DLITHE INTERNSHIP PROJECT REPORT

TOPIC: CONTROLLING LIGHTS AND FAN USING IR REMOTE

BHARATH D
EMAIL-4NM18EC038@NMAMIT.IN / BHARATHDNAYAR@GMAIL.COM
CONTACT :7899216194

Table of Contents

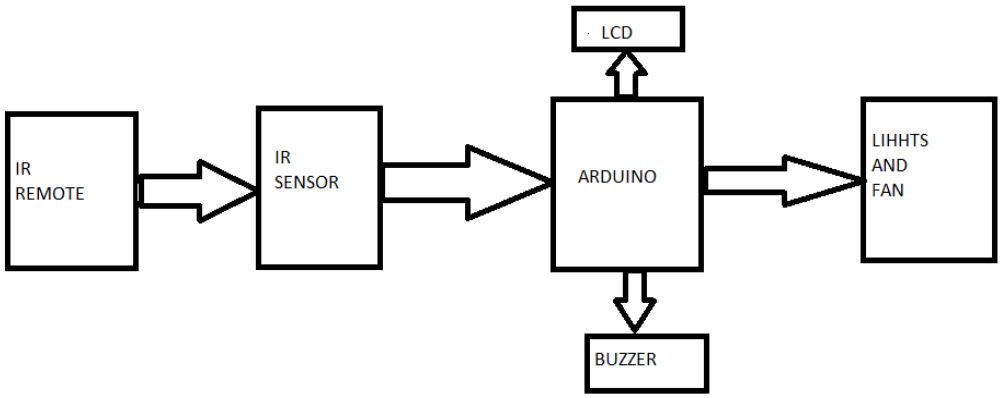
1.INTRODUCTION:

IR or infrared communication is one of the most common methods of wireless communication due to being easy to use and having an affordable price. Infrared light, with a wavelength longer than visible light, is not within the range of human vision. That's why it's a good option for wireless communications. When you press a button on your TV control, an LED on your control turns on and off continuously and causes a modulated infrared signal to send from the control to your TV. The command will execute after the signal is demodulated. IR receiver modules are used to receive IR signals. These modules work in 3, 8 KHz frequency. When the sensor is not exposed to any light at its working frequency, the Vout output has a value equal to VS (power supply). With exposing to a 38 kHz infrared light, this output will be zero.

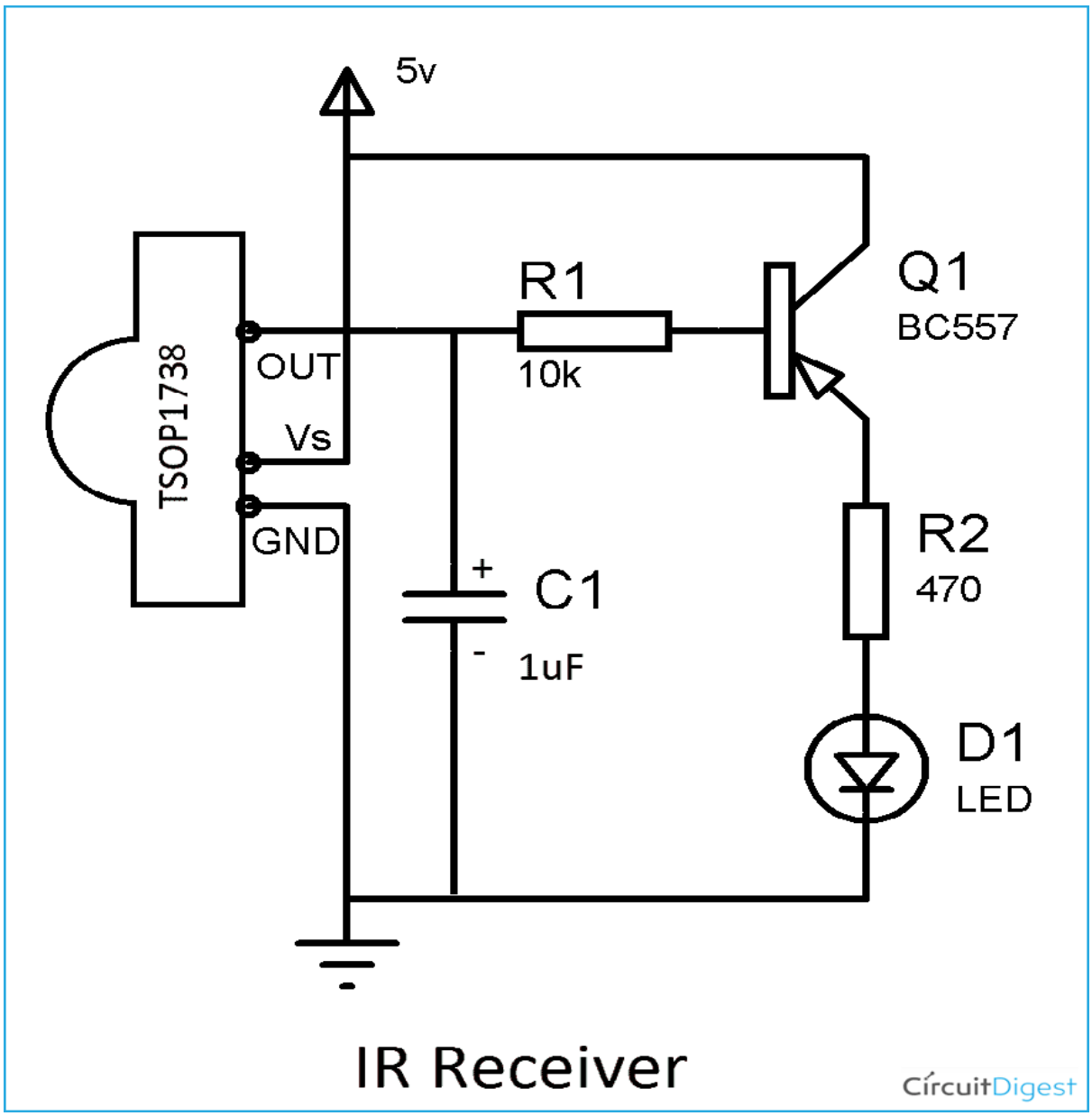
In this project IR remote , IR sensor with Arduino is used. Numbers like 1,2,3 is used to turn on/off the led's. And number 4 for to on/off the fan.

