DEPARTMENT OF PHYSICS, UNIVERSITY OF COLOMBO

PH3032-EMBEDDED SYSTEM LABORATORY

2021

FINAL PROJECT REPORT

8×8×8 LED cube with Interrupts

S14309 - BHATHIYA BANDARA

AUGUST 12, 2021

Table of Contents

Table of Figures

1. Introduction

A microcontroller is a small computer. Both have several things common like the CPU (central processing unit), execution of programs, presence of RAM (random-access memory) etc. Then we can assume microcontroller is “special purpose computer”. It is often low- power device and it is often small and low cost. Today the technology has advanced to such an extent that has come a need to display electronic messages to satisfy all purposes, whether it is business or domestic use. The solution found to satisfy this need is the matrix display systems using LEDs or LCDs. It provides instantaneous and flexible communications.

In this project, 8x8x8 LED cube was selected to build, which has 512 LEDs. LED cube is like a LED screen, but it is special in that it has a third dimension, making it 3D. In normal displays, it is normal to try to stack the pixels as close as possible in order to make it look better, but in a cube, one must be able to see through it. A LED cube does not have to be symmetrical all the time. It is possible to make an 8x9x10, or even oddly shaped ones. Here I have an 8x8x8 shaped one. The code is written in the C language using AVR studio and it is burned into the ATmega32 microcontroller using USBasp cable and ATmega32 development board.

This LED cube has 512 LEDs. Obviously, having a dedicated IO port for each LED would be very impractical. Therefore, I have to use 74HC595 shift registers to solve that issue. In here, actually LED cube relies on an optical phenomenon called persistence of vision (POV). If we flash a led fast, the image will stay on our retina for a little while after the led turns off. By flashing each layer of the cube one after another fast, it gives the illusion of a 3D image and looking at a series of 2D images stacked onto one another called multiplexing. Those two phenomenon were used in this Project.

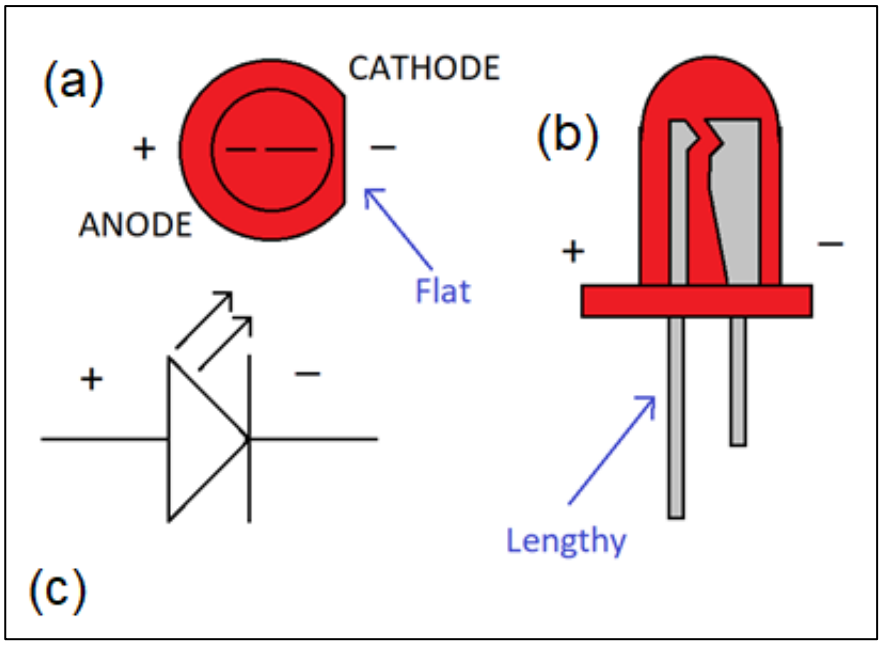
Basically, the project is a 3D LED cube display (8x8x8 pixel) which displays different patterns. Patterns can change using push buttons. All circuits were soldered on zero dot boars permanently. All the displaying patterns were stored in the microcontroller. All the patterns were controlled using nine shift registers. The patterns are displayed on a 3D structure, which is made up of steel rods. The complete display system circuit is power supply run on 5V, 4A that is provided externally. This unique way of displaying messages is a very eye catching. Therefore it can uses in the field of advertising, toys, decorating etc.

2. Theory

2.1 Components

2.1.1 LED

A light-emitting diode (LED) is a semiconductor light source that emits light when current flows through it. The color of the light is determined by the photon energy determines the wavelength of the emitted light, and hence its color. It depend on the energy required for electrons to cross the band gap of the semiconductor. In this project red color, LEDs were used. The main semiconductor materials used to manufacture Red LEDs are Aluminum gallium indium phosphide (AlGaInP) or Aluminum gallium arsenide (AlGaAs).

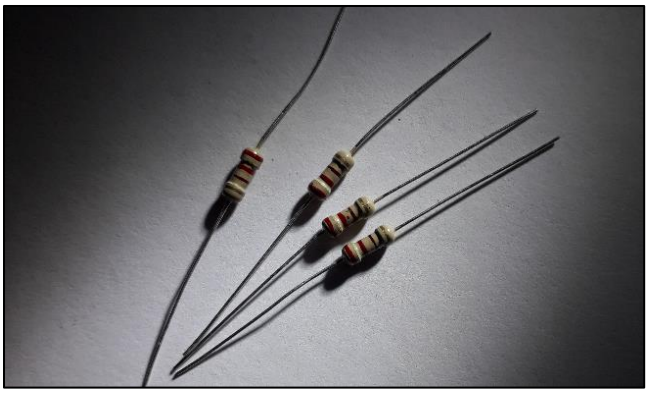


2.1.1-1 How to recognize anode and cathode of an LED

It is very important to recognize anode and cathode of the LED correctly. As seen in above figure an LED bulb has cathode and anode. Part (a) shows how the LED appear from above. We can easily recognize the flat edge. That flat edge represents The cathode side of the LED. Part (b) is the side appearance of the LED. When we considering the pins of the LED its easy to find the long one. That Lengthy pin is the cathode of the LED. Part (c) Shows the circuit diagram of the LED. In here, we can recognize the both anode and cathode as shown in the figure itself.

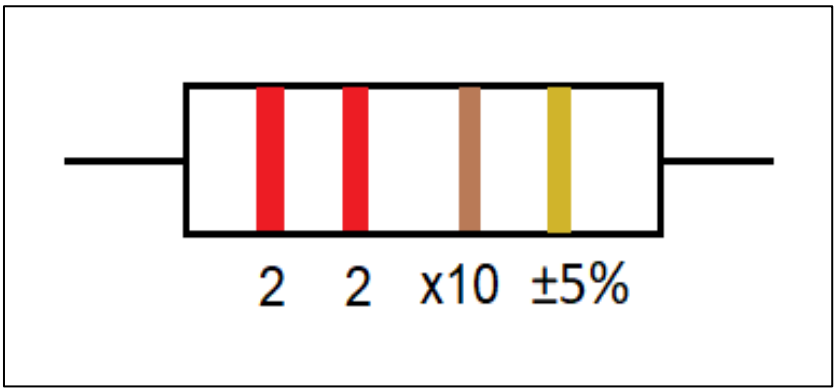
2.1.2 Resistor

Resistors are used to resist and reduce the flow of current through the electrical circuit. They can also be used to provide specific voltages to active components. Resistors are available with fixed or variable resistance values. Fixed resistor's resistance never changes. However, with variable resistors (sometimes called potentiometers), allow adjust the resistance. Resistor values are measured in Ohms. Commonly, resistors are constructed of metal or carbon and are available in a variety of mounting styles.



2.1.2-1 Appearance of the220 Ω resistor

Above figure shows couple of 220 Ω resisters those were used in the project. To recognize the resistance of a resistor there is a color code system.



2.1.2-2 Color code of the 220 Ω resistor

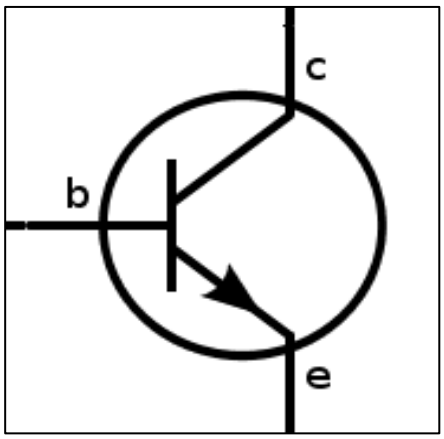
As seen in above figure, there is four color lines on the resistance surface to identify the resistance. The left side first code shows the first digit.It is red so its corresponding value is two. Second one shows second digit isa also red and. Its value also two. Now we have 22 in our hand. Then the third code. It is brown and It tells us the value must multiply by hundred. Then the value is 220. Last column for tolerance. In here the code is gold and it says the tolerance is five percent. There is a table to identify the resistance easily by checking the color codes.

Table 2.1.2-1 The table of color codes of resistors

	1st Digit	2nd Digit	Multiplier	Tolerance
Black	0	0	×1	5% Gold
Brown	1	1	×10	10% Silver
Red	2	2	×100	
Orange	3	3	×1000	
Yellow	4	4	×10000	
Green	5	5	×100000	
Blue	6	6	×1000000	
Purple	7	7		
Gray	8	8		
White	9	9		

2.1.3 Transistors

A transistor is a device that regulates current or voltage flow and acts as a switch or gate for electronic signals. Transistors consist of three layers of a semiconductor material, each capable of carrying a current. Transistors are basically, small electronic components with two main functions switch circuits and amplify signals. In this project Transistors were used to switching purpose.



