colors that combine to form a particular color using our eyes. Therefore, to compose the color, many trials and errors are required, which takes huge amount of time and resources. If a system can be developed that can identify the primary color constituents of a color, then the dye could be easily prepared.

In this project, we analyzed the red-green-blue (RGB) components of a light (reflected from any colored plain surface) using a color sensor and then display and save the results in a computer. We used a color sensor (TCS230) for analyzing the color, which converts color intensity to frequency. More specifically, its output is a square wave whose frequency is directly proportional to the intensity of light incident on the sensor surface. The RGB components are analyzed using the principles of light reflectivity and some color filters. The computer interfacing is done through Universal Serial Bus (USB). USB was chosen considering the drawbacks of other standard interfacing techniques. We also developed a GUI to display the data on screen and the software stores the analyzed data on a text file for future usage.

Chapter 1 : Introduction

1.1 What is a Color Analyzer?

It is often required to dye fabrics with a particular color in textile, printing and other garment related industries. Many custom colors and paints are not readily available off-the shelf in the market. Therefore, the color has to be manufactured using a mixture of some base or primary colors. However, to be able to properly produce a color, it is necessary to understand the ratio of the constituent primary colors of a particular color. This is usually done manually by a person who understands what ratio of primary color combination will produce a particular color. This process still requires a fair amount of trial & error and is therefore both time and resource consuming. Furthermore, depending upon the expertise of that person, the reproduced color might not exactly match the required color.

According to color theory, every color can be represented as a combination of three primary colors- red (R), green (G), and blue (B). These are considered as primary colors because human eyes are sensitive to these three colors. The wavelengths of light that corresponds to these primary colors are 470nm (blue), 530nm (green) and 700nm (red) [1]. We sense the color of an object if light is reflected from that object body. Since the reflected light contains a mixture of the three primary colors, therefore, we associate that color composed of the RGB mixture as the object's characteristic color.

In Figure 1.1, the additive RGB color model is shown. When red, green and blue colored light beam are superimposed, then a white color is formed. The RGB color model is additive in the sense that the three light beams are added together [2].

In our project, we developed a system which analyzes a particular color into its constituent primary colors. The system then sends the data to computer where it is viewed using our developed software and graphical user interface (GUI) and finally, saves the data in the computer's hard disk.

To analyze the color, we used a color sensor (TCS230). This is a programmable, color intensity to frequency converter manufactured by Texas Advanced Optoelectronic Solutions (TAOS). The sensor basically outputs a square wave of 50% duty cycle whose frequency is directly proportional to the intensity of the light falling on the sensor surface.

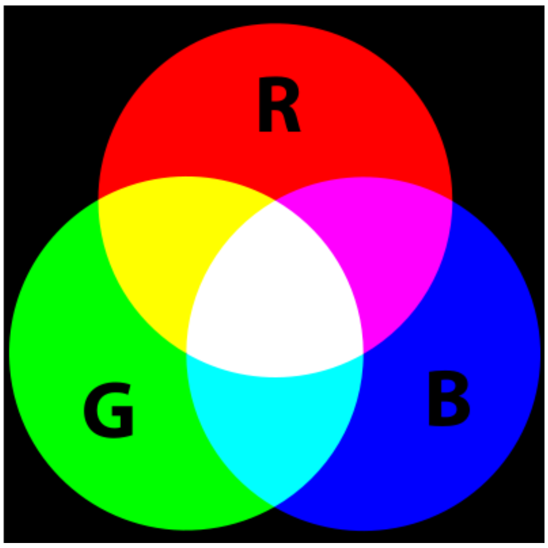


Figure 1.1 : RGB color model [2]

1.2 Features of the Proposed Color Analyzer

- **Color sensor:** The color sensor (TCS230) is used to obtain the three primary color components of the incident light on the sensor. The sensor has built-in photodiode arrays, whose output current is proportional to the incident light intensity. Through an internal oscillator, the current is converted to frequency. The sensor can also selectively apply red, green or blue filters to the incident light. Therefore, the incident light on the sensor surface can be analyzed into its constituent primary colors.

- **Microcontroller:** The microcontroller (PIC18F4550) is used here to measure the output frequency of the sensor and to handle the interface with a personal computer.

- **Computer interface:** The microcontroller sends the measured color frequencies to a personal computer using Universal Serial Bus (USB) port. The USB is a port available in computers and many electronic devices as standard interfacing port. It has many advantages over parallel or serial ports such as requiring fewer numbers of pins, having higher data transfer rates etc.

- **Software and graphical user interface (GUI):** Using our developed software, the measured color frequencies are displayed on the computer screen. The software also controls the data acquisition process. A GUI was created for easier handling of the data acquisition process. Finally, the software also stores the data on the computer's hard drive for record keeping and any further processing.

Chapter 2 : Hardware of Color Analyzer

2.1 Working Principle

Based on the principle of light reflection, when light is incident into a colored object then some amount of lights are absorbed and some are reflected. For example, a pure blue colored object will reflect only the blue portion of color spectrum and other colors will be absorbed [3]. Considering this, we use white LEDs as the light source. The light from the LEDs are incident on the colored object. Since white color contains all the fundamental colors, the object will absorb all the colors except its own color combination. The reflected light will have the red, green and blue (RGB) color contents of that object.

2.2 Major Components of the System

Our proposed color analyzer system contains the following major components,

- A color sensor (TCS230) to convert light intensity to frequency.
- A microcontroller (PIC18F4550) to measure the frequency, to control the data acquisition and to send data to the computer.
- White LEDs as light source.
- 7 segment displays and display drivers.

2.2.1 TCS230 Color Sensor [4]

Main Features of TCS230:

- High-resolution conversion of light intensity to frequency
- Programmable color filter selection and full-scale output frequency
- Digital output and communicates directly with a microcontroller
- Single-supply operation (2.7 V to 5.5 V)
- Low-profile surface-mount package

The color sensor TCS230 is used to extract the RGB contents. It is a small, highly integrated device package for color sensing purpose. It is an eight (8) pin small outline integrated circuit (SOIC). It outputs a square wave whose frequency is directly proportional to the intensity of light falling on the sensor surface. It is used in many color sensing applications [5]. The TCS230 color sensor has built in RGB color filters. Using the three filters, we can selectively allow only one color component to be detected by the sensor.

Figure 2.1 shows the pin diagram of the sensor. The pins S_0, S_1 are used for selecting the output scale factor or power shutdown mode, S_2, S_3 are used to select the type of filter, OE is the output enable pin, OUT is the frequency output pin, GND is the chip ground pin, VCC is

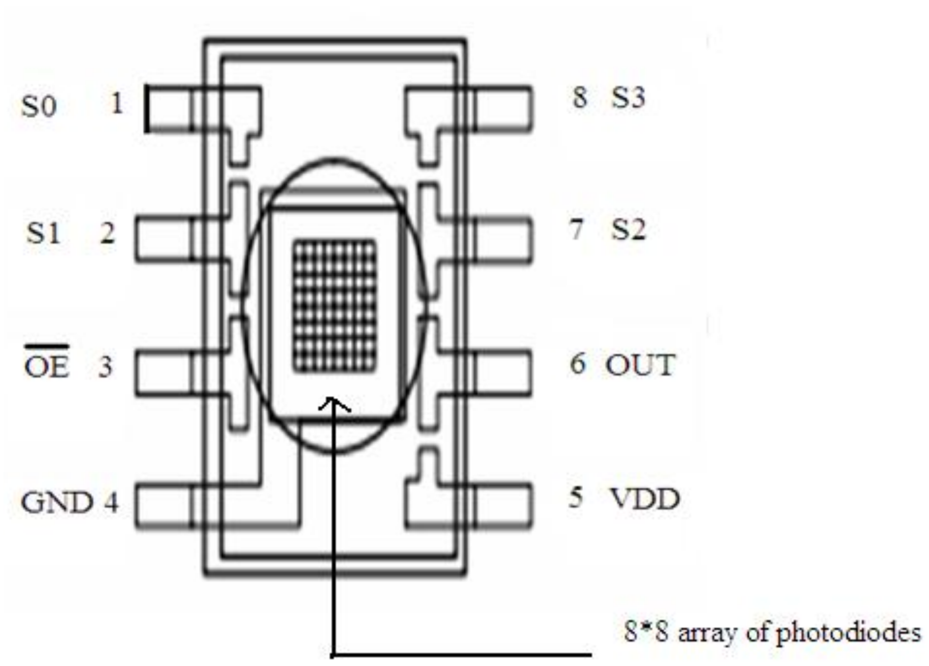


Figure 2.1: TCS 230 pin diagram

the operating voltage for the chips.

Functional Block Diagram:

Figure 2.2 shows the functional block diagram of the color sensor. The sensor contains mainly two components, the photodiode array and the current to frequency converter.