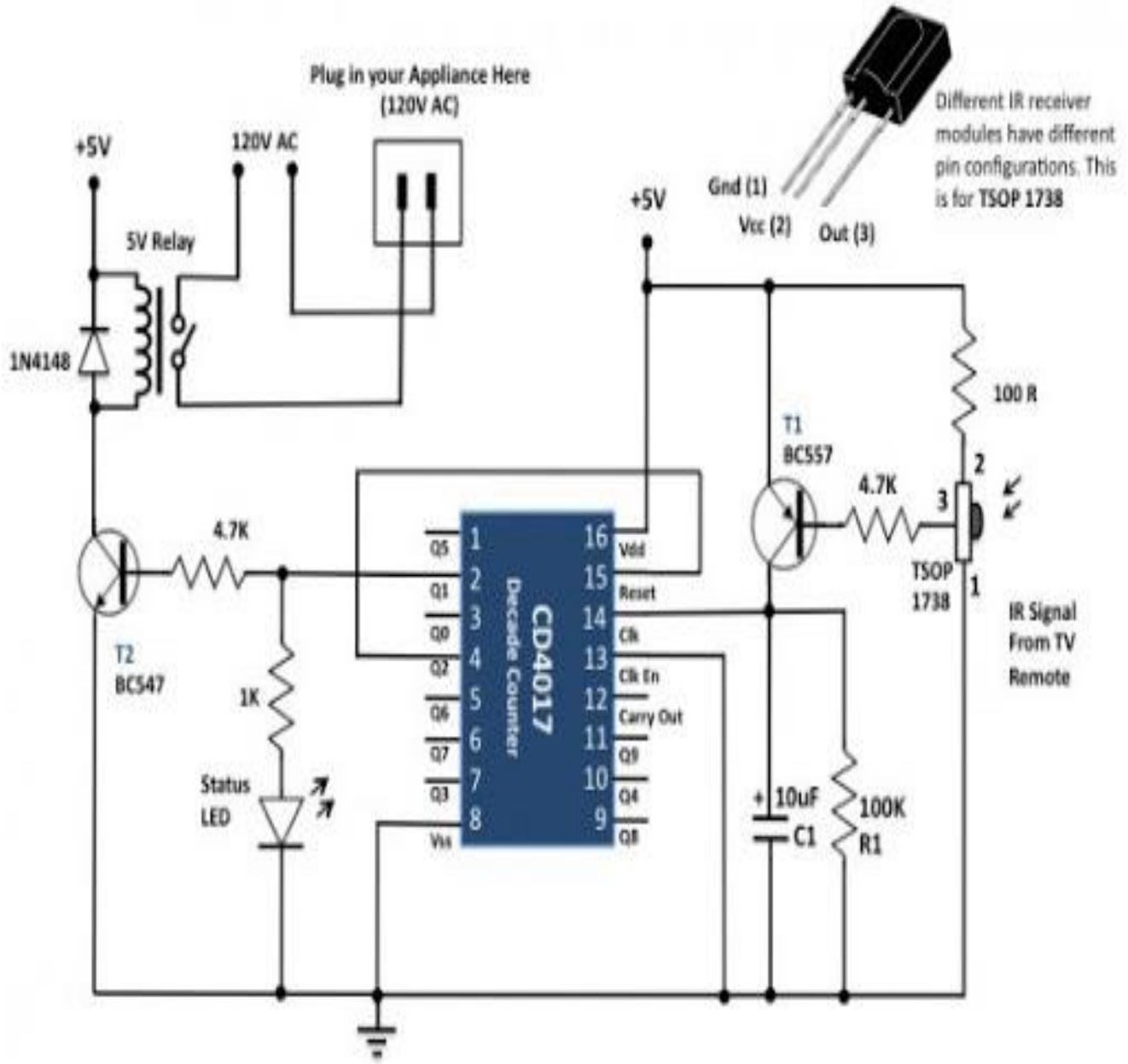
It is a case of the single room where it has 3 led and one fan . If all the lights and fan's are On here is a lcd display to warn "heavy power consumption " with a buzzer to notify .

2.DESIGN WITH BLOCK DIGRAM:



3.CIRCUIT DIAGRAM

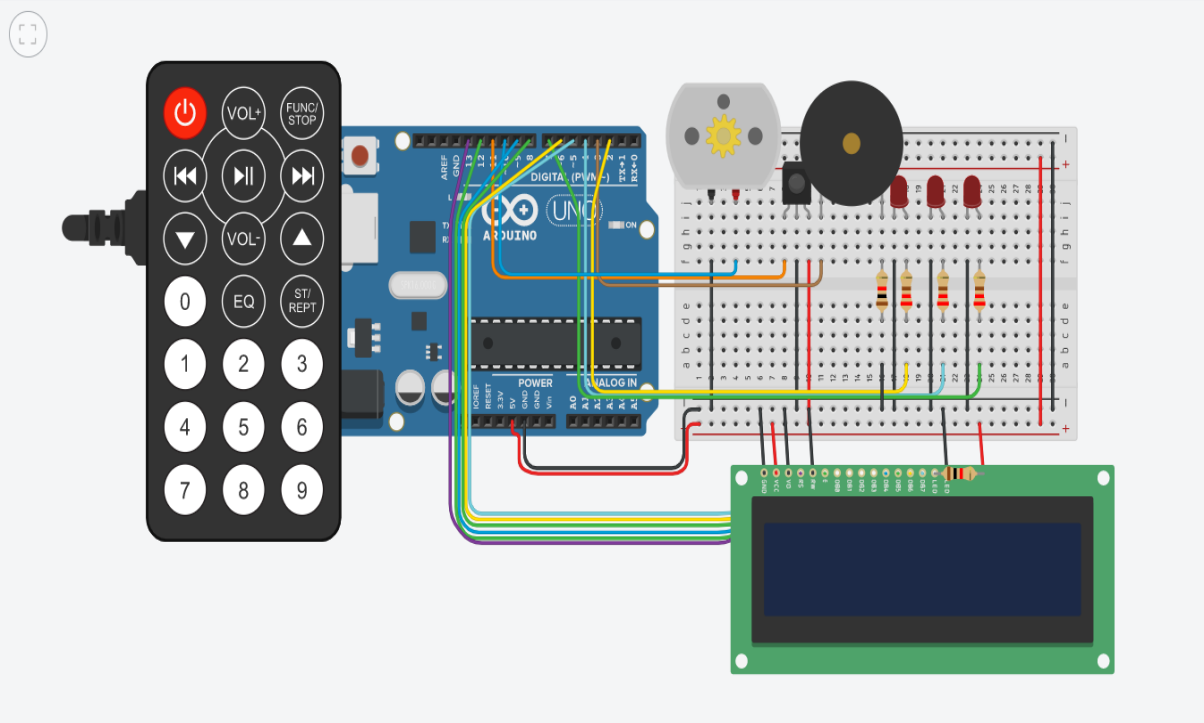



Plug in your Appliance Here
(120V AC)
+5V
120V AC
5V Relay
1N4148
4.7K
T2
BC547
1K
Status
LED
CD4017
Decade Counter
Q5
Q1
Q0
Q2
Q6
Q7
Q3
Vss
Vdd
Reset
Clk
Clk En
Carry Out
Q9
Q4
Q8
+5V
Gnd (1)
Vcc (2)
Out (3)
Different IR receiver modules have different pin configurations. This is for TSOP 1738
100 R
T1
BC557
4.7K
TSOP
1738
IR Signal
From TV
Remote
10uF
C1
100K
R1

(3A).WORKING:

Considering the case of the room where it has three led's and one fan . Which is automated by a remote control and a lcd display with buzzer. In remote, numbers are assigned for led's and fan. Like number 1,2,3&4 for led 1,2,3&fan respectively. Once the remote key is pressed it turns on or off the component depending on previous state .For each key is pressed in remote which component is on is notified in the lcd screen. If all led's and fan of the room is on we get buzzer sound with lcd alert 'HEAVY POWER' mode. If any one component is turned off then buzzer sound and heavy power notification turns off.

4.SIMULATION CIRCUIT



5.COMPONENTS LIST:

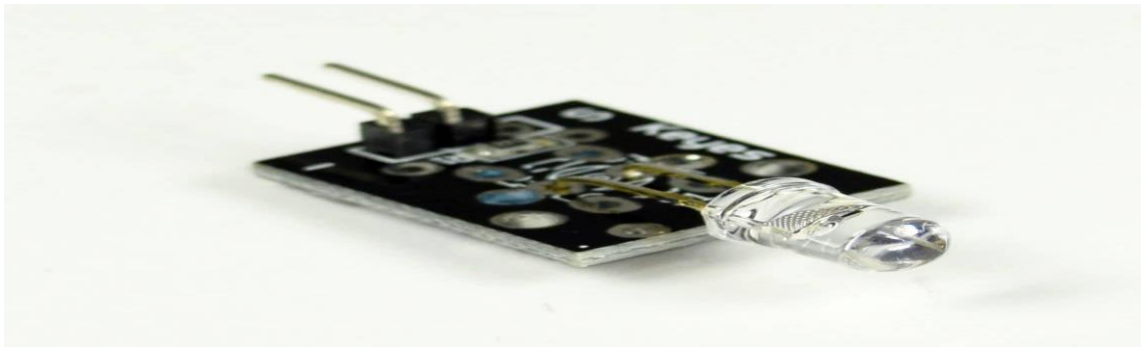
Name	Quantity	Component
U1	1	Arduino Uno R3
U2	1	IR sensor
D1 D2 D3	3	Red LED
R2 R3 R1	3	220 Ω Resistor
M1	1	DC Motor
U4	1	LCD 16 x 2
R4 R5	2	1 kΩ Resistor
PIEZO1	1	Piezo

6.ABOUT SOME COMPONENTS:

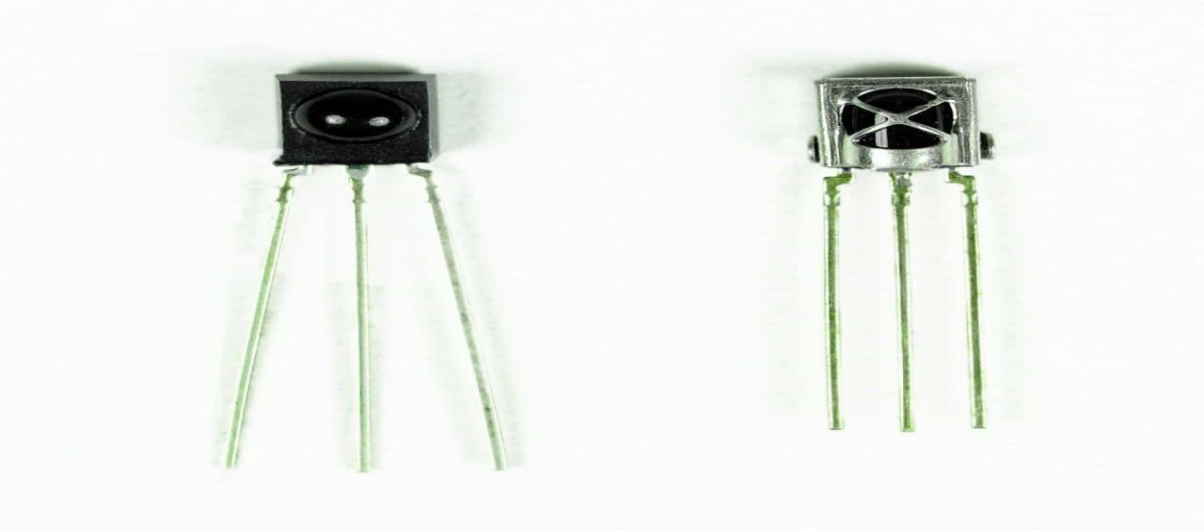
(6A)IR remote and sensors:

A typical infrared communication system requires an IR transmitter and an IR receiver. The transmitter looks just like a standard LED, except it produces light in the IR spectrum instead of the visible spectrum. If you have a look at the front of a TV remote, you'll see the IR transmitter LED:

The same type of LED is used in IR transmitter breakout boards for the Arduino. You can see it at the front of this Keyes IR transmitter:



The IR receiver is a photodiode and pre-amplifier that converts the IR light into an electrical signal. IR receiver diodes typically look like this:

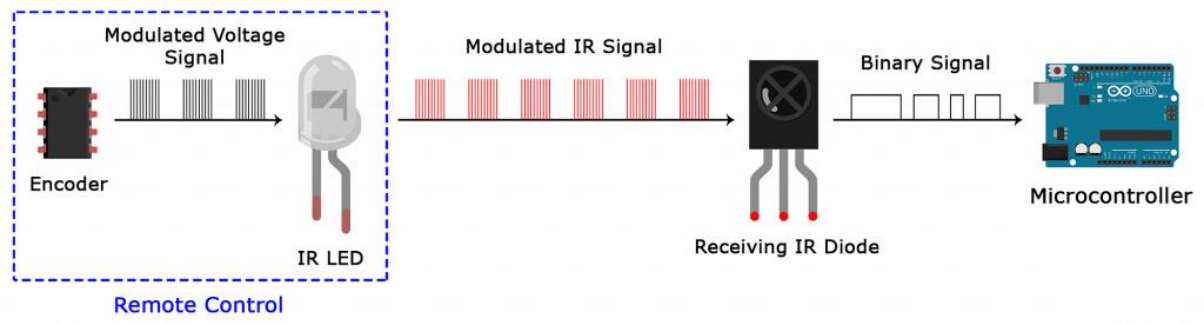


Some may come on a breakout board like this.

7.IR SIGNAL MODULATION

IR light is emitted by the sun, light bulbs, and anything else that produces heat. That means there is a lot of IR light noise all around us. To prevent this noise from interfering with the IR signal, a signal modulation technique is used.

In IR signal modulation, an encoder on the IR remote converts a binary signal into a modulated electrical signal. This electrical signal is sent to the transmitting LED. The transmitting LED converts the modulated electrical signal into a modulated IR light signal. The IR receiver then demodulates the IR light signal and converts it back to binary before passing on the information to a microcontroller:



The modulated IR signal is a series of IR light pulses switched on and off at a high frequency known as the carrier frequency. The carrier frequency used by most transmitters is 38 kHz, because it is rare in nature and thus can be distinguished from

ambient noise. This way the IR receiver will know that the 38 kHz signal was sent from the transmitter and not picked up from the surrounding environment.

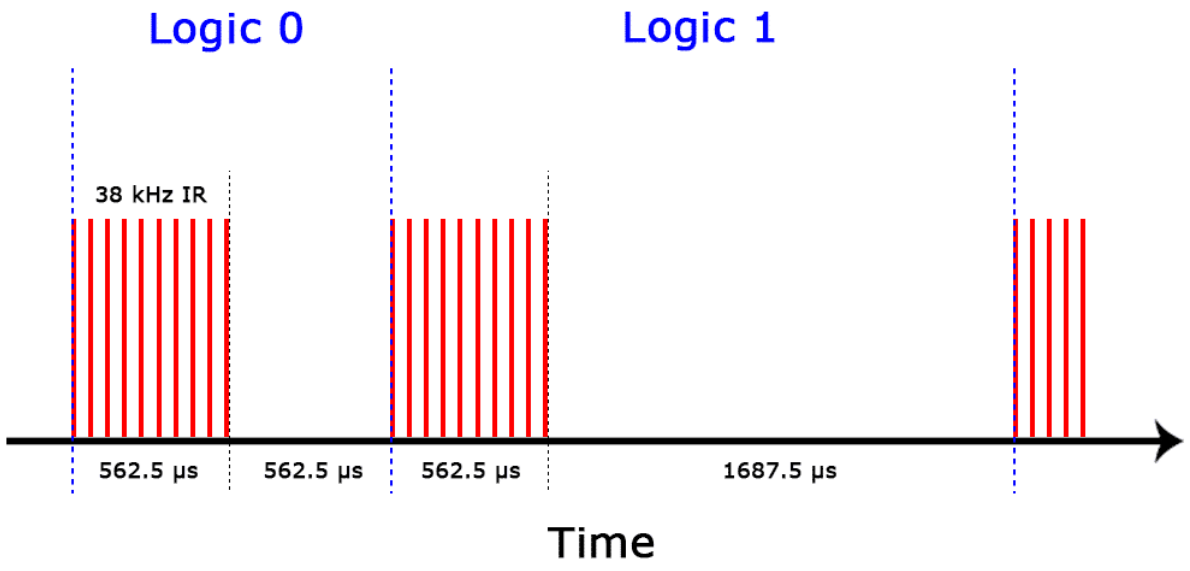
The receiver diode detects all frequencies of IR light, but it has a band-pass filter and only lets through IR at 38 kHz. It then amplifies the modulated signal with a pre-amplifier and converts it to a binary signal before sending it to a microcontroller.

8.IR TRANSMISSION PROTOCOLS

The pattern in which the modulated IR signal is converted to binary is defined by a transmission protocol. There are many IR transmission protocols. Sony, Matsushita, NEC, and RC5 are some of the more common protocols.

The NEC protocol is also the most common type in Arduino projects, so I'll use it as an example to show you how the receiver converts the modulated IR signal to a binary one.

Logical '1' starts with a 562.5 µs long HIGH pulse of 38 kHz IR followed by a 1,687.5 µs long LOW pulse. Logical '0' is transmitted with a 562.5 µs long HIGH pulse followed by a 562.5 µs long LOW pulse:



This is how the NEC protocol encodes and decodes the binary data into a modulated signal. Other protocols differ only in the duration of the individual HIGH and LOW pulses.

9.IR CODES

Each time we press a button on the remote control, a unique hexadecimal code is generated. This is the information that is modulated and sent over IR to the receiver. In order to decipher which key is pressed, the receiving microcontroller needs to know which code corresponds to each key on the remote.

Different remotes send different codes for the keypresses, so we'll need to determine the code generated for each key on your particular remote. If you can find the datasheet, the IR key codes should be listed.

10.PROGRAMMING THE IR RECEIVER

Once you have the receiver connected, we can install the Arduino library and start programming. In the examples below, I'll show you how to find the codes sent by your remote, how to find the IR protocol used by your remote, how to print key presses to the serial monitor or an LCD, and finally, how to control the Arduino's output pins with a remote.