2.1.3-1 Circuit diagram of the transistor

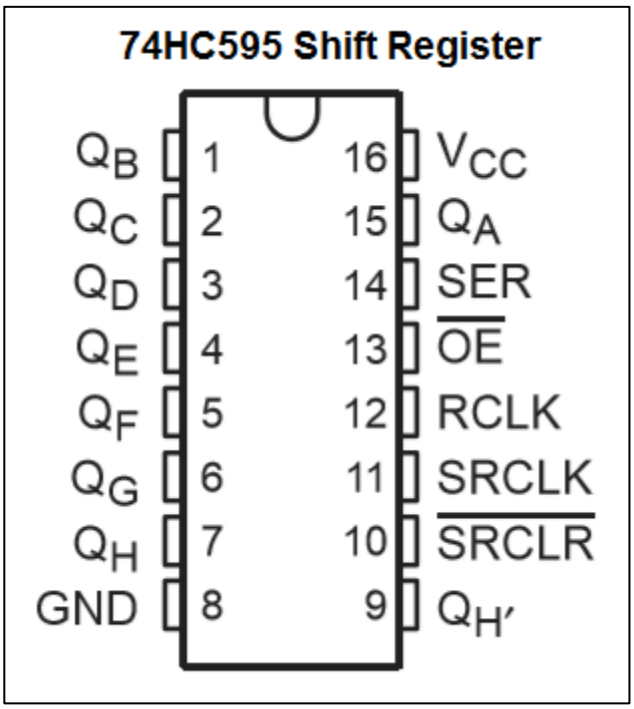
Above figure shows transistor symbol of a circuit diagram. It has three legs called Base (b), Collector (c) and Emitter (e). If you turn it ON, current can flow through it from the collector to the emitter. When it is OFF, no current can flow. If BE junction is Forward Bias and BC is Reverse Bias we called the transistor in in the active mode. There is two types of transistors. Bipolar Junction Transistors (BTJ). In addition, Transistor can divided in two types as NPN type and PNP type. In this project, CTC 1061 transistors were used.

2.1.3-2 CTC1061 Transistor and its legs configuration

Above figure shows CTC 1061 Transistor and its legs. It is NPN type Transistor. It can also call 2SC1061. There are many Replacement and Equivalent components for this transistor. We can replace this with SC1826, 2SC1985, 2SC1986, 2SC2075, 2SC3179, 2SC3851, 2SC3851A, 2SC4007, 2SC4007-F, 2SC4008, 2SC4008-F, 2SC4550, 2SC4551, 2SC4552, 2SD1133, 2SD1133-D, 2SD1134, 2SD1134-D, 2SD1266, 2SD1266-O, 2SD1266A, 2SD1266A-O, 2SD1274, 2SD1274A, 2SD1274B, 2SD1761, 2SD1761-F, 2SD1762, 2SD1762F, 2SD2012, 2SD2061, 2SD2061F, 2SD2394, 2SD2394-F, 2SD2395, 2SD2395-F, 2SD313, 2SD313F, 2SD613, 2SD613-F, 2SD823, BD203, BD241A, BD241B, BD241C, BD243A, BD243B, BD243C, BD303, BD535, BD537, BD539A, BD539B, BD539C, BD543A, BD543B, BD543C, BD545A, BD545B, BD545C, BD797, BD799, BD801, BD807, BD809, BD949, BD951, BD953, BDT81, BDT81F, BDT83, BDT83F, BDT85, BDT85F, BDX77, D44H11, D44H11FP, D44H8, KSD2012 or KSD2012-G transistors.

2.1.4 Shift register

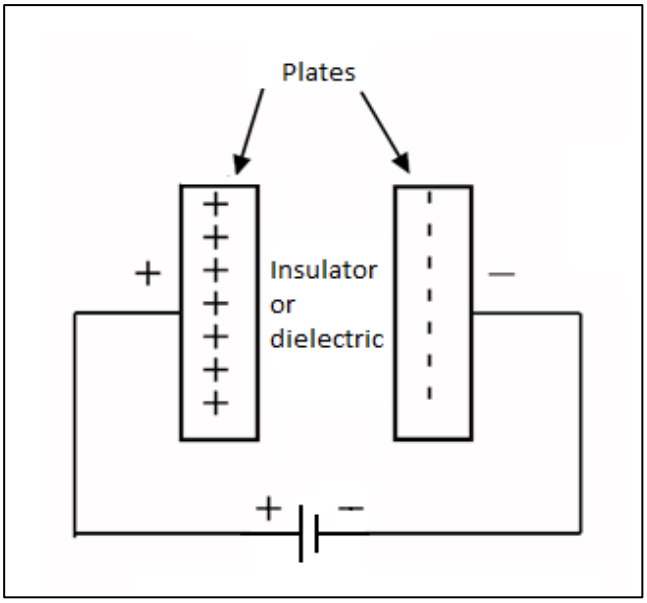
Shift register is a device that loads the data present on its inputs and then moves or shifts it to its output once every clock cycle. Data may be fed in or out of a shift register serially, that is one after the other from either the left or the right direction or all together at the same time in a parallel configuration. In this project, 74HC595 shift registers were used.



2.1.4-1 Pin configuration of 74HC595 shift register

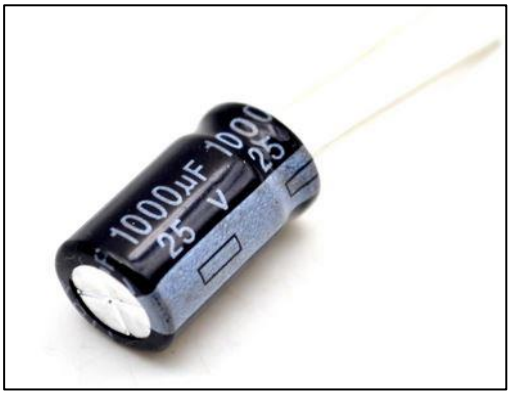
Above figure shows the pin configuration of 74HC595 shift register. The 74HC595 is a high-speed CMOS 8 bit shift register fabricated with silicon gate C2MOS technology with 8 bit D type storage register. This shift register has eight outputs. Here, pin no.15 is the output zero. Pin no 1 - 7 are the other seven outputs. Pin no.8 is the ground pin. Pin no.9 is the VCC pin. We can connect shift registers using Pin no.9 and pin no.14 (If we connect two registers, first register's pin no.9 connect with second register's pin no.14). Pin no.14 is the data input pin. Pin no.11 is for clock signal input. Pin no.12 for latch input. Pin no.10 is direct-overriding clear pin. Both the shift register and storage register use positive-edge triggered clocks. If both clocks are connected together, the shift register state will always be one clock pulse ahead of the storage register. 74HC595BQ, 74HC595D, 74HC595DB, 74HC595N, 74HC595PW, 74HC595U, 74HCT595, 74HCT595BQ, 74HCT595D, 74HCT595DB, 74HCT595N and 74HCT595PW are Replacement and Equivalent components for this 74HC595 shift register.

2.1.5 Capacitor



2.1.5-1 Sketch of the resistor and how chargers behave with it

Capacitor is a device for storing electrical energy, consisting of two conductors in close proximity and insulated from each other. As seen in above figure simple example of capacitor is the parallel-plate capacitor. If positive charges with total charge +Q are deposited on one of the conductors and an equal amount of negative charge −Q is deposited on the second conductor, the capacitor is said to have a charge Q. For this project A 1000 uF capacitor was used in between VCC and ground when the power suppling to the microcontroller.

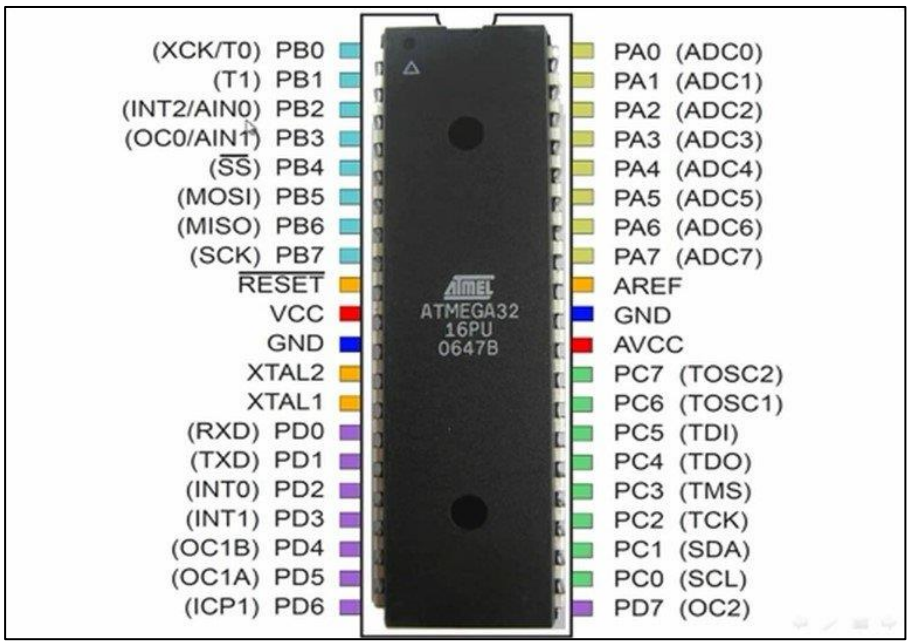


2.1.5-2 1000 uF capacitor.