Photodiode Array [6]:

A silicon photodiode is a p-n junction capable of converting light into either current or voltage, depending upon the mode of operation. The common, traditional solar cell which is used to generate electric solar power is a large area photodiode. The mechanism of a photodiode is known as the inner photoelectric effect [6]. When a photon of sufficient energy is incident on the diode, it excites an electron, thereby creating a free electron (and a positively charged electron hole).

In Figure 2.1, the photodiode array is shown at the center of the sensor. The sensor has 64 photodiodes in total, out of which, 16 photodiodes have blue filters, 16 have green filters, 16 have red filters, and 16 photodiodes are clear with no filters. The filters can be selected using the pins S_2 and S_3. When the red filter is applied to the incident light on the sensor, the sensor lets the red color to pass through and blocks the other colors. Similarly when blue (or green) filters are applied, the sensor lets the blue (or green) color to pass through and blocks the other colors.

When the light reflected from any colored object is incident on the photodiode array of the color sensor, for each filter, the photodiode array produces a current which is directly proportional to the intensity of that filtered light.

Current to Frequency Converter:

The sensor uses current to frequency converter to convert the output current from the array of photodiodes to a proportional frequency. The frequency can be sampled at 100%, 20% or 2%, using the pins S_0 and S_1.

The sensor gives a square wave as output which has 50% duty cycle. The frequency of the square wave is therefore, directly proportional to light intensity. It allows digital inputs and digital output so that it can be directly interfaced to a microcontroller or other logic circuitry.

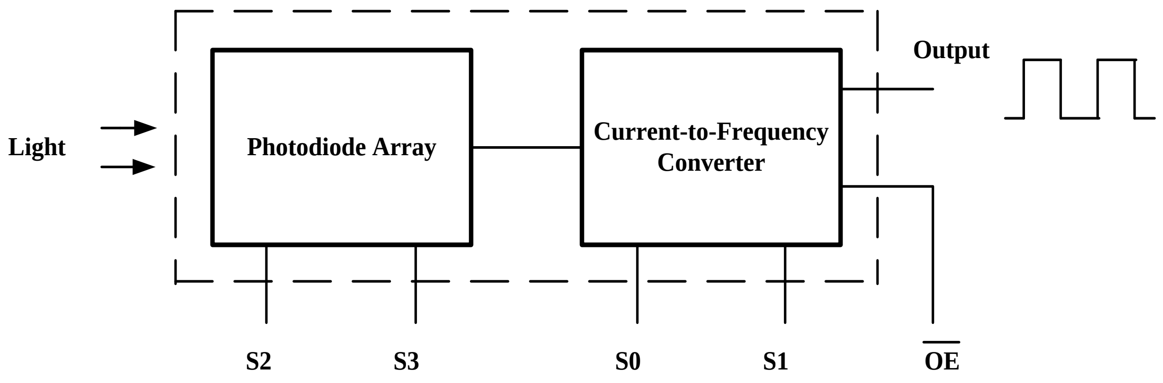


Figure 2.2: Functional block diagram of TCS230

Tables 2.1 and 2.2 summarize the pin operations of the sensor.

Table 2.1 : Pin function of TCS230 for output frequency sampling

State of pin S_0	State of pin S_1	Output frequency sampling
Low	Low	No output
Low	High	2%
High	Low	20%
High	High	100%

Table 2.2 : Pin function of TCS230 for selecting different filters

State of pin S_2	State of pin S_3	Selected filter
Low	Low	Red
Low	High	Blue
High	Low	Clear (no filter)
High	High	Green

2.2.2 PIC18F4550 Microcontroller

The microcontroller is used in our project to measure the frequency, control the data acquisition process and to send the data to the computer. It has many desirable features such as a built in USB transceiver for computer interfacing and several 8 bit and 16 bit timers for measuring frequency.

Features of PIC18F4550 [7] **:**

- 8 bit microcontroller
- 32 I/O pins (16 bidirectional)
- On board USB transceiver with on-chip voltage regulator
- Interface for off-chip USB transceiver
- 8 bit and 16 bit timers with software selectable pre-scalers

Figure 2.3 shows the pin diagram of the microcontroller for reference.

PIC18F4550 Pin Diagram:

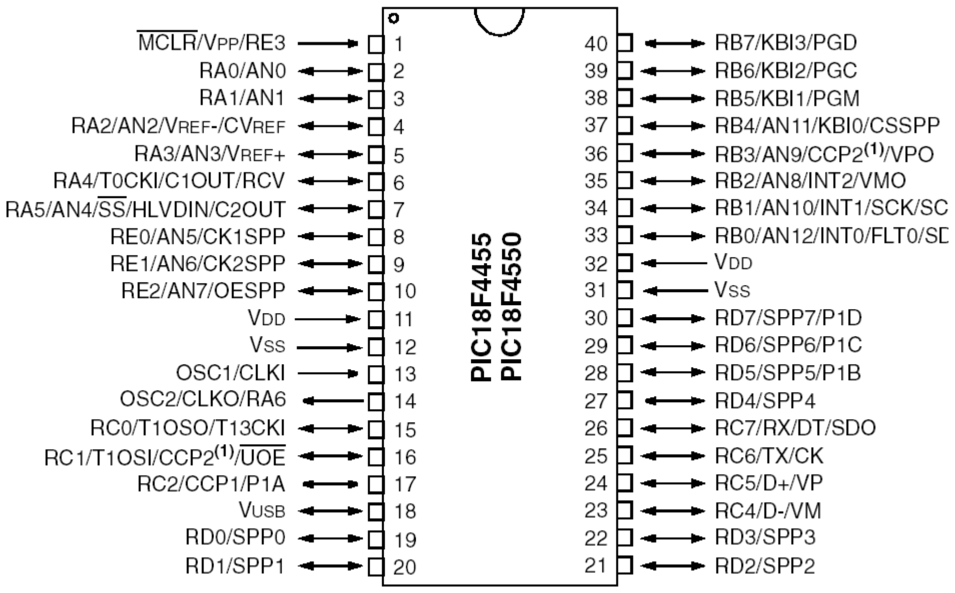


Figure 2.3: Pin diagram of PIC18F4550

USB Transceiver: The Universal Serial Bus (USB) is the most commonly used interface in personal computing today. With a high data transfer rate of up to 12 Mega bits per second, this interface can be found on many computer peripherals (e.g. monitors, keyboards, mice, speakers etc.) as well as on many consumer electronic devices.

The built in USB transceiver in the microcontroller handles the data transfer and handshaking procedures internally. Therefore, USB communication is possible without any external transceivers or hardware. Using the serial interface engine (SIE) of the microcontroller [7], it can communicate with a USB host (e.g. a computer) directly through the internal transceiver or using an optional external transceiver. It also has a 1 Kbyte USB RAM for uninterrupted data transfer.

USB Power Requirements: A USB device can be powered in two ways, i.e. from the USB bus (a bus powered device), or from external source (a self-powered device). The amount of current that can be drawn from the bus is limited to 100 mA to 500 mA. Generally, self-powered devices have larger current requirements (e.g. a USB hub). In our project, we configured the system to receive power from the bus.

Timer: There are several 8 bit and 16 bit timers built in to this microcontroller, namely TIMER0, TIMER1, TIMER2 and TIMER3. We used TIMER0 in our project to measure the sensor output frequency. The timer increments at every clock cycle with the program counter in the default mode. The increment rate can be further controlled with software. Using the microcontrollers interrupt, the timer can be made to count a clock frequency available at pin no. 33 as an interrupt. The details of the procedure to measure the frequency are described in the next chapter.

2.2.3 7 Segment display and display driver

In the project, we used four 7 segment displays to display the output frequency of the sensor directly on the hardware. Although 7 segment displays can be directly used with a microcontroller without the drivers, but they require too many I/O pins and multiplexing. So, in our project, the displays are connected to the microcontroller using BCD (binary coded decimal) to 7 segment display drivers. The 7 segment displays are available in two types, common anode and common cathode. The display drivers corresponding to these two types of displays are numbered 7447 and 7448. In our project, we used common anode type displays and the 7447 drivers. Figure 2.4 shows the pin connections of a common anode 7 segment display with a 7447 display driver.