11.INSTALL THE IRREMOTE LIBRARY

We'll be using the IRremote library for all of the code examples below. You can download a ZIP file of the library from here.

To install the library from the ZIP file, open up the Arduino IDE, then go to Sketch > Include Library > Add .ZIP Library, then select the IRremote ZIP file that you downloaded from the link above.

12.ARDUINO:

The Arduino Uno is a microcontroller board based on the ATmega328. It has 20 digital input/output pins (of which 6 can be used as PWM outputs and 6 can be used as analog inputs), a 16 MHz resonator, a USB connection, a power jack, an in-circuit system programming (ICSP) header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.

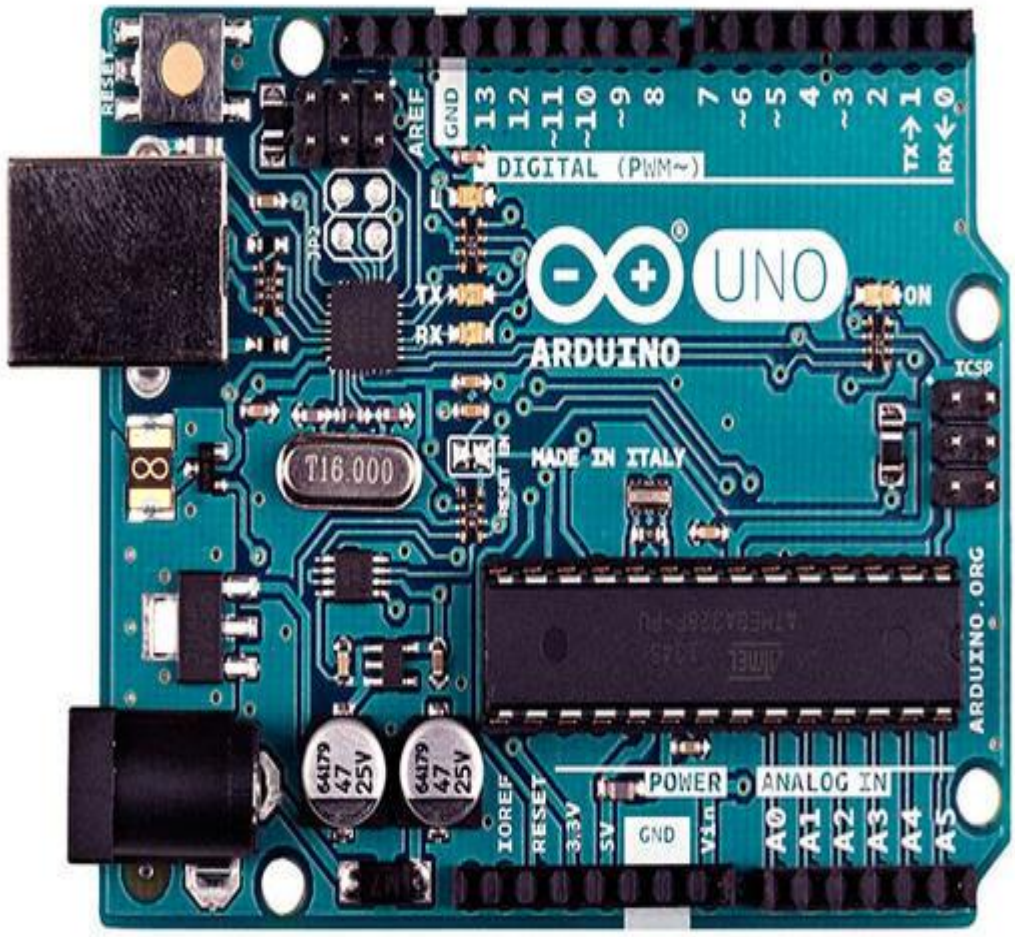
The Uno differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features an ATmega16U2 programmed as a USB-to-serial converter. This auxiliary microcontroller has its own USB bootloader, which allows advanced users to reprogram it.

The Arduino has a large support community and an extensive set of support libraries and hardware add-on "shields" (e.g. you can easily make your Arduino wireless with our Wixel shield), making it a great introductory platform for embedded electronics.

The USB controller chip changed from ATmega8U2 (8K flash) to ATmega16U2 (16K flash). This does not increase the flash or RAM available to sketches.

Three new pins were added, all of which are duplicates of previous pins. The I2C pins (A4, A5) have been also been brought out on the side of the board near AREF. There is a IOREF pin next to the reset pin, which is a duplicate of the 5V pin.

The reset button is now next to the USB connector, making it more accessible when a shield is used.