Above figure shows 1000 uF capacitor appearance. It has Capacitance of 1000µF, Rated voltage is 25 V, tolerance of it is ± 20% and lead spacing is 5.0mm. Usually, all of the information are printed on the capacitor cover. So we can easily identify.

2.1.6 Microcontroller

Microcontrollers are compact integrated circuits designed to control a specific operations in an embedded system. Typical microcontrollers are include a processor, memory and input and output peripherals. It can be called as “Special purpose computers” as well. For this practical ATmega32 microcontroller was used.



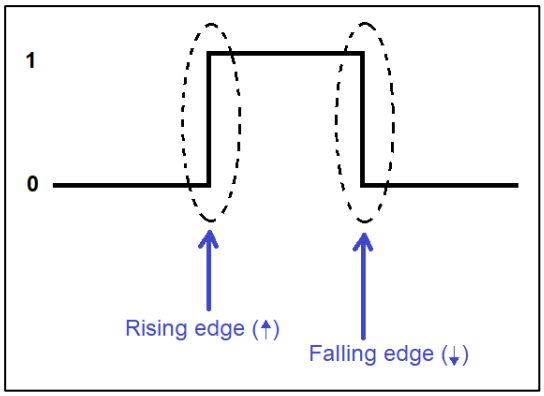
2.1.6-1 ATmega32 microcontroller pin configuration

Above figure shows that ATmega32 microcontroller. It is a low-power CMOS 8-bit microcontroller based on the AVR enhanced RISC architecture. Here this AVR core combines a rich instruction set with 32 general purpose working registers (PORT A,B,C,D and 8 pins for each ports). All the 32 registers are directly connected to the Arithmetic Logic Unit (ALU). And also there are more 8 pins with special purpose for each in ATmega32.

2.2 Used Theories behind components

2.2.1 How shift register works

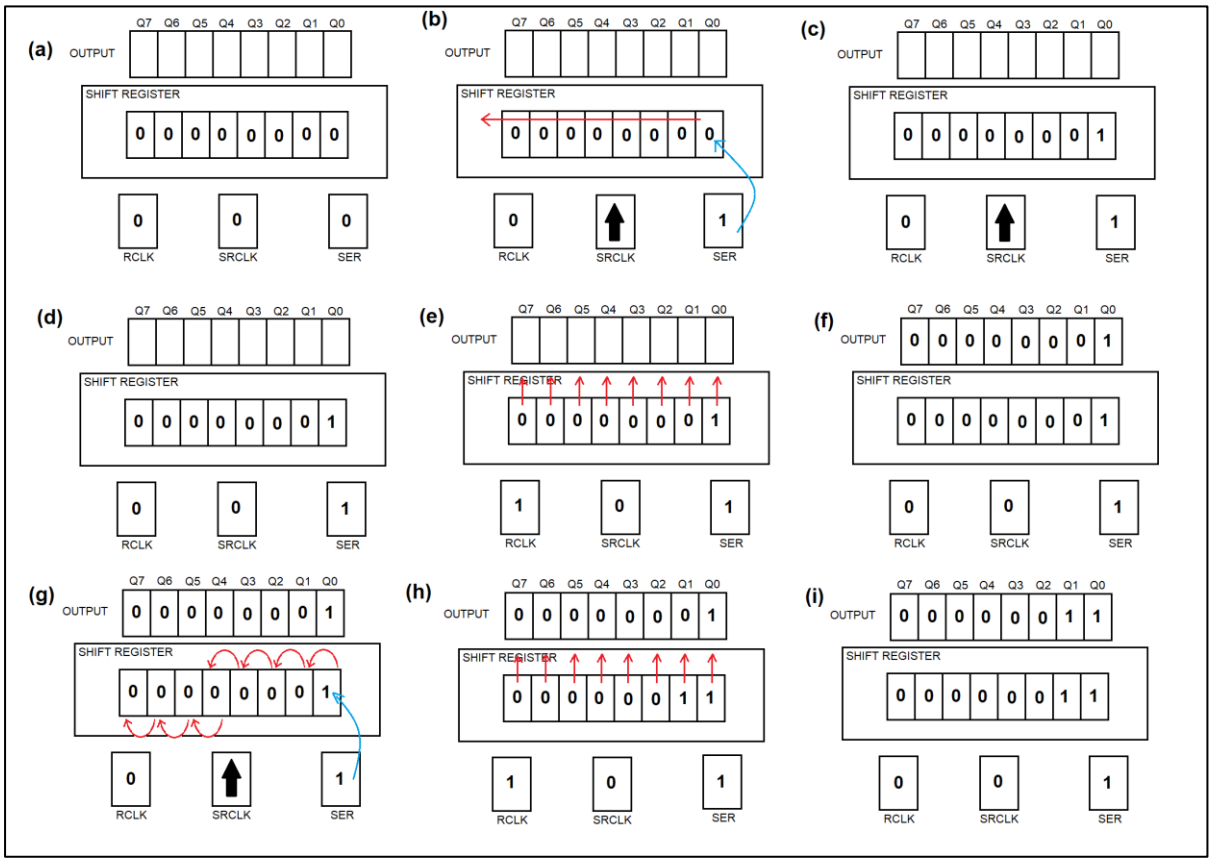
2.2.1.1 Edges of a clock pulse



2.2.1-1. 1 -Rising edge and falling edge

As seen in above figure clock pulse have two types of edges those are called rising edge and falling edge. In this project, rising edge incident was used.

2.2.1.2 How the shift register works



2.2.1-2. 1-How the shift register works with steps

Above figure shows how can we input data to shift register and get output from it.

(a) This shows eight outputs Q0 to Q1. Those are actually pin no.15 is Q0 and pin no.1-7 for Q1 –Q7, Shift register memory inside the shift register and also three input pins. SER stands for pin no.14 and it is the Data input pin, RCLK stands for pin no.12 and it is the data latch pin and SRCLK stands for pin no.11 and it is the clock pin.

(b) As shown in here, suppose we give 1 (in this example we use high state (1) for explanation easiness) to SER. it save in the memory when we make a rising edge SRCLK. As seen in figure the previous data bits are moving towards as well.

(c) Here, we can see the data we input is located in the memory.

(d) Now, SRCLK is zero again and memory has some stored data inputs.

(e) Here, we can see when we make the RCLK as 1 , the stored date latch to the output.

(f) This shows final outputs of the procedure.

(g) Here, we try to give another input to the register. Here the SER is already 1 so when we Give another rising edge to the SRCLK, this 1 update to the last bit of the memory. Previous data bits are moving towards to make the blank.

(h) Here, when we make RCLK high, the new stored data latch to the output.

(i) This shows final outputs of the new procedure.

2.2.2 How microcontroller work

2.2.2.1 Interrupts

An interrupt is a some signal to the main proses indicating an event that need immediate action or attention. There are two kind of interrupts. Those are external and internal interrupts. For activate interrupts there is a library called <avr/interrupt.h>. to enable interrupts Sei() should be add first. There are different registers for interrupts. For external interrupts, there are MCUCR,MCUCSR,GICR and GIFR.

2.2.2.2 Registers

2.2.2.2.1 MCUCR – MCU Control Register

7	6	5	4	3	2	1	0
				ISC11	ISC10	ISC01	ISC00

- **Bits 0:1**
  ISC00 and ISC01 stands for Interrupt Sense Control 0 Bit 0 and Bit 1.those bits are for generate interrupt request in INT0. This bits has 4 different selections.

- **Bits 2:3**
  ISC10 and ISC11 stands for Interrupt Sense Control 1 Bit 0 and Bit 1.those bits are for generate interrupt request in INT1. This bits also has 4 different selections.

2.2.2.2.2 GICR – General Interrupt Control Register

7	6	5	4	3	2	1	0
INT1	INT0	INT2					

- **Bit 5**

  INT2 stands for External Interrupt Request 2 Enable. When the INT2 bit is set (one) the external pin interrupt is enabled.

- **Bit 6**

  INT2 stands for External Interrupt Request 0 Enable. When the INT0 bit is set (one) the external pin interrupt is enabled.

- **Bit 7**

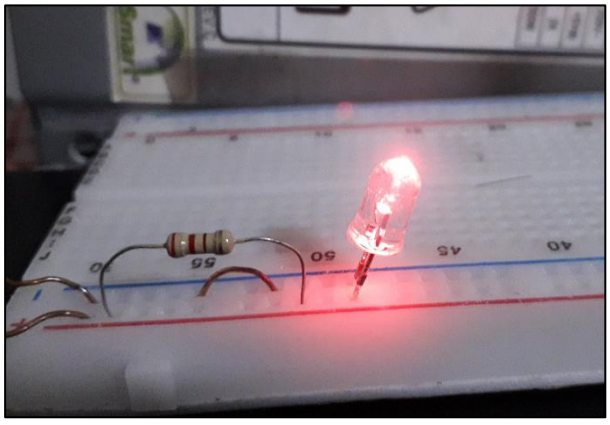
  INT2 stands for External Interrupt Request 1 Enable. When the INT1 bit is set (one) the external pin interrupt is enabled.