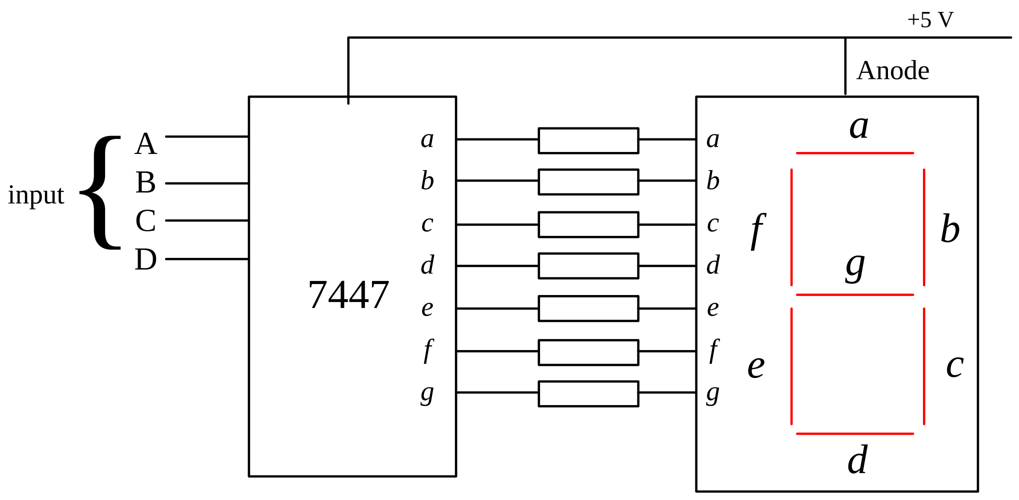


Figure 2.4: Seven segment display connected to a driver

Chapter 3: Software and PC Interfacing

3.1 Analyzing a Color using TCS230

As explained in Chapter 1, we have used white LEDs as light sources for the object for which the color has to be detected; because white light has all three lights (RGB) additively combined. When light is incident on an object, it will absorb some portion of light and reflect a portion of that light. The reflected light will contain the color spectrum containing the frequencies (wavelengths) corresponding to the object's color.

With the sensor receiving the reflected light, each different combination of filter selection pins will select a specific filter (i.e. red, green or blue) which allows red, green or blue light intensities of the incident light to be determined. For example, when S_2 and S_3 are both low (Table 2.2), then the red filter gets selected, and only red light will pass through the photodiodes and green and blue lights will be blocked. Similarly, when S_2 is low and S_3 is high, then only blue light will pass and the other colored lights will be blocked. Again, if S_2 is high and S_3 is high, then green light will pass and the other colored lights will be blocked. So, we can get the red, green and blue color frequencies from sensor output. These frequencies are proportional to color intensities.

3.1.1 Algorithm for Color Analyzing

The algorithm for color analyzing is shown in Figure 3.1. The sensor sequentially selects the red, green and blue filters. This selection is controlled by the microcontroller. The output frequency is then measured for each selected filter and then temporarily stored in the microcontroller's memory. After the frequencies for all the three colors are obtained, the data is sent to the computer via USB port. The color analysis ends with deactivating the sensor.

3.1.2 Algorithm for Frequency Measurement

Our system measures frequency of the sensor's output using the microcontroller's timer and interrupts. It uses the timer to count up to one second and in the meantime, the microcontroller counts the interrupts at the interrupt pin, where the sensor output is connected. For each positive going edge of the square wave (from the sensor output), the sensor's output pin gives a clock pulse as input to the interrupt pin of microcontroller. Since a positive going edge arrives once in every cycle, the microcontroller counts the cycles of the square wave. After 1 second, the timer is turned off and the total interrupt count is the measured frequency. The algorithm of the frequency measurement is shown in Figure 3.2.

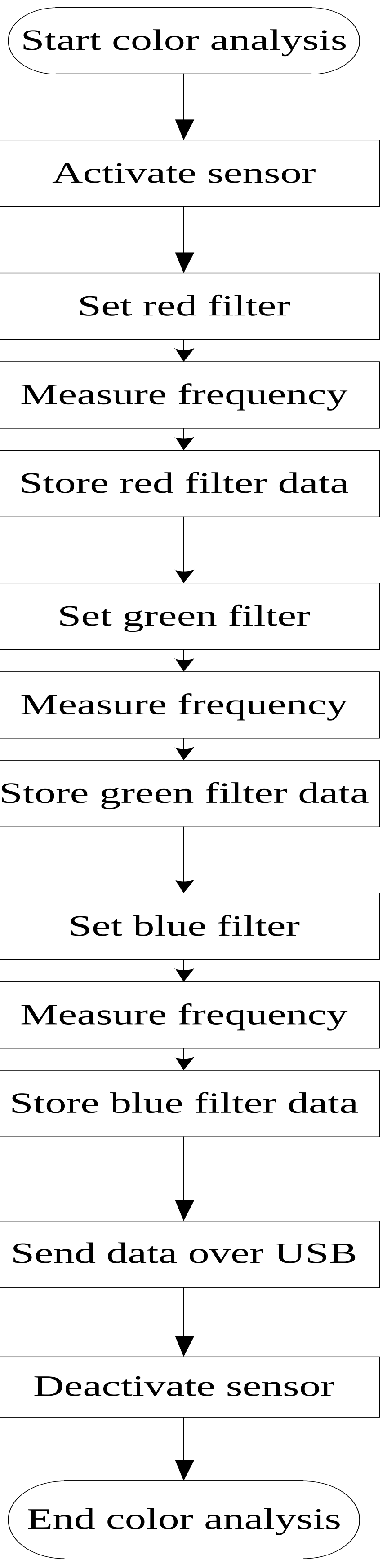


Figure 3.1 : Flow chart for analyzing color

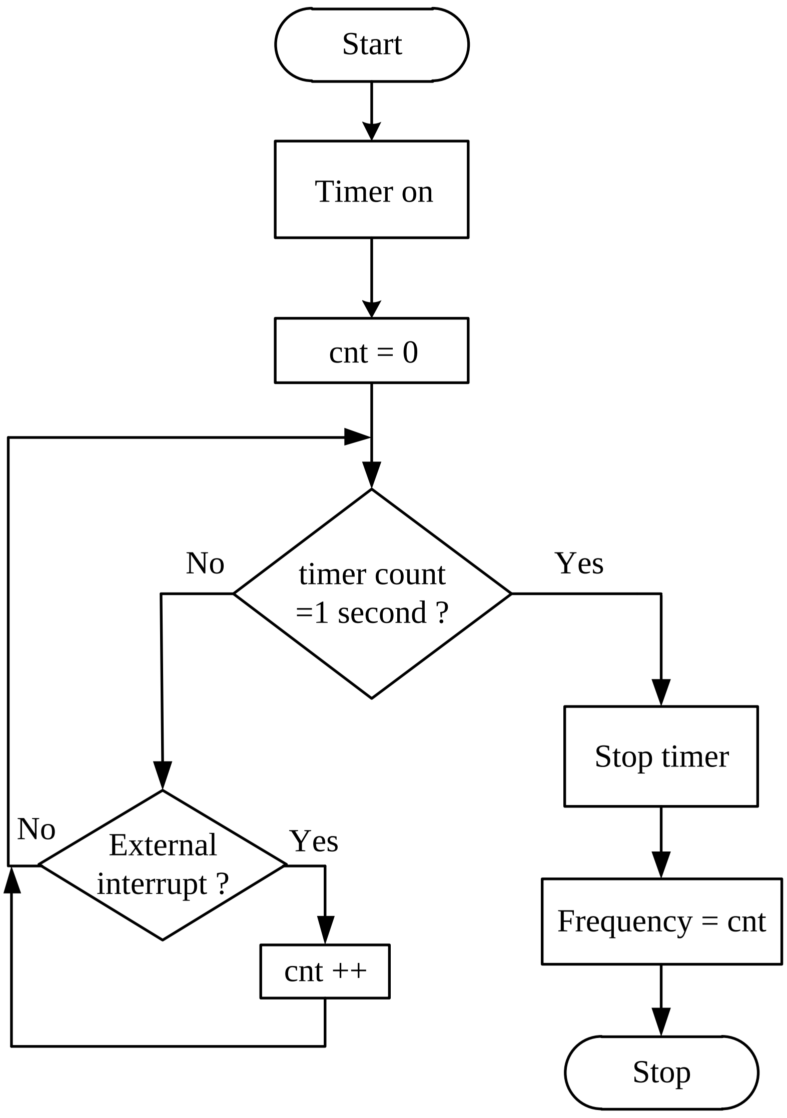


Figure 3.2 : Algorithm for frequency measurement

After measuring frequencies for each type of filters, the microcontroller has all the color components ready to be transmitted. The color data is then sent to the computer through Universal Serial Bus (USB).

3.2 PC Interfacing

Interfacing with a computer means establishing communication between the computer and the user or device. Most computer interfaces are bi-directional (both receive and send) but some are unit-directional (only receive or send) such as mouse or graphics adapter [8]. In order to display and store the data, we developed a graphical user interface and stand-alone software.

Universal Serial Bus (USB)

Universal Serial Bus (USB) is a connection technology which attaches other peripheral devices to a computer. It provides fast data exchange. USB enables the effortless connection of peripheral devices without the need to install special drivers on the computer.

We use USB because its data transfer rate is higher which is helpful to send the data quickly from our hardware to computer. Furthermore, now-a-days serial port (RS-232) and parallel ports are not available in portable laptop computers. So, by using USB in our project, the hardware can be used anywhere with access to a laptop computer.