13.CODE:(note: code directly from arduino ide is submitted separately(in .ino form)

```
/*libraries for ir remote,ir sensor&lcd*/
#include<IRremote.h>
#include<LiquidCrystal.h>
LiquidCrystal lcd(13,12,9,8,6,5);//creating the object and assigning the digital pin number of arduino to lcd
int RECV_PIN = 11;//the pin comes out of ir sensor is connected to 11 th digital pin of arduino
int buzzer = 3;//buzzer is connected to 3rd digital pin of arduino
int led1 = 2;// led 1 is connected to 2nd digital pin of arduino
int led2 = 4;//led2 is connected to 4th digital pin of arduino
int led3 = 7;//led 3 is connected to 7th digital pin of arduino
int motor= 10;//motor is connected to 10th digital pin of arduino
int itsONled[]={0,0,0,0,0,0};//which sets the initial value
/*the initial state of LEDs is OFF*/
#define code1 16582903 //code received from button number 1
#define code2 16615543 //code received from button number 2
#define code3 16599223 //code received from button number 3
#define code4 16591063 //code received from button number 4
IRrecv irrecv(RECV_PIN);//IRrecv is the pre defined function from library, irrecv is the object
decode_results results;//decode is the fuction and results is the object

void setup()//initial setup
{
  Serial.begin(9600);//to get the value serially with baud (data transferring)rate.
  lcd.begin(16,2);//which initiallizes the lcd as 16*2(16-coloumns,2-rows)
  digitalWrite(buzzer, LOW);//which sets the initial value of buzzer pin =0
  irrecv.enableIRIn();//which enables to get value from ir remote to sensor
  pinMode(led1, OUTPUT);//directing led1 as output
  pinMode(led2, OUTPUT);//directing led2 as output
  pinMode(led3, OUTPUT);//directing led3 as output
  pinMode(motor, OUTPUT);//directing motor as output
```

```
  pinMode(buzzer, OUTPUT);//directing buzzer as output
}

void loop()//main part of code
{
  if (irrecv.decode(&results))//the signal received from remote will collected in ir sensor .which will be decoded
  {
    unsigned int value = results.value;//value is a variable of type unsigned int which stores the result from results.vale
    switch(value)//starting of swith case
    {
      case code1://which tells it is the first case
      if(itsONled[1]==1)                  //if first led1 is on then
      {                                   //turn it off when button is pressed
        digitalWrite(led1, LOW);          //and set its state as off
        itsONled[1]=0;                    //set cursor at (0,0) position and print( "    ")
        lcd.setCursor(0,0);
        lcd.print("    ");                //which basically clears the screen
      }
      else                                //else if first led is off
      {
        digitalWrite(led1, HIGH);         //turn it on when button is pressed
        itsONled[1]=1;                    //and set its state as on
        lcd.setCursor(0,0);               //set position of cursor at (0,0)
        lcd.print("LED1");                //  and print what is given
      }
      break;                              //which break the  body of if comes out of  switch
      case code2:                         //it is the 2nd case
      if(itsONled[2]==1)//if 2nd led is on  then turn it off
      {
        digitalWrite(led2, LOW);//when the switch is pressed
```

```
 itsONled[2]=0;//and set its state as off
 lcd.setCursor(4,0);//set cursor at (4,0) and print(____)
 lcd.print("    ");//which mainly clear screen
}
else
{
 digitalWrite(led2, HIGH);//else if 2nd led is off turn it on
 itsONled[2]=1;//when button is pressed and set its state as on

 lcd.setCursor(4,0);//set cursor at (0,0)
 lcd.print("LED2");//print as led2
}
break;//which break the body of if comes out of switch
case code3://it is the 3rd case
if(itsONled[3]==1)//if 3rd led is on then turn it off when switch is pressed
{
 digitalWrite(led3, LOW);
 itsONled[3]=0;
 lcd.setCursor(8,0);//set the cursor at(8,0)
 lcd.print("    ");//clear screen
}
else
{
 digitalWrite(led3, HIGH);//else if 3rd led is off turn it on when switch is pressed
 itsONled[3]=1;
 lcd.setCursor(8,0);//set cursor at (8,0)
 lcd.print("LED3");//print as led 3
}
break;//which comes out of if and switch condition
case code4:
if(itsONled[4]==1)//if motor is on set its state as off when button is pressed
```

```
    {
      digitalWrite(motor, LOW);//and set its state as off
      itsONled[4]=0;
      lcd.setCursor(13,0);// set the position of cursor at(13,0)
      lcd.print("   ");//clear screen
    }
    else//else if motor is off set its state as on
    {
      digitalWrite(motor, HIGH);//if button is pressed
      itsONled[4]=1;//set its state as high
      lcd.setCursor(13,0);//set cursor at (13,0)
      lcd.print("FAN");//print fan
    }
    break;//which comes out of if and switch
   }
  if((digitalRead(2)&&digitalRead(4)&&digitalRead(7)&&digitalRead(10))==1)//if          led1,led2,led3
&fans are ON
  {//tone(buzzer,750);//this code is not work because IRremote.h library is used which will miss lead
    digitalWrite(buzzer,750);//which on the buzzer with delay of 500
    //delay(500);
    lcd.setCursor(1,1);//set cursor at(1,1)
    lcd.print("HEAVY POWER");//print the message

  }
  else//if any one of the led or fan is off then
  {//noTone(buzzer);//code not work because IR remote.h library is used which will miss lead
    digitalWrite(buzzer, LOW);//which sets the buzzer off
    //delay(500);
    lcd.setCursor(1,1);// which set cursor at(1,1)
    lcd.print("           ");//which clear the screen
```

```
  }
   Serial.println("VALUE OBTAINED FROM TEMOTE",value);
   irrecv.resume();//which receive the next value
 }
}
```

14.Library used and links

IRremote Library:

<IRremote.h>

IRremote, by Ken Shirriff, allows you to receive or transmit Infrared Remote Control codes. You can make your projects controlled by a remote, or make them control other devices like televisions and stereo components.

Basic Usage

IRremote acts like 2 libraries, one for sending and one for receiving. Usually it's easiest to find the codes to transmit by first using the receiver.

Receiving:

IRrecv irrecv(receivePin)

Create the receiver object, using a name of your choice.
irrecv.enableIRIn()

Begin the receiving process. This will enable the timer interrupt which consumes a small amount of CPU every 50 µs.
irrecv.decode(&results)

Attempt to receive a IR code. Returns true if a code was received, or false if nothing received yet. When a code is received, information is stored into "results".
results.decode_type: Will be one of the following: NEC, SONY, RC5, RC6, or UNKNOWN.
results.value: The actual IR code (0 if type is UNKNOWN)
results.bits: The number of bits used by this code
results.rawbuf: An array of IR pulse times
results.rawlen: The number of items stored in the array
irrecv.resume()

After receiving, this must be called to reset the receiver and prepare it to receive another code.
irrecv.blink13(true)

Enable blinking the LED when during reception. Because you can't see infrared light, blinking the LED can be useful while troubleshooting, or just to give visual feedback.

Transmitting:IRsend irsend;

Create the transmit object. A fixed pin number is always used, depending on which timer the library is utilizing.
irsend.sendNEC(IRcode, numBits);

Send a code in NEC format.
irsend.sendSony(IRcode, numBits);

Send a code in Sony format.

irsend.sendRC5(IRcode, numBits);

Send a code in RC5 format.
irsend.sendRC6(IRcode, numBits);

Send a code in RC6
irsend.sendRaw(rawbuf, rawlen, frequency);

LINKS TO DOWNLOAD IRremote.h:

https://github.com/z3t0/Arduino-IRremote/zipball/master

http://downloads.arduino.cc/libraries/github.com/z3t0/IRremote-2.5.0.zip

LiquidCrystal library:

<LiquidCrystal.h>

This library allows an Arduino board to control LiquidCrystal displays (LCDs) based on the Hitachi HD44780 (or a compatible) chipset, which is found on most text-based LCDs. The library works with in either 4- or 8-bit mode (i.e. using 4 or 8 data lines in addition to the rs, enable, and, optionally, the rw control lines).