3. Methodology

1. All needed components were identified and collected.

Component name	No of items
RED LED	514
Dot Board	2
Male header pins	10
Female header pins	10
74HC595	9
1000uF capacitor	1
220 Ohm Resistor	68
CTC 1061	8
Solder(40 W)	1
Soldering iron	20m
Bucket Wires (1 roll)	3
Jump wires	1
Atmega32	1
74HC595 header	8
Push button	2

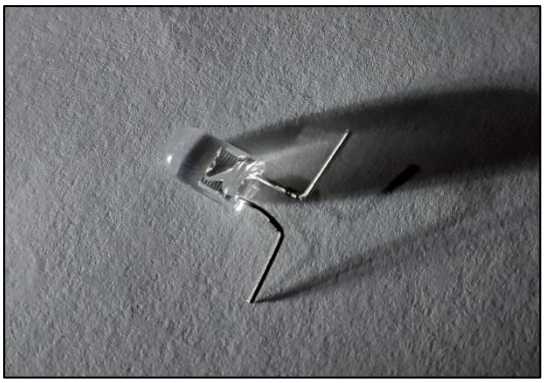
2. All LED were tested and make sure working or not.



3-1 Testing setup of LEDs

As seen in above figure first of all LEDs were tested and checked those are working or not. It's very important to have working LEDs.

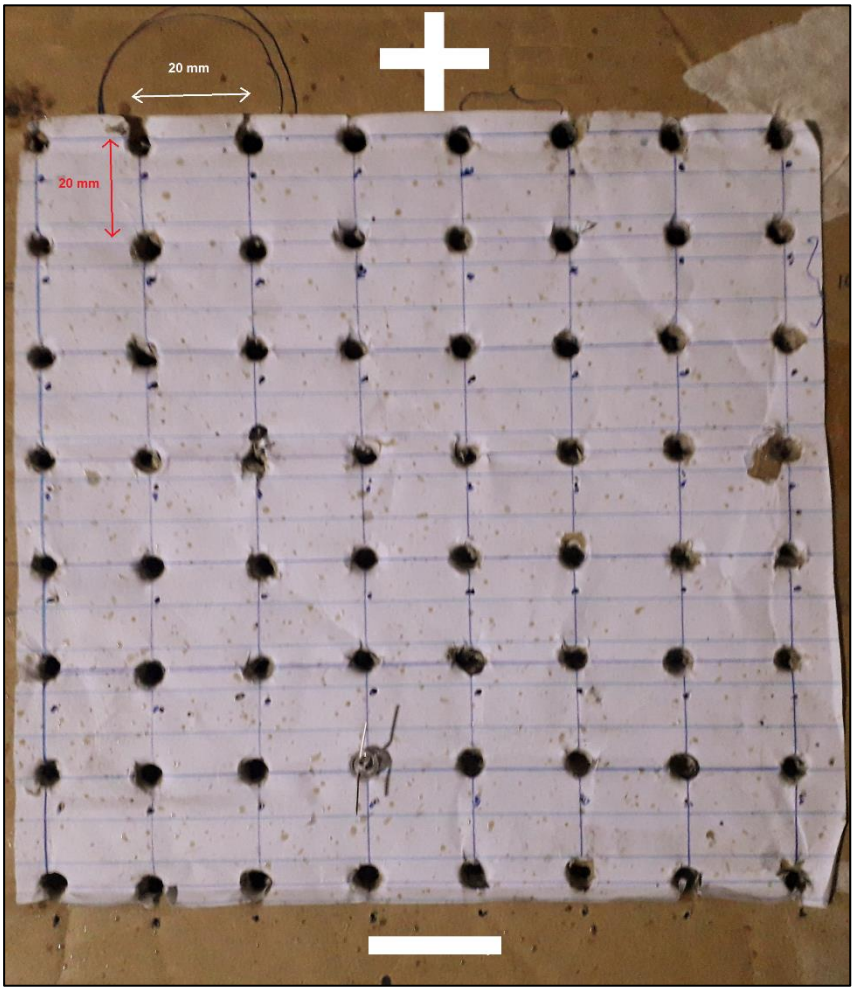
3. Arranged LED for the further working.



3-2 LED arrangement for the setup

After checking LEDs, working LEDs were chosen and arrange As seen in above figure. Bothe anode and cathode were cut and bent opposite directions.

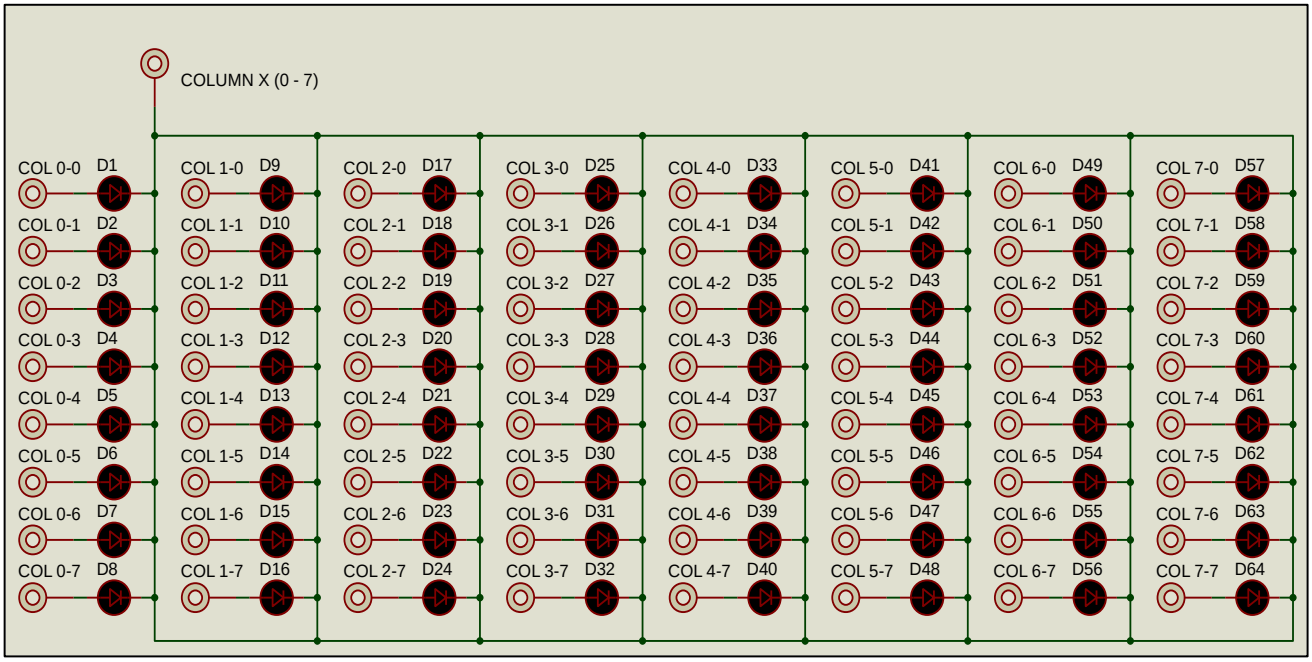
4. Arranged a foundation setup to arrange LEDs properly in 8×8 metrics.



3-3 Foundation arrangement (8 × 8 matrix setup)

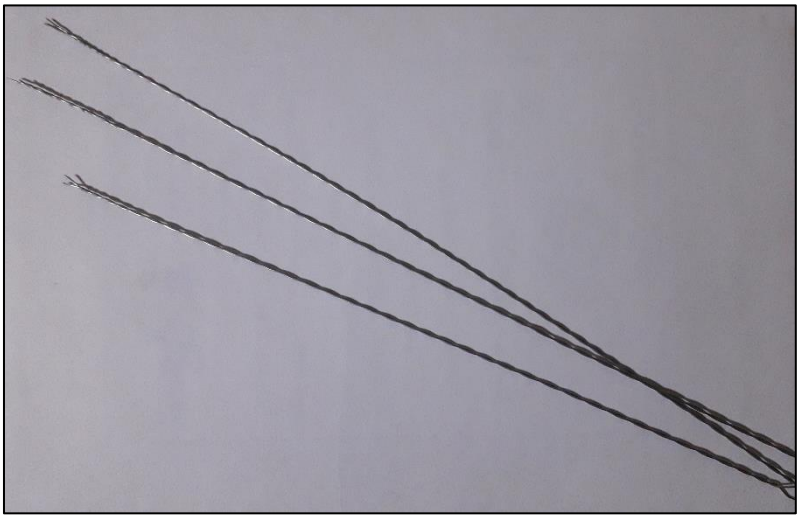
As seen in above figure the foundation setup was arranged using thick cardboard. 140 mm long square was drawn and Split it to 20 mm squares carefully. As seen in figure holes were created every edge of squares (each hole must not be very small or big. It must be enough for include an LED to the hole). Cathode side and anode side were defined and marked one opposite side.

5. Arrange LED



3-4 How to solder one LED layer

After arranging the foundation circuit LEDs were arranged considering corresponding LED anode with predefined anode side of the setup and cathode as well. Then, all cathodes were connected by using bucket wires and soldered. Bucket wires are not strong enough. So wires were twisted to make strong.