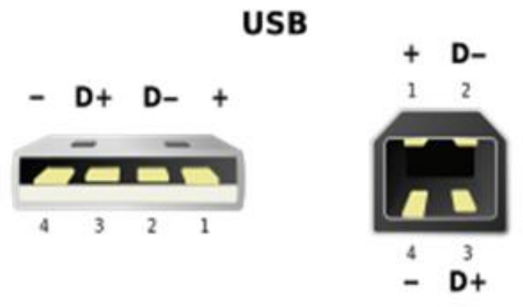


Figure 3.3: USB port [9]

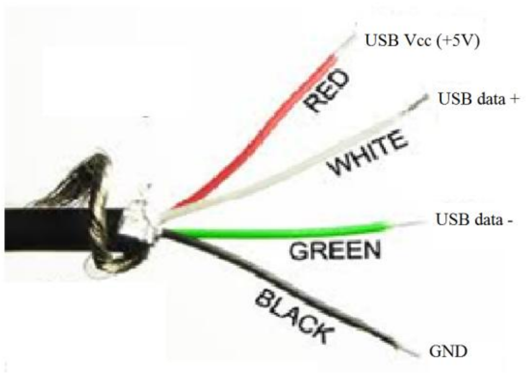


Figure 3.4: USB wiring [10]

Figure 3.3 and 3.4 show the USB port and the cable used for USB connections. There USB cable has four connections, VDD, GND, D+, and D-. VDD and GND are used for bus power (i.e. an external +5 V supply from which the USB devices may draw power). The specification provides for no more than 5.25 V and no less than 4.75 V (5 V±5%) between the positive and negative bus power lines [11], which are indicated by red and black. D+ and D- are the communications lines and are indicated by white and green wire.

Advantages:

1. USB ports can handle transmission of large quantities of data in a very short time.

2. USB-based device don’t always need separate power supply.

3.3 Graphical User Interface (GUI)

A GUI concentrates on the ease of using software rather than on the commands involved. Figure 3.5 shows the GUI built for our project. The GUI was built using MATLAB. It has three push buttons, i.e. ‘Start’, ‘End’ and ‘Sense’. The start button establishes communication with the microcontroller via the USB port. Then, pushing the ‘Sense’ button will activate the sensor, obtain the frequencies for red, green and blue color components, and

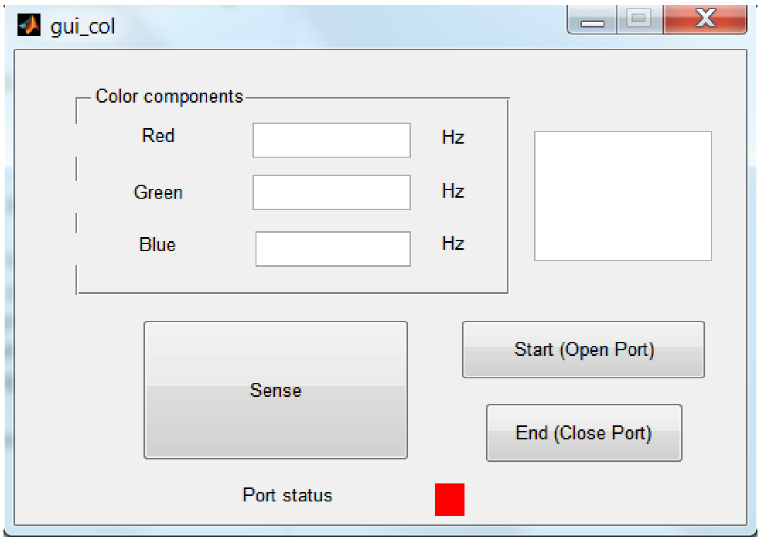


Figure 3.5: GUI for the color analyzer

display the frequency values on screen. It will also re-generate the color using the measured frequency values and save the data on the computer's hard drive. Pressing the 'End' button stops the communication between the computer and the microcontroller. A colored box indicates the port's status. It shows red if the port is closed and green when it is open.

The MATLAB code for the GUI is provided in Appendix B.

Chapter 4 : Hardware Construction

4.1 Introduction

The components are connected as shown in Figure 4.1. In addition to the sensor, the hardware consists of the microcontroller, four 7 segment displays and drivers.

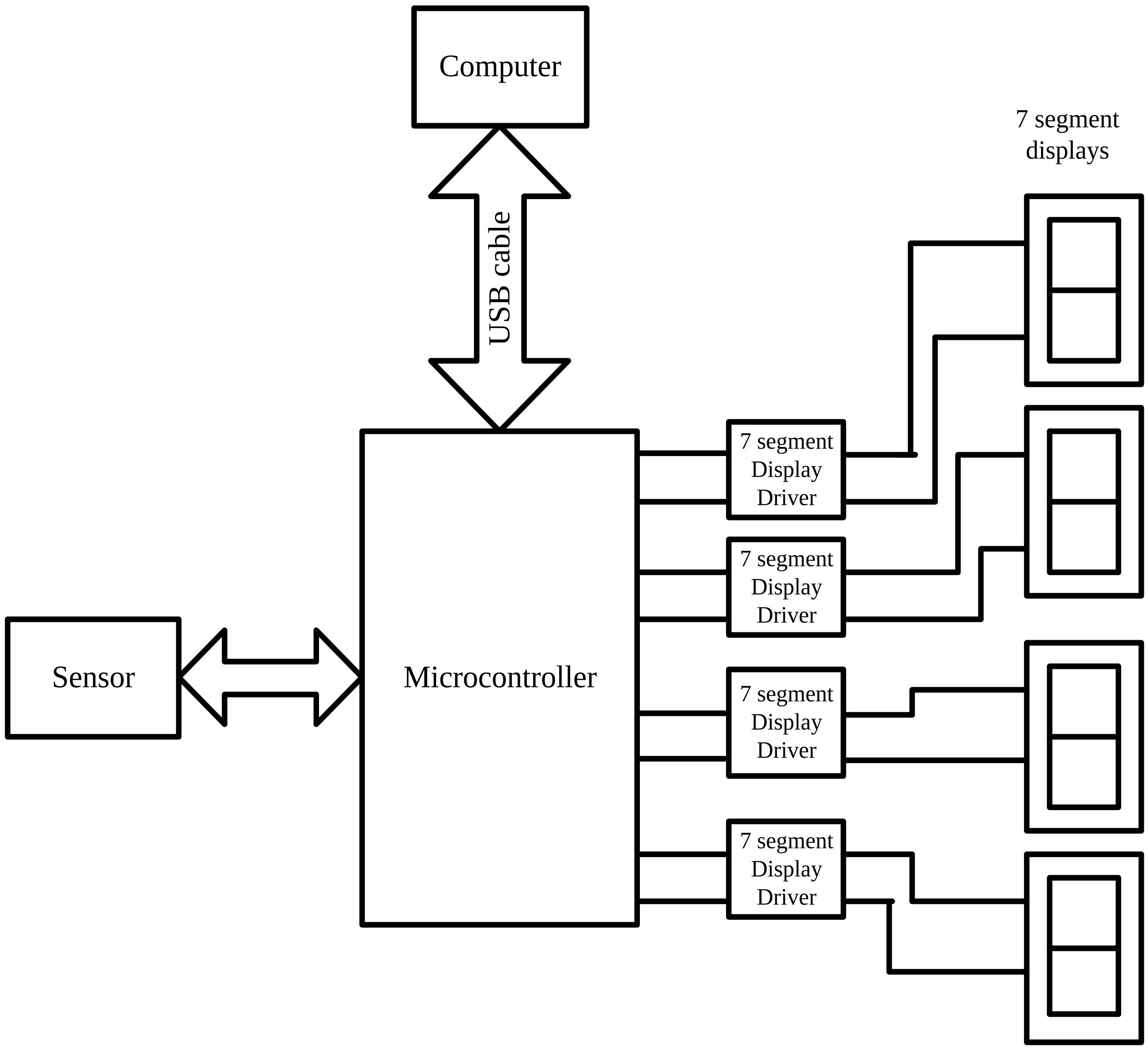


Figure 4.1 : Block diagram of the hardware arrangement

We first tested the system on a breadboard, but since the sensor is surface mounted type, it had to be attached to a printed circuit board (PCB). After we finished the initial tests, we constructed the whole circuit on a PCB.

4.2 List of Components

- **Sensor Section**
  - Color sensor (TCS230).
  - White LED array.
- **Microcontroller Section**
  - A Microcontroller (PIC18F4550).
  - 7 segment display driver.
  - 7 segment display.
  - 20 MHz crystal oscillator.
- **Others**
  - USB cable.
  - Computer.

4.2.1 Sensor Section

- **LED Array with Sensor:**

We use special types of white LED in our project whose intensities are higher. The LED's are surface mounted types. Figure 4.2 shows the white LED used in our project.

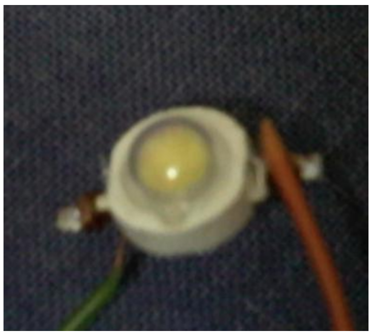


Figure 4.2 : Light emitting diode (LED)

We arranged 6 LEDs around the sensor in a circular shape, because, for a circular arrangement, we will get better focusing at the center. The sensor was placed at the center of the circle. Figure 4.2 shows the white LED used in our project.