To use this library

#include <LiquidCrystal.h>

- Autoscroll: Shift text right and left.
- Blink: Control of the block-style cursor.
- Cursor: Control of the underscore-style cursor.
- Display: Quickly blank the display without losing what's on it.
- Hello World: Displays "hello world!" and the seconds since reset.
- Scroll: Scroll text left and right.
- Serial Display: Accepts serial input, displays it.
- Set Cursor: Set the cursor position.
- Text Direction: Control which way text flows from the curso

Function

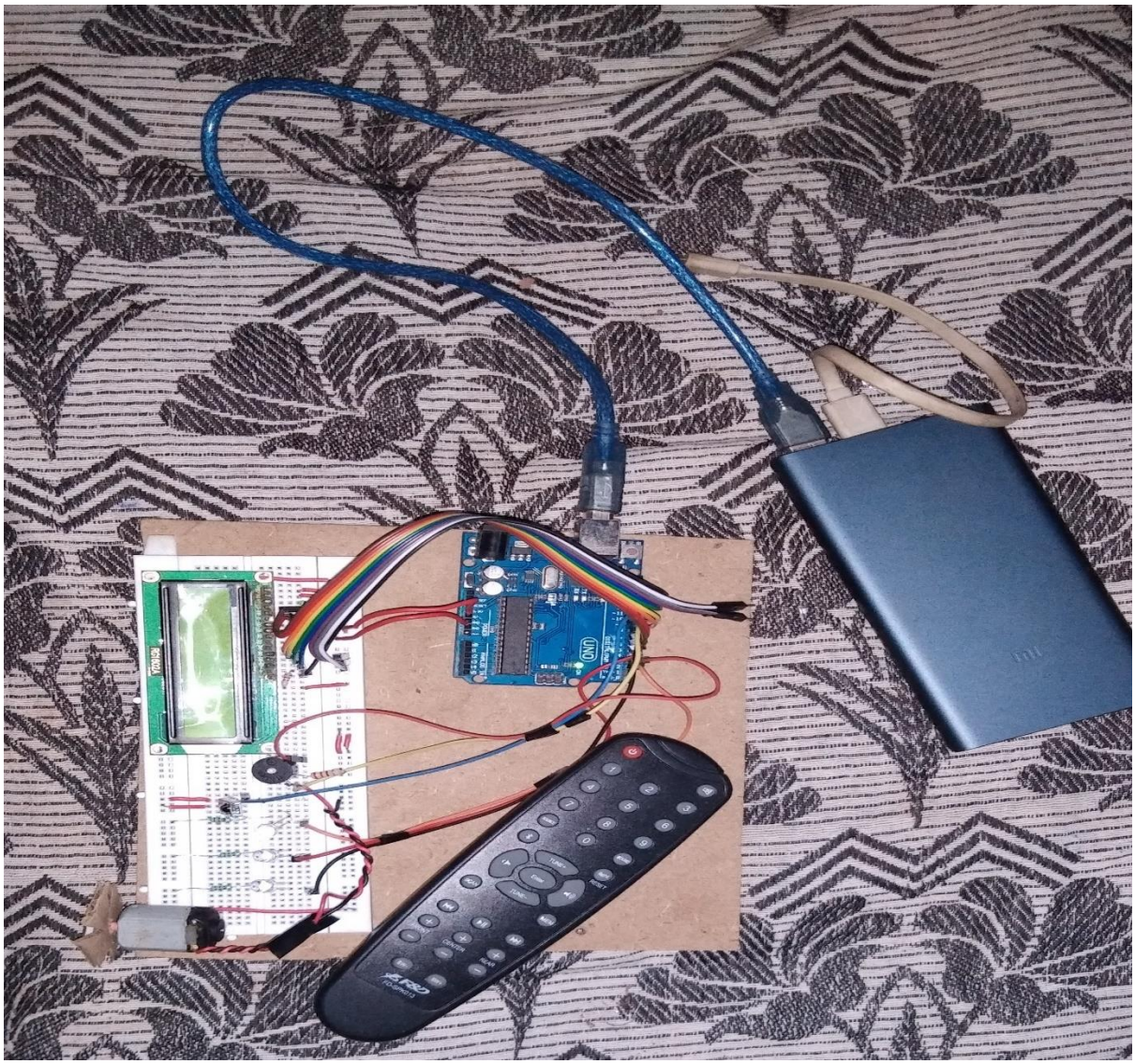
- LiquidCrystal()
- begin()
- clear()
- home()
- setCursor()
- write()
- print()
- cursor()
- noCursor()