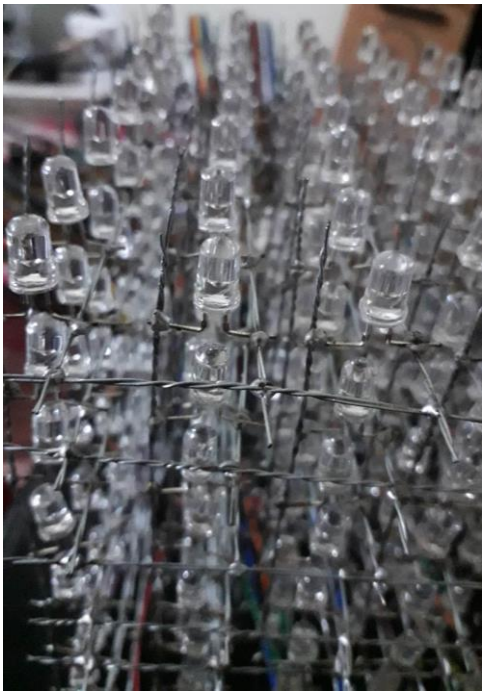


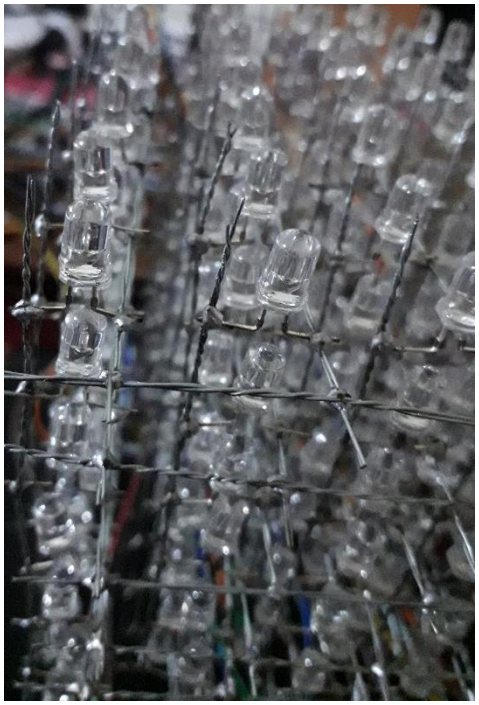
3-5 Twisted bucket wires

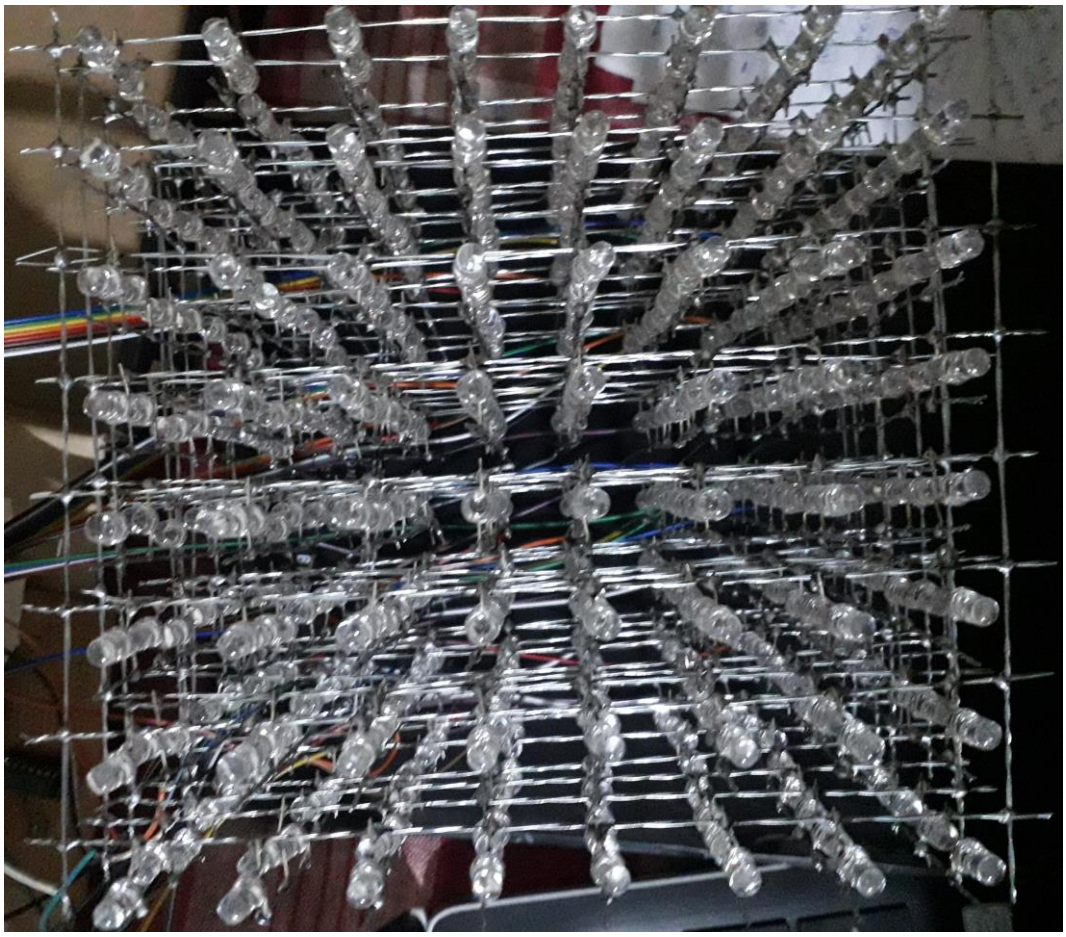
As seen in above figure bucket wires were arranged by twisting properly. 144 wires required for the setup (10 wires for one level × 8 + 64 wires for connect all levels together). All soldering must do very slowly and very carefully without damaging LEDs or previous soldered parts.

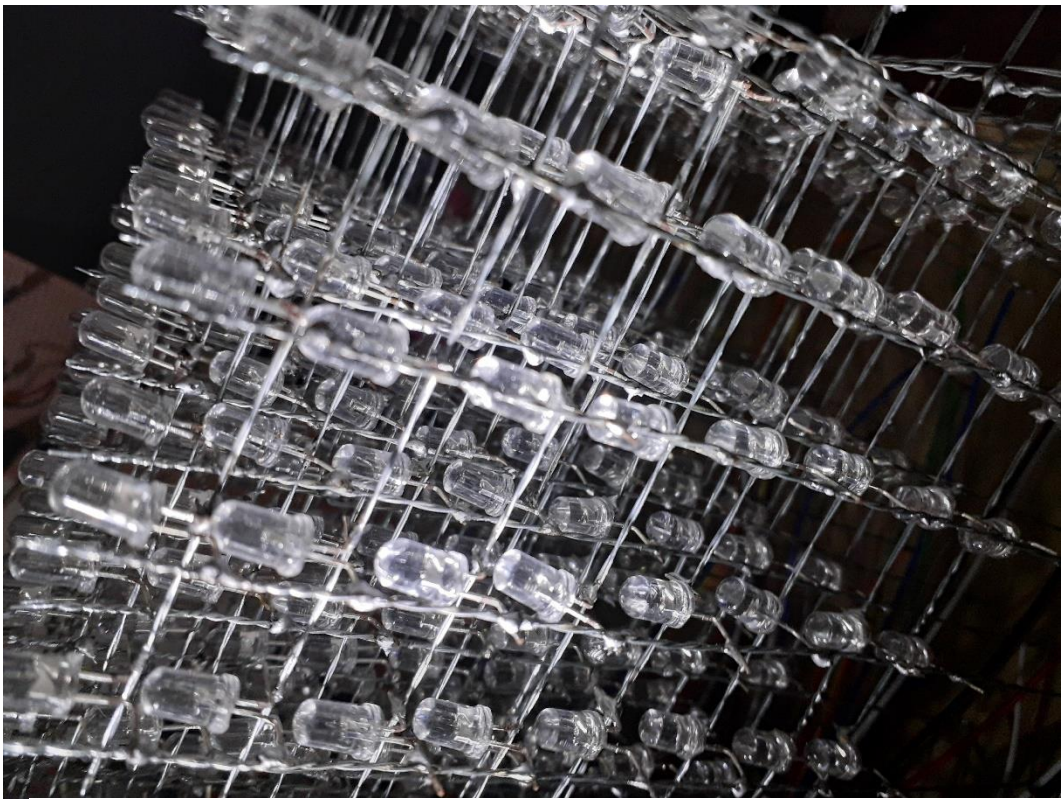
- One level took more than three and half hours to complete (arranging 64 LEDs + twisting 10 wires + soldering)
- It took more than six hours to soldering all levels together.