We tested the arrangement of LEDs around the sensor for different radii, and found out that, if we make the radius of the circular array to nearly 1.5 cm and keep the angle between two adjacent LED's almost 30 degrees, then, we can obtain best data. In Figure 4.3, a schematic of LED arrangement around the sensor has been shown, and Figure 4.4 shows the actual arrangement. The sensor has eight pins which are connected to the microcontroller via a flat straight wire. The LED's are connected in series around the sensor.

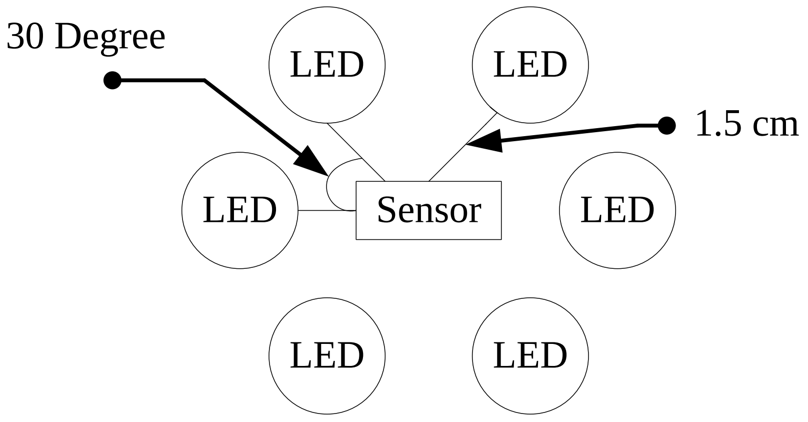


Figure 4.3 : Layout of LED's around the color sensor

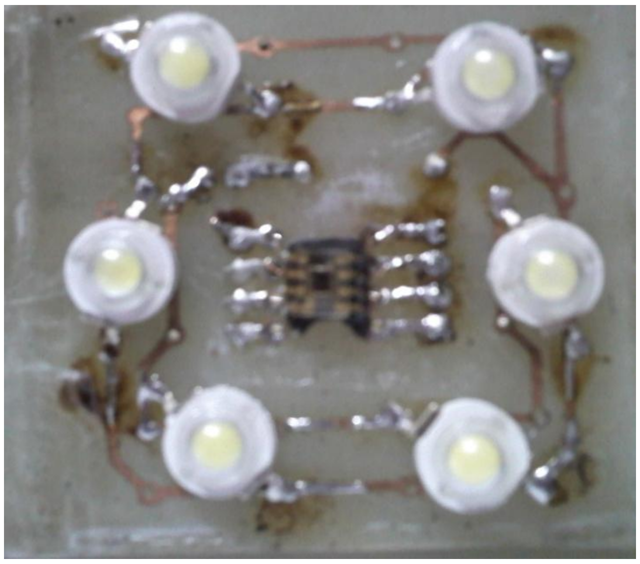


Figure 4.4 : LED array around Sensor

To reduce the interference due to ambient light, we used a black cover with slanted sides around the sensor. With the cover, the height from the object to the sensor is approximately 1.9 cm. Figure 4.5 shows our design. Light emitted from the LEDs are reflected from the surface, and only the directly reflected light will be incident on the sensor.

The sensor has eight pins which are connected to the microcontroller via a flat straight wire. The LED's are connected in series around the sensor.

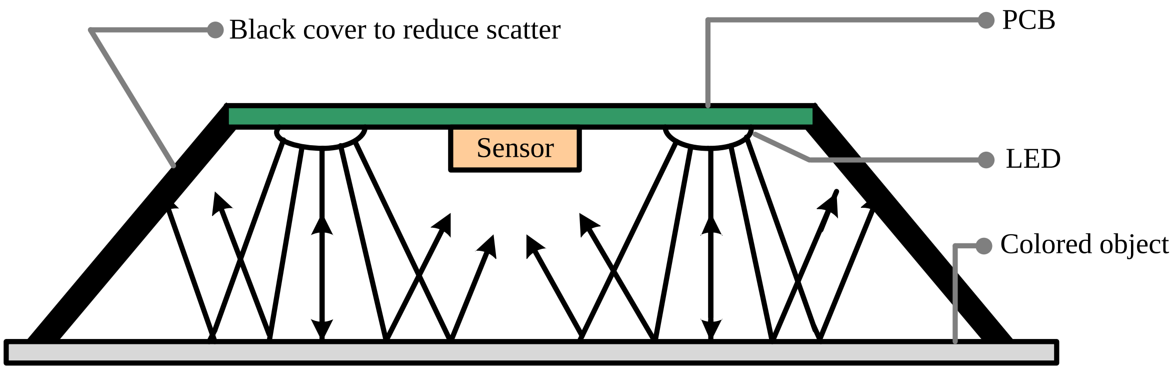


Figure 4.5 : Light reflection from a colored surface

4.2.2 Microcontroller Section

In this section we connected the microcontroller to the display drivers, 7 segment displays and complete the USB connection. We also added some important components like crystal oscillator and finally connect the sensor hardware.

We used the crystal oscillator because microcontroller's internal oscillator is not enough for USB data transmission. To speed up USB data transmission, we used an external 20 MHz crystal oscillator.

To display the frequencies without the computer, we used four common anode 7 segment displays, and four 7447 display drivers. Figure 4.6 and 4.7 show the hardware and the display respectively.

Figure 4.8 shows an LED used to indicate whether the USB cable is properly connected to the computer or not. Finally, Figure 4.9 and 4.10 show the completed hardware for the color analyzer.

The complete schematic diagram of the circuit is provided in Appendix A.