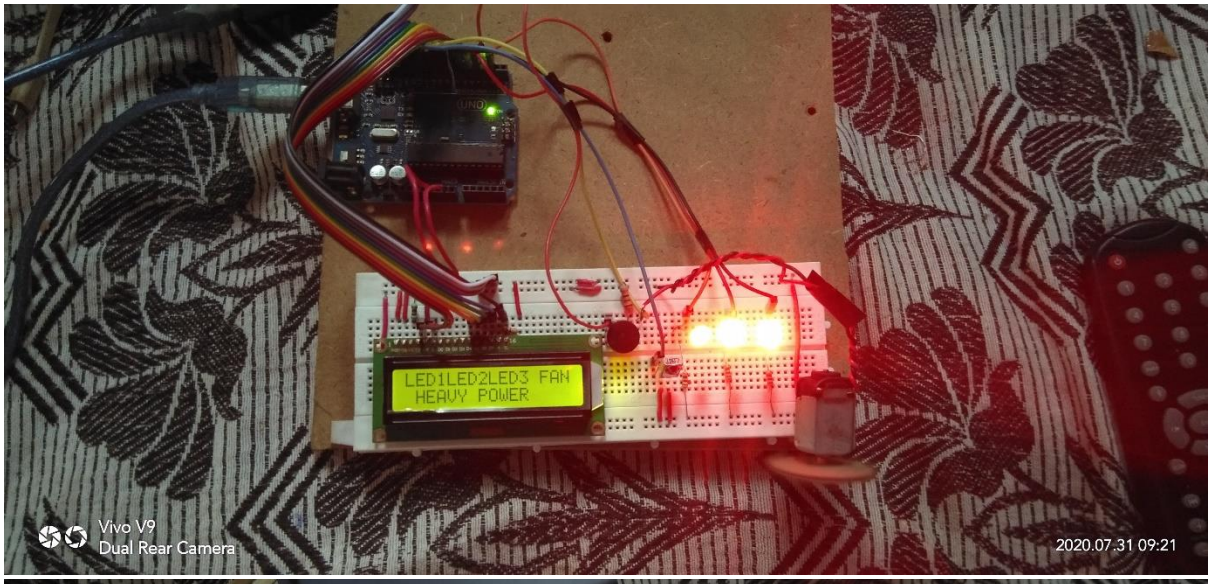
- blink()
- noBlink()
- display()
- noDisplay()
- scrollDisplayLeft()
- scrollDisplayRight()
- autoscroll()
- noAutoscroll()
- leftToRight()
- rightToLeft()
- createChar()

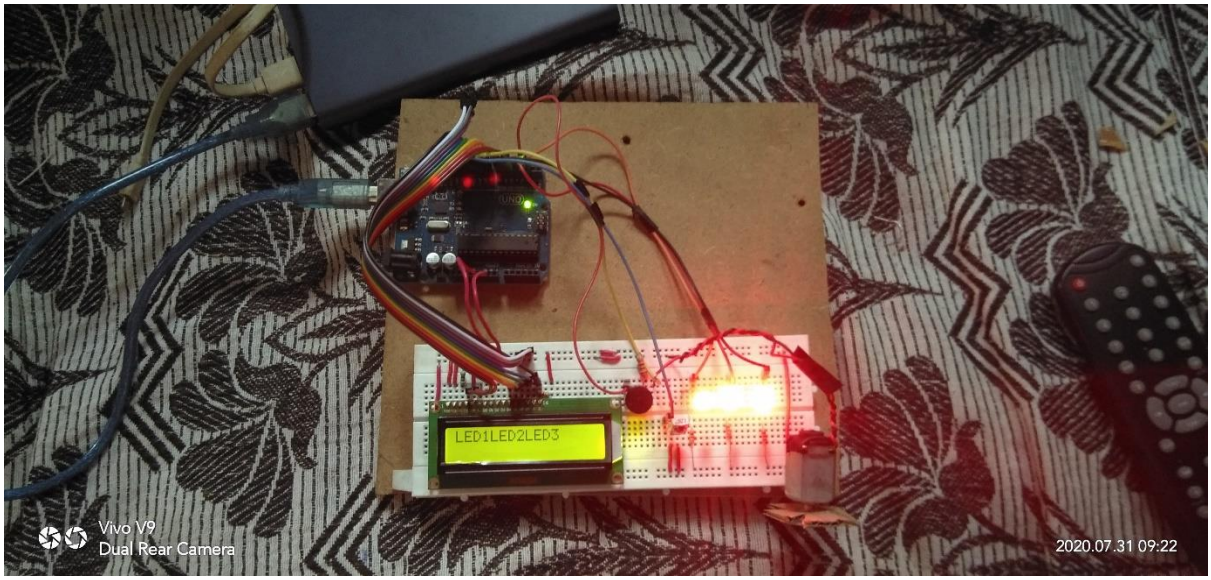
Link to download LiquidCrystal.h:

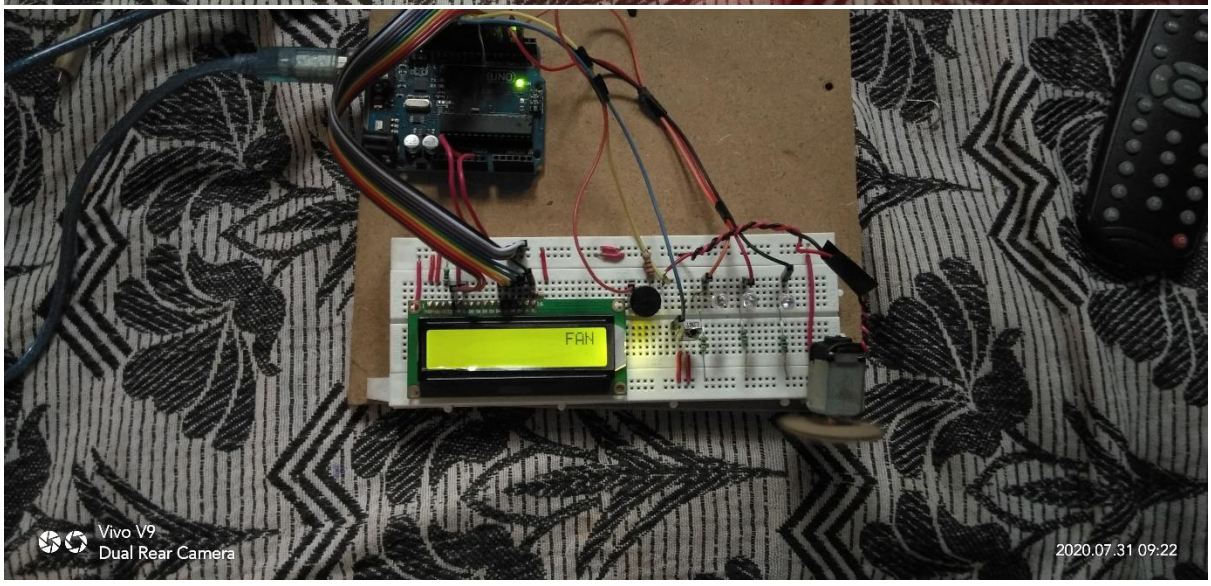
http://downloads.arduino.cc/libraries/github.com/arduino-libraries/LiquidCrystal-1.0.7.zip

SOME IMAGES OF THE PROJECT:









Tinker cad link : https://www.tinkercad.com/things/3cF9qso50cU-ir-remote-with-led