6. Connect all levels together and finish the cube.



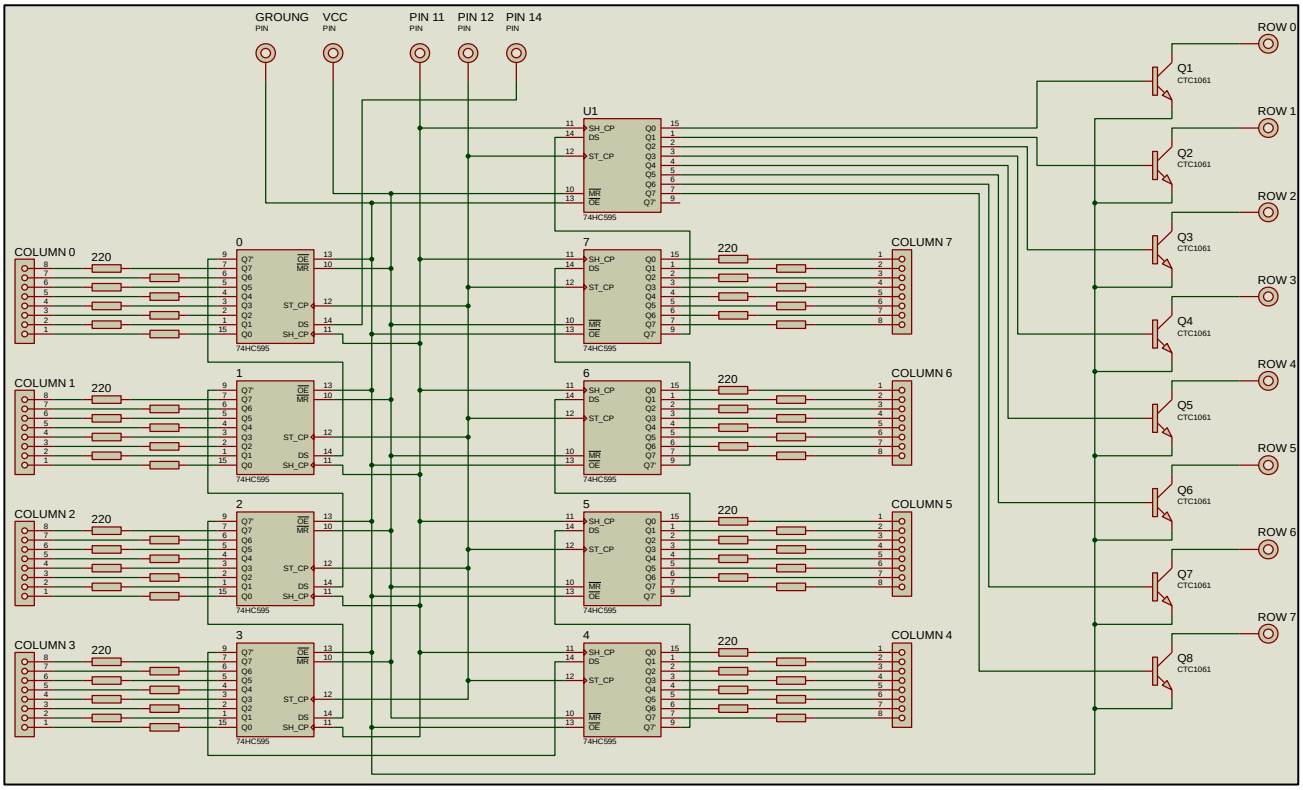






3-6 Figure panel of finished LED cube

7. Design the main circuit.

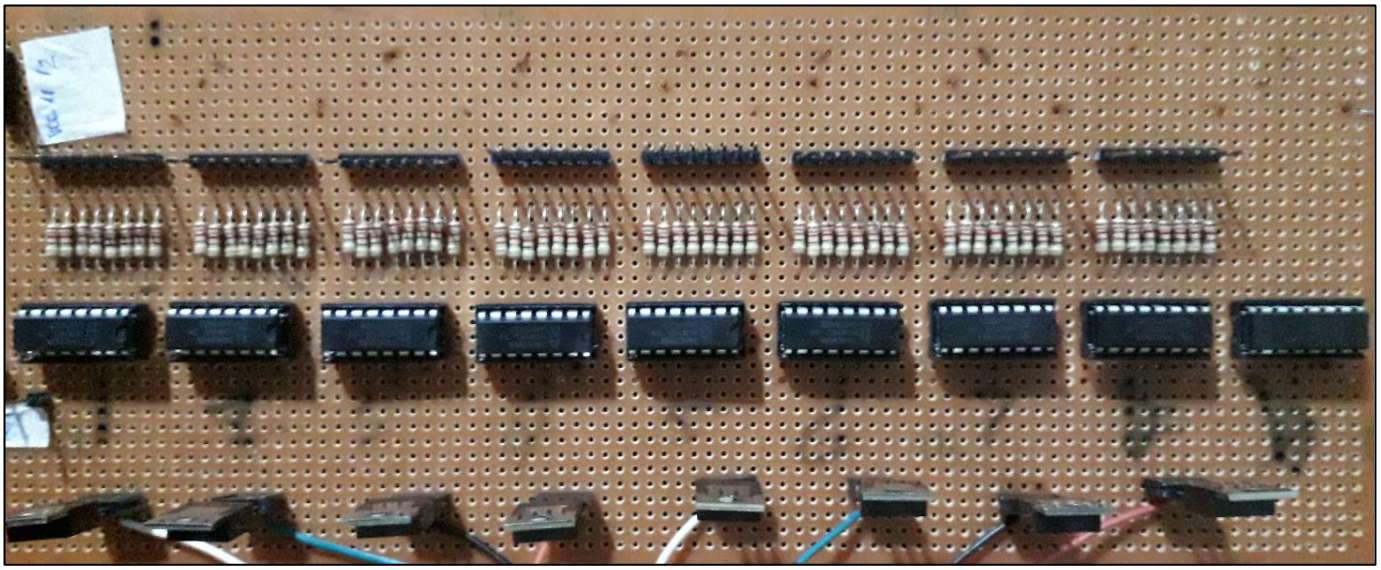


3-7 Main circuit diagram design

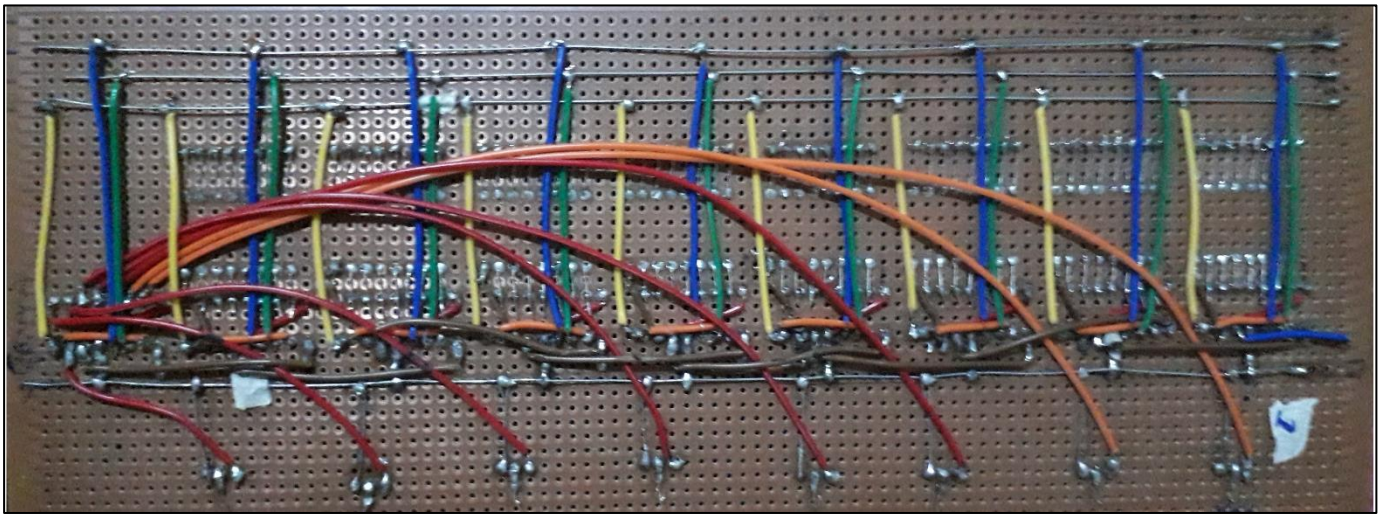
As seen in above figure the main circuit was designing including nine 74HC595 shift registers 220 Ω resistors (64 resistors), eight CTC1061 transistors, male and female pin headers and wire connections.

- Shift registers (74HC595) – eight shift registers' all 8 outputs (pin 15, 1, 2, 3, 4, 5, 6 and 7) were connected with LED cube columns. Last shift register all outputs were connected with transistor's pin 1. Pin 8 and 13 were connected with VCC. Pin10 and 16 were connected to the ground. Pin 11,12 and 14 for communication.
- Resisters (220 Ω) – resistors were used here between shift resistors and LEDs.
- Transistors (CTC1061) – Pin 1 was connected with shift resister output. Pin2 was connected LED cube row. Pin3 was connected with ground.
- Headers – headers were used to connects columns rows because of maintaining easiness. shift registers and transistors are were sensitive devices therefore those were also connected with using headers.

8. Built the main circuit.



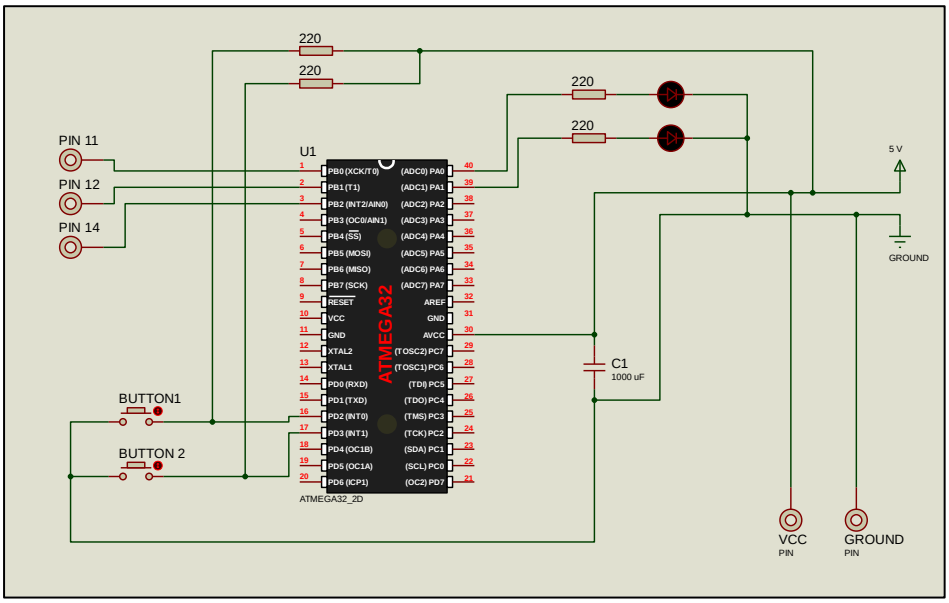
3-8 Front view of the main circuit



3-9 Back view of the main circuit

Above two figures shows the built main circuit. This circuit took lot of time to finish. It was very important soldering carefully. No pin or wire should be short circuit. All the ground pins, VCC pins, pin no 11 and 12 were connected all together.