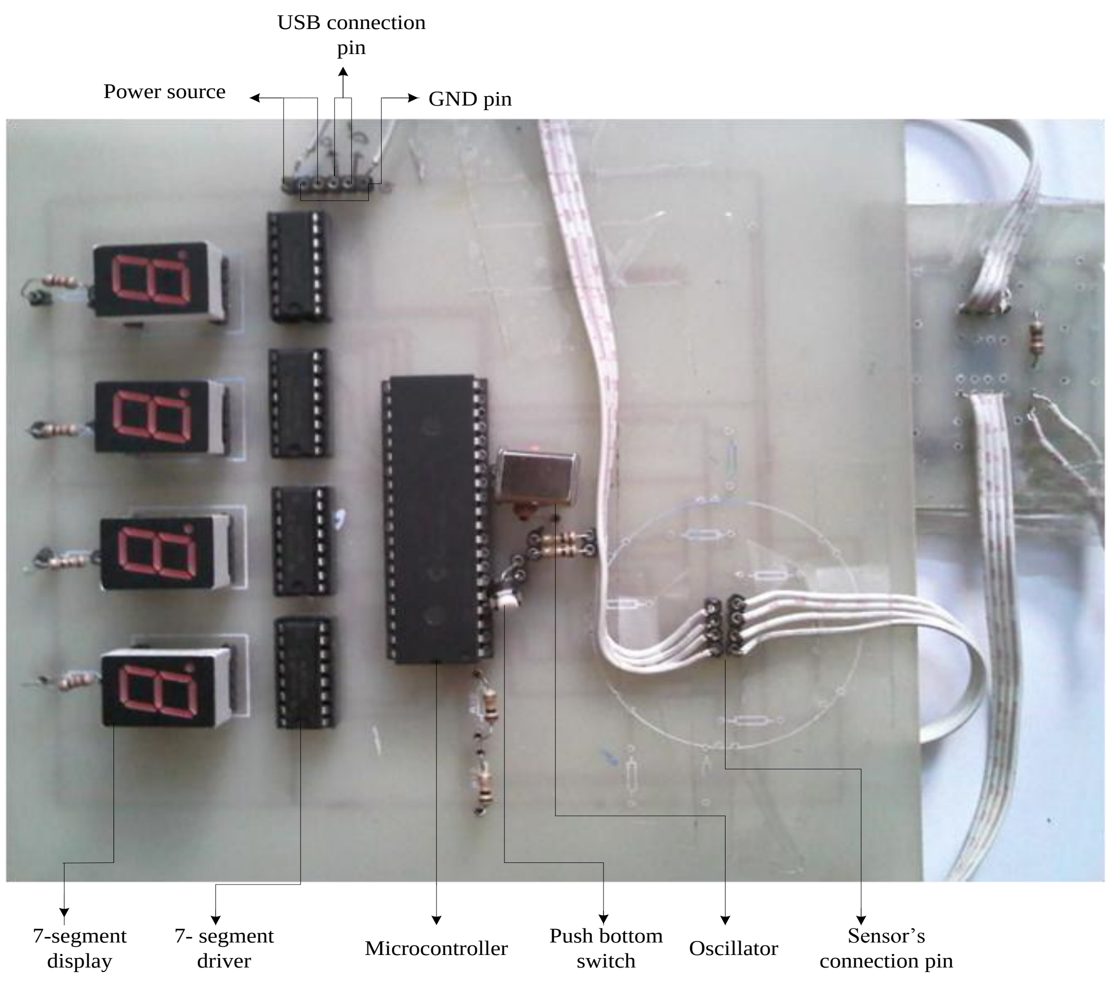


Figure 4.6: Hardware for color analyzer

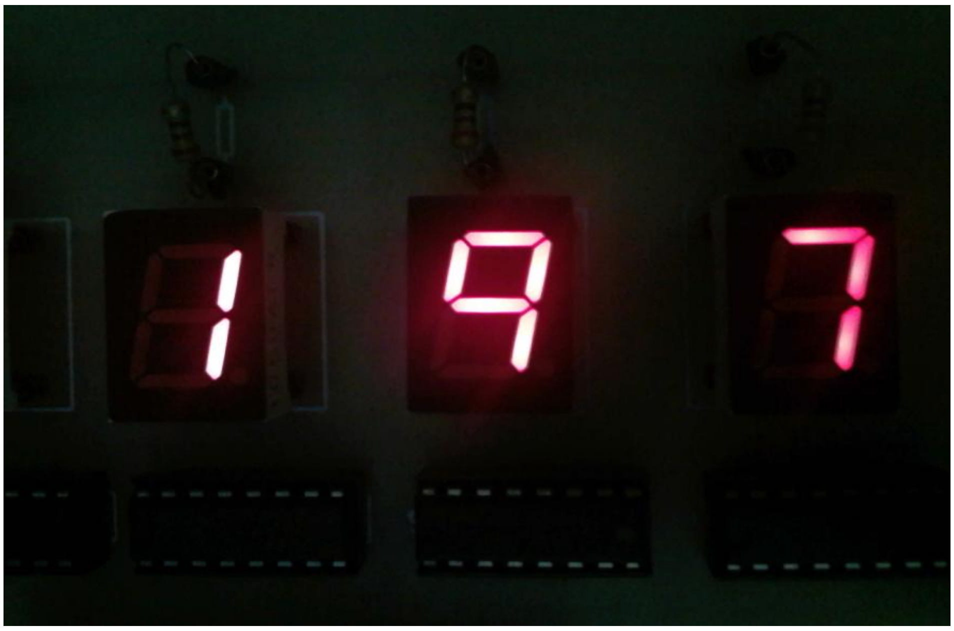


Figure 4.7: Display frequency reading at seven segment display

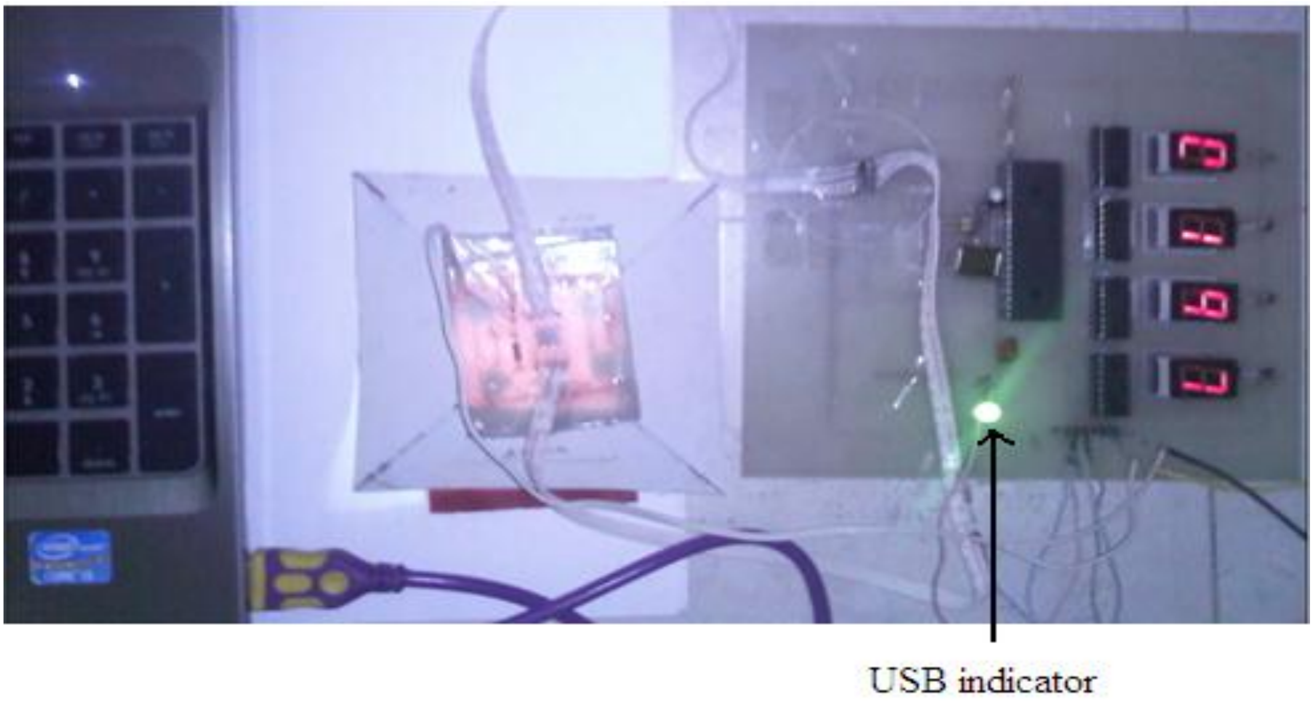


Figure 4.8: Color analyzer powered up

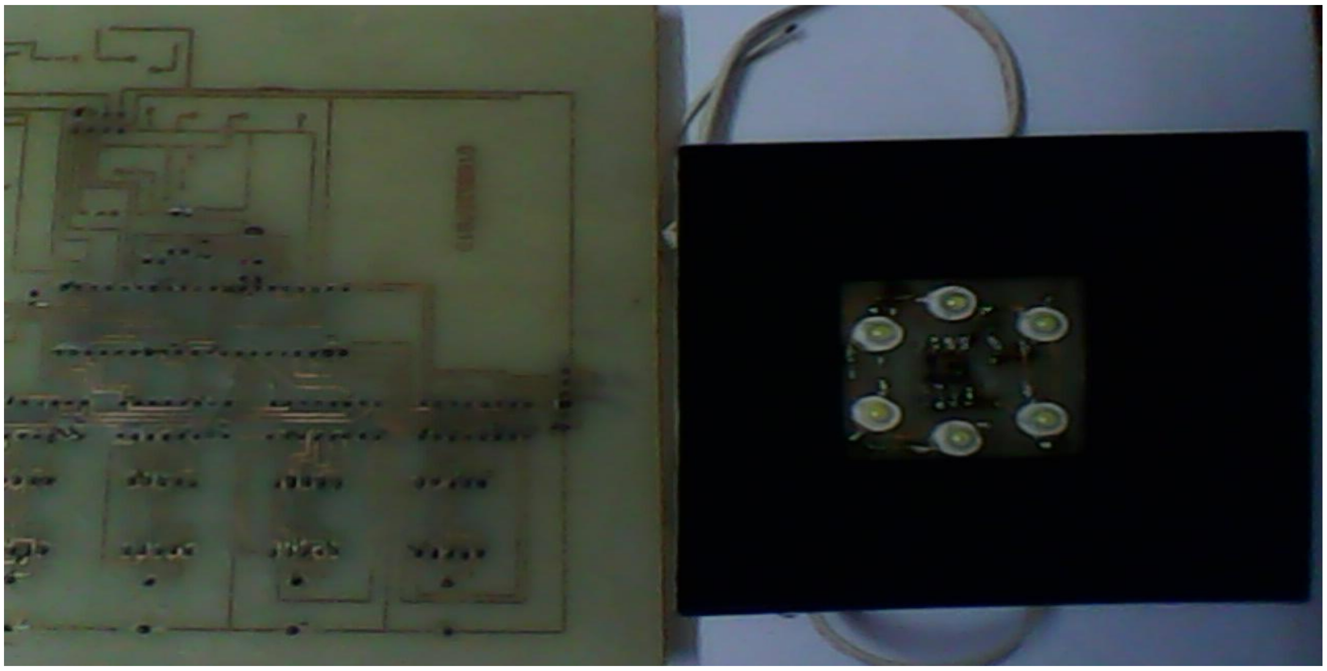


Figure 4.9: Bottom view of hardware

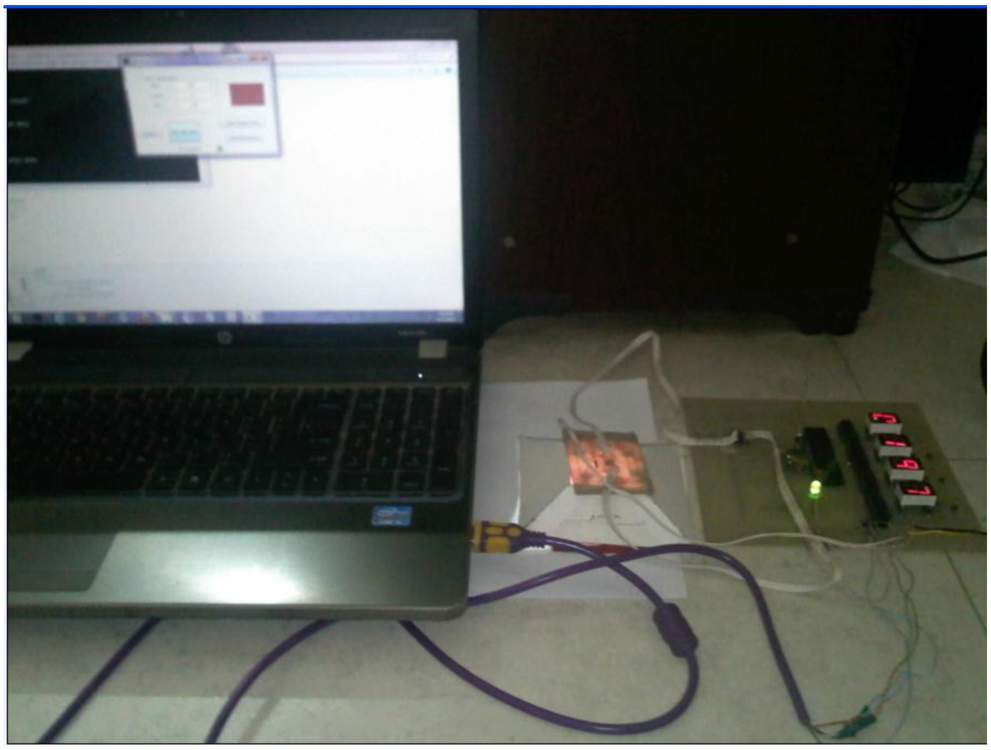


Figure 4.10: Complete hardware connection

Connection with PC:

After connecting the USB cable to the computer, we run the software. The USB device gets power from the computer's USB port, but the microcontroller, the sensor and the white LEDs require an external 5 V power supply.

Chapter 5 : Experiment and Data Collection

We have tested the color analyzer on different colored surfaces. Keeping in mind the target industrial use, we tested on plain papers and cloth. Table 4.1 and 4.2 summarize the data obtained through our experiment. The tables show the red, green and blue frequency which is proportional to their corresponding irradiance for the tested colors. The table also shows the regenerated color which is obtained by the combination of red, green and blue frequencies.

Table 5.1 : Color analysis on different colored paper

Tested colors (on paper)	Name	Frequencies of color components			Regenerated color
		Red (Hz)	Green (Hz)	Blue (Hz)	
🟥	Red	196	51	60	🟥
🟩	Green	135	164	113	🟩
🟦	Blue	114	178	251	🟦
🟨	Yellow	421	280	160	🟨
	White	509	416	439	
⬛	Black	35	26	25	⬛
🟧	Orange	400	137	125	🟥
🟪	Violet	29	20	21	⬛

Table 5.2 : Color analysis on different colored cloths

Tested colors (on cloth)	Color name	Frequencies of color components			Regenerated color
		Red (Hz)	Green (Hz)	Blue (Hz)	
🟥	Red	270	49	64	🟥
🟩	Green	49	102	75	🟩
🟦	Blue	47	40	64	⬛
🟫	Yellow	357	175	98	🟧
	White	509	416	439	
🟪	Black	37	22	21	⬛
🟪	Pink	367	94	171	🟥

The RGB response of the sensor and the RGB response of computer monitor are not the same. Since the RGB responses are calibrated differently, so, the regenerated color doesn’t match the original color exactly. If the RGB response of the sensor could match with the RGB response of the monitor, then the error can be removed.

Chapter 6 : Conclusion

A color analyzer plays an important role in textile and dyeing industries. We made a microcontroller based color analyzer with computer interfacing. The interfacing has created an opportunity to store huge amount of data, because, we cannot store too much data into the microcontroller due to its limited memory.

Though we met the necessary demand of making our project, but we have some limitations. If we can further improve some aspects, then our project may have a professional look.

There are some problems and errors in our project. First of all, the colors of the tested materials and the obtained color by the color analyzer are not exactly same. This is due to the difference in our sensor's RGB response and the RGB response of the computer monitor. If it is possible to match the sensor and the monitor's RGB response, then, the obtained color by color analyzer would match more accurately with the tested color. Another source of error might be the white LEDs that were used as light source. The LEDs are not exactly white, but have a yellowish tinge. Furthermore, if we could get standard color charts, then the color analyzer could be calibrated against those colors and the errors can be removed.

Another problem is the high price of the sensor. Although we used a color sensor in our project, the color analyzer could also be made using photodiode arrays at a relatively low cost. But, the photodiode arrays in a geometric arrangement would result in poor reception of the reflected light. The color sensor has a smaller footprint which increases the scope for accurate focusing and better reception of the reflected light. The fact that the sensor is more accurate than photodiode arrays could be ensured through experiments.

List of References

[1] *"Spectral colors",* http://en.wikipedia.org/wiki/Color

[2] "*Additive primaries*", http://en.wikipedia.org/wiki/Primary_color

[3] Tom Duncan, "Reflection at plane surfaces", *Advanced physics, materials and Mechanics*, John Murray, 1973, 2nd ed., p.132

[4] "*TCS 230 Data sheet",* http://www.unihedron.com/projects/darksky/tcs230-e33.pdf

[5] "*Arduino TCS230 Color Recognition Sensor module",* http://www.geeetech.com/wiki/index.php/Arduino_TCS230_Color_Recognition_Sensor_module

[6] "*Photodiode*", http://en.wikipedia.org/wiki/Photodiode

[7] *"PIC18F4550 Data sheet*", http://ww1.microchip.com/downloads/en/devicedoc/39632c.pdf

[8] "*Interface (computing)",* http://en.wikipedia.org/wiki/Interface_(computing)

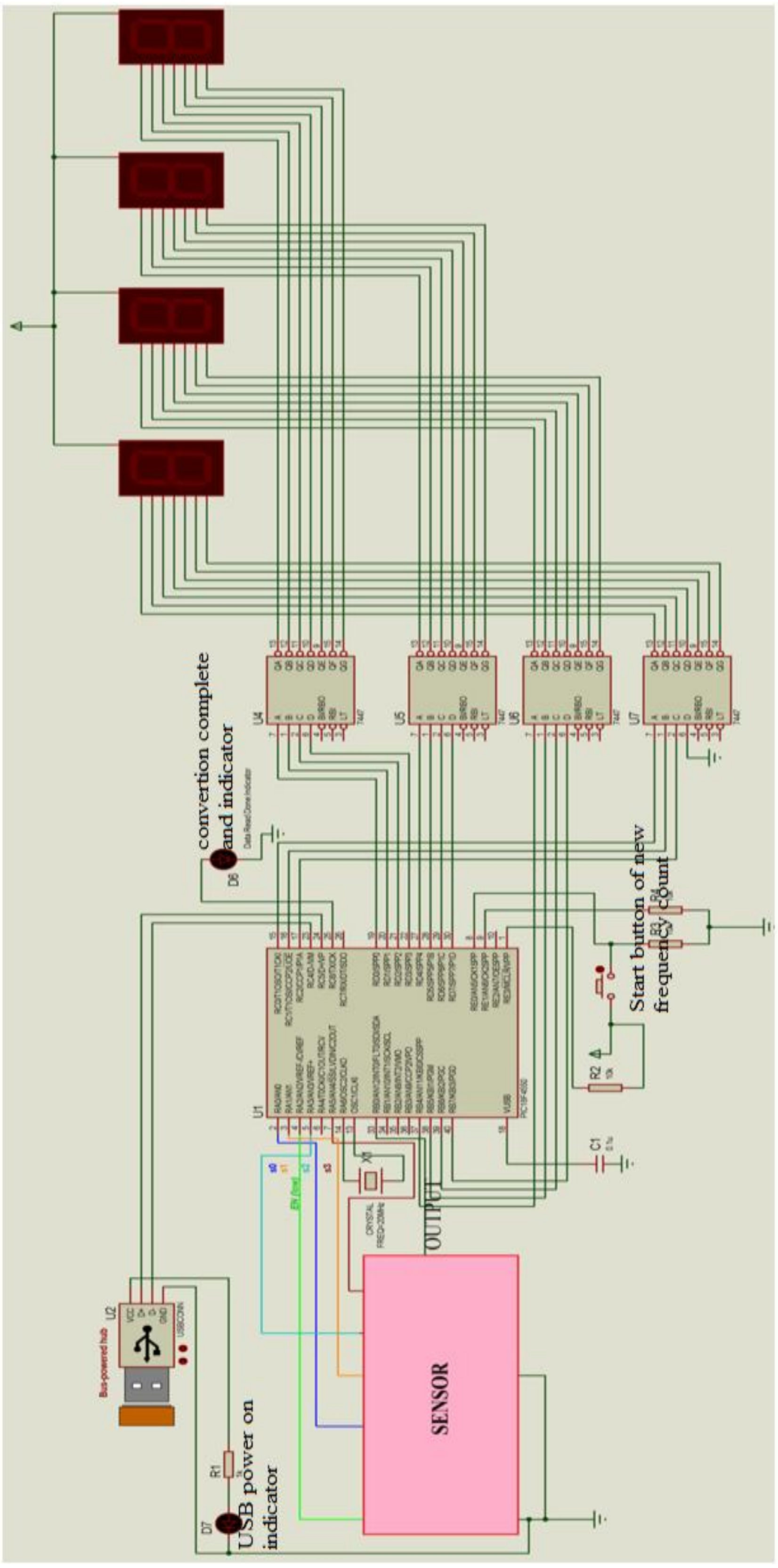
[9] "*USB standard connectors"* http://en.wikipedia.org/wiki/File:USB.svg