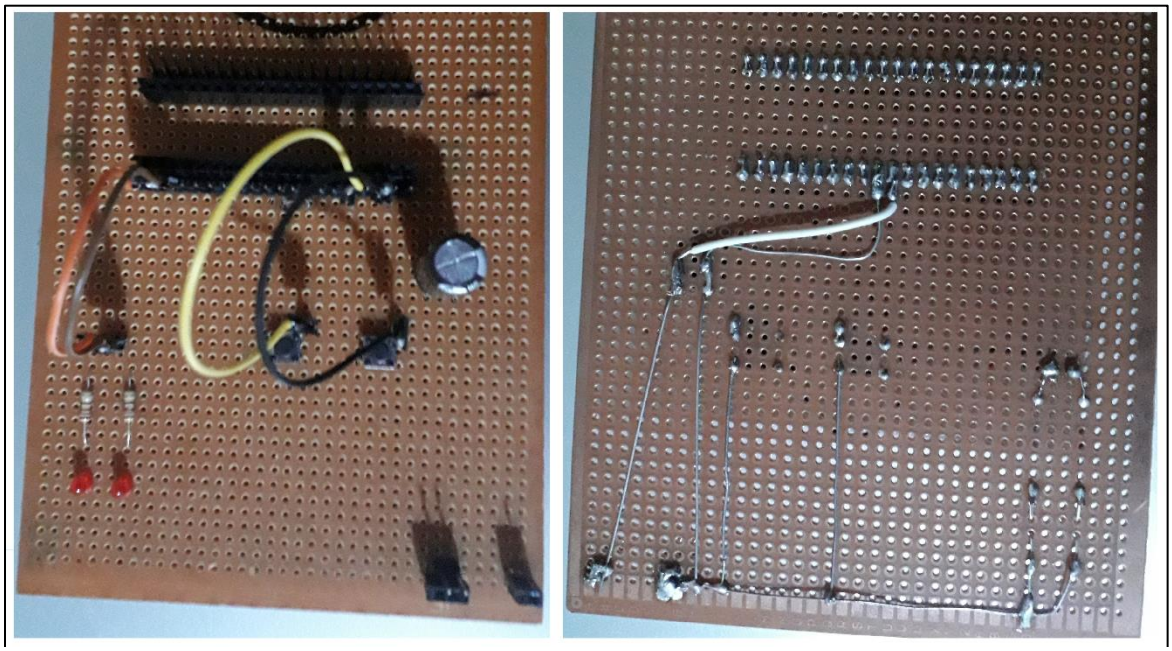
9. Design the Atmega32 circuit.



3-10 Circuit diagram of the ATmega32 circuit

Atmega32 pins were used as seen in above figure. 1000 uF resistor was used between VCC and ground.

10. Built the ATmega32 circuit



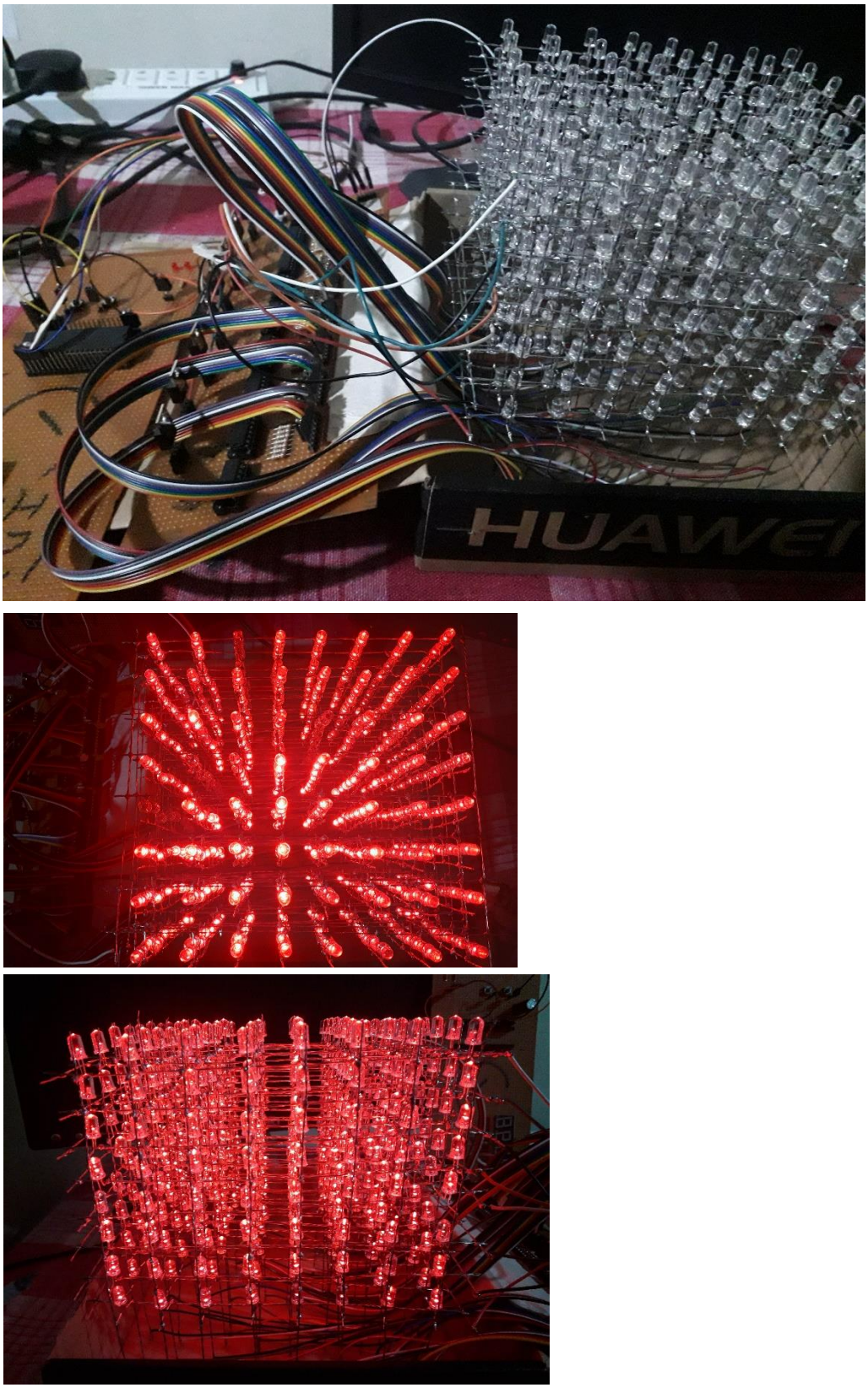
3-11 Front and back of the Atmega32 circuit

Above figure shows Front and back view of the Atmega32 circuit. As seen in there 2 LEDs, 2 push buttons, headers for Atmega32, A 1000 uF capacitor and Pins for connect Power supply were soldered on a zero pin dot board.

11. Different LED patters were coded using winAVR programme.
12. Codes were uploaded using USBasp cable and ATmega32 development board.
13. 5 V, 4 A power supply were connected to the circuit.

4. Results

4.1 Final setup



4-1 Figure panel of the final setup

5. Discussion

I have successfully completed the circuit building part, LED building part and connecting all together. When, considering the coding part, giving interrupts and making very basic patterns were successfully done but could not complete very complete meaningful code. The code on the AVR consists of two main parts. The cube interrupt routine (external interrupts by INT0 and INT1) and effect code for fancy animation on the cube. It was not easy to soldering LED cube and main circuit. When I finally finished soldering, I thought coding part would be the easy part. Nevertheless, it turned out that making fancy and animation in the cube was very harder than is sounds. However, I could make very primary patterns eventually.

So many problems encountered during the project. At the beginning, collecting components were not easy and some components were not able find. Therefore, I changed the circuit using equivalent components. When the soldering I realize soldering iron is always not enough as estimated. I lost five or six LEDs while testing them and soldering them. At the beginning, I could not find any suitable wire to build up the cube structure. Then, I decided to use bucket wires. Those are not strong enough to bare the structure then I had to twist them. Actually, it cannot use any wire because some wires cannot solder using usual soldering irons (some copper wire not good enough). While, doing coding I realized it is better to study about how shift register transfer data. Therefore, I studied about that using its datasheet.

The LED cube is very fancy thing. Therefore, there is so many possibilities to develop. Here in this project I used single color LEDs but using pulse width modulation and RGB LEDS the cube can rebuilt more fancier than single color one. Modify the cube with having little bit more space in between levels and same level also (expand the size of the cube) or using small LEDs can increase the resolution if the cube, It can develop the interrupting part to some remote controlling method.