[10] *"USB cable wiring",* http://forum.xda-developers.com/showthread.php?t=1828032

[11] *"Power",* http://en.wikipedia.org/wiki/Universal_Serial_Bus#Power

Appendix A

Circuit diagram



Appendix B

GUI Code:

```
function varargout = gui_col(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @gui_col_OpeningFcn, ...
                   'gui_OutputFcn',  @gui_col_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
function gui_col_OpeningFcn(hObject, eventdata, handles, varargin)
handles.output = hObject;

guidata(hObject, handles);
function varargout = gui_col_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;

function red_Callback(hObject, eventdata, handles)
function red_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function green_Callback(hObject, eventdata, handles)
function green_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function blue_Callback(hObject, eventdata, handles)
function blue_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function port_open_Callback(hObject, eventdata, handles)
s = serial('COM3');
set(s,'BaudRate',4800);
fopen(s);
function port_close_Callback(hObject, eventdata, handles)
s = serial('COM3');
fclose(s)
delete(s)
function run_pushbutton_Callback(hObject, eventdata, handles)
clc
s = serial('COM3');
set(s,'BaudRate',4800);
fopen(s);
out = fscanf(s);
fclose(s)
delete(s)
test_data= out;
valid_data=zeros(1,12);
```

```
red_data=zeros(1,4);
green_data=zeros(1,4);
blue_data=zeros(1,4);
buff=zeros(1,4);
loop_end_value=length(test_data);
for i=1:1:loop_end_value
    if test_data(i)=='$' && test_data(i+13)=='#'
        for x=1:1:12
        valid_data(x)=test_data(i+x);
            if x==4
                for p=1:4
                    buff(p)=valid_data(p);
                end
            red_data=char(buff); %transfer data form array to string
            elseif x==8
                for p=1:4
                    buff(p)=valid_data(p+4);
                end
            green_data=char(buff); %transfer data form array to string
            elseif x==12
                for p=1:4
                    buff(p)=valid_data(p+8);
                end
            blue_data=char(buff); %transfer data form array to string
            end
        end
    end
end
re=str2num(red_data)/1000;
gr=str2num(green_data)/1000;
bl=str2num(blue_data)/1000;
set(gca,'Color',[re,gr,bl]) %Set the color to the object according to the
code
set(handles.red,'string',red_data);
set(handles.green,'string',green_data);
set(handles.blue,'string',blue_data);
function exit_Callback(hObject, eventdata, handles)
close
function color_test_Callback(hObject, eventdata, handles)
r=get(handles.red,'string');
g=get(handles.green,'string');
b=get(handles.blue,'string');
re=str2num(r)/1000;
gr=str2num(g)/1000;
bl=str2num(b)/1000;
set(gca,'Color',[re,gr,bl])
```

Appendix C

Microcontroller code:

```
#include<p18f4550.h>
#include<delays.h>
#include <timers.h>
#define _XTAL_FREQ 20000000
#pragma config FOSC = HS
#pragma config PWRT = OFF
#pragma config BOR = OFF
#pragma config MCLRE = ON
#pragma config PBADEN = OFF
#pragma config ICPRT = OFF
#pragma config LVP = OFF
#pragma config WDT = OFF,DEBUG=OFF
int nofilter=0;
int red=0;
int green=0;
int blue=0;
int cnt=0;                 //for frequency
int overflow=0;                    //for 1 second count

void display(int x){
       int unit, unit1, unit2;
       unit=x%10;
       x=x/10;
       unit1=x%10;
       x=x/10;
       unit2=x%10;
       x=x/10;
       PORTD=(unit1<<4)+unit;
       PORTB =(unit2<<4);//shifting for use lowest 4 bit
       PORTC =x;
}

//Freq mesure function
int freq_mes(void){
       overflow=0;
       TMR0L=0x00;
       cnt=0;
       INTCONbits.TMR0IE = 1;      // Enable TMR0 overflow interrupt.
       INTCONbits.INT0IE=1;
       while(overflow<77);               //20Mhz/4/256/256=76.3
       INTCONbits.INT0IE=0;              //interrupt off
       INTCONbits.TMR0IE =0;       // Disable TMR0 overflow interrupt.
       return cnt;
}

void init(void){
       TRISA = 0;          // Set PORTA to outputs
       PORTA = 0;          // Clear all outputs
       TRISB = 0x01; // Set PORTB to outputs without RB0
       PORTB = 0;    // Clear all outputs
       TRISD=0;
       PORTD=0;
       TRISC=0;
       PORTC=0;
       RCONbits.IPEN = 1;
       TRISEbits.TRISE0 = 1;
       TRISEbits.TRISE1 = 1;
       ADCON0=0;
       ADCON1=0x0F;                // Set all pins as digital I/O
       CMCON=0x07;                 // Set all comparators as digital I/O
       LATAbits.LATA0 = 0;               //for 20%output
       LATAbits.LATA1 = 1;
```

```
        LATAbits.LATA2 = 1;               //output off
        INTCON2bits.INTEDG0=1;            //select rising edge interrupt
        INTCONbits.INT0IE=0;              //innitially interrupt 0
        INTCONbits.INT0IF = 0;
        OpenTimer0( TIMER_INT_ON & //timer0 select as 8bit and 256 prescale
                T0_8BIT &
                T0_SOURCE_INT &
                T0_PS_1_256 );
        INTCON2 = 0x84;                //TMR0 high priority  T1CON = 0;

        INTCON2bits.TMR0IP = 0;           // timer0 interrupt prirority setup
        INTCONbits.GIEL = 1;              // global interrupt enabled
        INTCONbits.GIEH = 1;              // global interrupt enabled
        INTCONbits.GIE =1;                // Enable interrupts
        INTCONbits.TMR0IF = 0;         // Clear the TMR0 overflow interrupt
flag.
        INTCONbits.TMR0IE = 0;         // initially stop TMR0 overflow
interrupt.
        Delay10KTCYx(500);
}

void sens(void){
        LATAbits.LATA3 = 1;
        LATAbits.LATA5 = 0; // select no filter condition
        LATAbits.LATA2 = 0; //Sensor output enable
        Delay10KTCYx(50);
        nofilter=freq_mes();              //output without filtering
        display(nofilter);                //display no filter freq

        LATAbits.LATA3 = 0;
        LATAbits.LATA5 = 0; // select red filter condition
        Delay10KTCYx(50);
        red=freq_mes();     //read output with red filtering
        display(red);       //display red freq

        LATAbits.LATA3 = 0;
        LATAbits.LATA5 = 1; // select blue filter condition
        Delay10KTCYx(50);
        blue=freq_mes();    //read output with blue filtering
        display(blue);              //display blue freq

        LATAbits.LATA3 = 1;
        LATAbits.LATA5 = 1; // select green filter condition
        Delay10KTCYx(50);
        green=freq_mes();   //read output with green filtering
        display(green);     //display green freq
        LATAbits.LATA2 = 1;

        //nofilter, red,blue,green veriable will be sent on PC by USB
}
//Interrupt function

void InterruptServiceHigh(void);
#pragma code InterruptVectorHigh = 0x08

void InterruptVectorHigh(void){
        _asm
        goto InterruptServiceHigh
                _endasm
}

#pragma code
#pragma interrupt InterruptServiceHigh

void InterruptServiceHigh(){
        if(INTCONbits.INT0IF)
        {
                cnt++;
                INTCONbits.INT0IF = 0;
        }
```

```
}
void timer_isr (void);

#pragma code low_vector=0x18

void low_interrupt (void){
       _asm GOTO timer_isr _endasm
}

#pragma code
#pragma interruptlow timer_isr

void timer_isr (void){
       INTCONbits.TMR0IF=0;
       overflow++;
}

//Main functionvoid
void main(void){
       init();
       for(;;) // do forever
       {
              if(PORTEbits.RE0){
                     sens();
              }
              if(PORTEbits.RE1){
                     //usb function
              }
       }
}
```