6. Reference

Atmel®, 2016. Atmel ATmega32A [DATASHEET]. In: *Atmel-8155I-ATmega32A_Datasheet_Complete-08/2016.* Atmel-8155I-ATmega32A_Datasheet_Complete-08/2016 ed. s.l.:Atmel®, pp. 69-72.

semiconductors, P., 2003 Jun 25. *74HC595;74HCT595 DATA SHEET.* s.l.:Philips semiconductors .

Shift Register - Parallel and Serial Shift Register (electronics-tutorials.ws)

WinAVR Tutorial | AVR Freaks

YouTube

7. Appendix

7.1 Budget

Table 6.1.1 Table of components with price

Component name	**No of items**	**price of one item**	**Total Price**
RED LED	514	Rs. 3.00	Rs. 1,542.00
Dot Board	2	Rs. 125.00	Rs. 250.00
Male header pins	10	Rs. 25.00	Rs. 250.00
Female header pins	10	Rs. 25.00	Rs. 250.00
74HC595	9	Rs. 50.00	Rs. 450.00
1000uF capacitor	1	Rs. 25.00	Rs. 25.00
220 Ohm Resistor	68	Rs. 1.00	Rs. 68.00
CTC 1061	8	Rs. 50.00	Rs. 400.00
Solder(40 W)	1	Rs. 600.00	Rs. 600.00
Soldering iron (1m = Rs.30.00)	20	Rs. 30.00	Rs. 600.00
Bucket Wires (1 roll = Rs.60.00)	3	Rs. 60.00	Rs. 180.00
Jump wires	1	Rs. 300.00	Rs. 300.00
Atmega32	1	Rs. 800.00	Rs. 800.00
74HC595 header	8	Rs. 15.00	Rs. 120.00
Push button	2	Rs. 5.00	Rs. 10.00
Total Price			**Rs. 5,845.00**

(Components were bought from Vijitha electronics Kandy and Sanjeewa electronics Kandy on 2021.07.07)

7.2 The code

```
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>

#define SIFT_PORT PORTB
#define SIFT_DDR DDRB

#define SIFT_DS_POS PB2//Data pin location

#define SIFT_SH_CP_POS PB0//Shift Clock (SH_CP) pin location
#define SIFT_ST_CP_POS PB1//Store Clock (ST_CP) pin location

#define BUTTON_PORT PORTD
#define BUTTON_DDR DDRD

#define BUTTON_LED_PORT PORTC
#define BUTTON_LED_DDR DDRC

#define SIFTDataHigh() (SIFT_PORT|=(1<<SIFT_DS_POS))//Low level macros to change
data (DS)lines
#define SIFTDataLow() (SIFT_PORT&=(~(1<<SIFT_DS_POS)))//Sends a clock pulse on
SH_CP line

volatile unsigned int p=0, btn = 0;

void SIFTInit(){
   SIFT_DDR|=((1<<SIFT_SH_CP_POS)|(1<<SIFT_ST_CP_POS)|(1<<SIFT_DS_POS));
//Make the Data(DS), Shift clock (SH_CP), Store Clock (ST_CP) lines output

}

void SIFTPulse(){//Pulse the Shift Clock

   SIFT_PORT|=(1<<SIFT_SH_CP_POS);//HIGH
   SIFT_PORT&=(~(1<<SIFT_SH_CP_POS));//LOW
}

void SIFTLatch(){//Sends a clock pulse on ST_CP line

  SIFT_PORT|=(1<<SIFT_ST_CP_POS);//HIGH
  _delay_loop_1(10000);

  SIFT_PORT&=(~(1<<SIFT_ST_CP_POS));//LOW
  _delay_loop_1(10000);
}
```

```
void SIFTWrite(int data){//Send each 8 bits serially

  for(int i=0;i<8;i++){//Order is MSB first

    if(data & 0b10000000){//Output the data on DS line according to the

      SIFTDataHigh();//MSB is 1 so output high
    }else{

     SIFTDataLow();//MSB is 0 so output high
    }

    SIFTPulse();  //Pulse the Clock line
    data=data<<1;  //Now bring next bit at MSB position
  }

  SIFTLatch();//Now all 8 bits have been transferred to shift register NOW Move them
to output latch at one
}

void Wait(){
  for(int i=0;i<20;i++){

    _delay_loop_2(10000);
  }
}

int main(){
  BUTTON_LED_DDR |= 1<< PC0 | 1<< PC1;

  BUTTON_LED_PORT &= ~ (1<< PC0 | 1<< PC1);

  BUTTON_DDR &= ~(1<<PD2);//inturrupt0
  BUTTON_LED_PORT &= ~(1<< PA0);

  GICR |= (1<<INT0 | 1<<INT1);
  MCUCR |= 1<<ISC10| 1<<ISC11 | 1<<ISC00| 1<<ISC01 ;
  sei();
  btn=0;
  p=0;

  SIFTInit();

  while(1){
    if(p==0){
      int led_pattern[8]={
        0b11111111,
```

```
        0b11000111,
        0b11100111,
        0b11111111,
        0b01111110,
        0b00111100,
        0b00011000,
        0b00000000,
        };
        for(int i=0;i<8;i++){
          SIFTWrite(led_pattern[i]);//Write the data to sHIFT REGISTER
          Wait(); //Wait
        }
    }else if (p==1){
      int led_pattern[8]={
        0b11111111,
        0b11111110,
        0b11111100,
        0b11111000,
        0b11110000,
        0b11100000,
        0b11000000,
        0b10000000,
        };
        for(int i=0;i<100;i++){
          SIFTWrite(led_pattern[i]);//Write the data to sHIFT REGISTER
          Wait(); //Wait
        }
    }else if (p==2){
      int led_pattern[8]={
        0b11111111,
        0b11000111,
        0b11100111,
        0b11111111,
        0b01111110,
        0b00111100,
        0b00011000,
        0b00000000,
        };
        for(int i=0;i<20;i++){
          SIFTWrite(led_pattern[i]);//Write the data to sHIFT REGISTER
          Wait(); //Wait
        }
    }else if (p==3){
      int led_pattern[8]={
        0b00000000,
        0b11111111,
        0b00000000,
```

```
        0b00000000,
        0b11111111,
        0b00000000,
        0b00000000,
        0b11111111,
        };
        for(int i=0;i<20;i++){
          SIFTWrite(led_pattern[i]);//Write the data to sHIFT REGISTER
          Wait(); //Wait
        }
      }else if (p==4){
        int led_pattern[8]={
        0b01111111,
        0b00111111,
        0b00011111,
        0b00001111,
        0b00000111,
        0b00000011,
        0b00000001,
        0b00000000,
        };
        for(int i=0;i<20;i++){
          SIFTWrite(led_pattern[i]);//Write the data to sHIFT REGISTER
          Wait(); //Wait
        }
      }else if (p==5){
        int led_pattern[8]={
        0b11111111,
        0b00000000,
        0b11111111,
        0b00000000,
        0b11111111,
        0b00000000,
        0b11111111,
        0b00000000,
        };
        for(int i=0;i<20;i++){
          SIFTWrite(led_pattern[i]);//Write the data to sHIFT REGISTER
          Wait(); //Wait
        }
      }else if (p==6){
        int led_pattern[8]={
        0b11111111,
        0b01111110,
        0b00111100,
        0b00111100,
        0b01111110,
```

```
        0b11111111,
        0b01111110,
        0b00111100,
        };
        for(int i=0;i<8;i++){
          SIFTWrite(led_pattern[i]);//Write the data to sHIFT REGISTER
          Wait(); //Wait
        }
      }else if (p==7){
        int led_pattern[8]={
        0b01010101,
        0b10101010,
        0b01010101,
        0b10101010,
        0b01010101,
        0b10101010,
        0b01010101,
        0b10101010,
        };
        for(int i=0;i<40;i++){
          SIFTWrite(led_pattern[i]);//Write the data to sHIFT REGISTER
          Wait(); //Wait
        }
      }else if (p==8){
        int led_pattern[8]={
        0b01010101,
        0b10101010,
        0b01010101,
        0b10101010,
        0b01010101,
        0b10101010,
        0b01010101,
        0b10101010,
        };
        for(int i=0;i<20;i++){
          SIFTWrite(led_pattern[i]);//Write the data to sHIFT REGISTER
          Wait(); //Wait
        }
      }else if (p==9){
        int led_pattern[8]={
        0b10000001,
        0b10000001,
        0b01111110,
        0b01111110,
        0b10000001,
        0b10000001,
        0b01111110,
```

```
            0b01111110,
            };
            for(int i=0;i<50;i++){
              SIFTWrite(led_pattern[i]);//Write the data to sHIFT REGISTER
              Wait(); //Wait
            }
        }else{
          int led_pattern[8]={
            0b11111111,
            0b11000111,
            0b11100111,
            0b11111111,
            0b01111110,
            0b00111100,
            0b00011000,
            0b00000000,
            };
            for(int i=0;i<8;i++){
              SIFTWrite(led_pattern[i]);//Write the data to sHIFT REGISTER
              Wait(); //Wait
            }
        }
    }
return 0;
}

ISR(INT0_vect){

   p++; btn++;
   BUTTON_LED_PORT = 0b00000001;
   _delay_ms(100);
   BUTTON_LED_PORT = 0x00;

}
ISR(INT1_vect){

   p--; btn--;
   BUTTON_LED_PORT = 0b00000010;
   _delay_ms(100);
   BUTTON_LED_PORT = 0x00;

}
```