GR20 Regulations

III B.Tech I Semester

Micro Controller and Internet of Things Lab

(GR20A3054)

Department of Computer Science and Engineering

GOKARAJU RANGARAJU
INSTITUTE OF ENGINEERING AND TECHNOLOGY
(Autonomous)

SYLLABUS

Gokaraju Rangaraju Institute of Engineering and Technology

Micro Controller and Internet of Things Lab

Course code: GR20A3054 **L:3 T:0 P:0 C:3**

Task 1

Write an ARDUINO Program for Smart street light management system.

Task 2

Write an ARDUINO Program for Smart reserve break monitoring system.

Task 3

Write an ARDUINO Program for Smart soil state monitoring system.

Task 4

Write an ARDUINO Program for Micro controller-Activator Interface.

a)DC Motor

b)Relay

Task 5

Write an ARDUINO Program for Moving curtail display using LCD.

Task 6

Write an ARDUINO Program for Sensor /Activator Interfacing using ESP32.

Task 7

a) Write an ARDUINO Program to Develop a Mobile App for simple interface.

b)Design a Mobile App to work with data of a data.

Task 8

Internet enabled remote range indicator.

Task 9

Internet enabled smart room lighting system.

Task 10

Internet enabled Smart Garden Maintainence.

Task 11

Internet enabled Display of Ambient Parameter.

Task 12

Internet Enabled Home Safety and security system.

Teaching Methodologies:

1. Power Point Presentation

2. White Board

TEXT BOOKS:

1) Embedded Controllers using C and Arduino/2E by Jmes M.Fiore

2) Web References:

- https://www.arduino.cc/en/Tutorial/HomePage
- https://www.w3schools.com/python/andhttps://pythonprogramming.net/introductionraspberry-pi-tutorials/

Course Objectives and Course Outcomes

Course Objectives:

The objective of this course is to

1. Understand the principles of LED, LCD and DC Motor.
2. Understand the working principles of sensors.
3. Explore communication devices and interfacing.
4. Understand the installation of Arduino IDE and Raspberry Pi.
5. Understand IoT Principles.

Course Outcomes:

At the end of this course student will be able to

1. Develop the programs for controlling DC Motor using Arduino/ Raspberry Pi.
2. Develop the programs on Arduino using sensors.
3. Develop the programs using communication devices.
4. Implement IoT programs using Raspberry Pi.
5. Explore IoT Projects using Arduino/Raspberry Pi.

INDEX

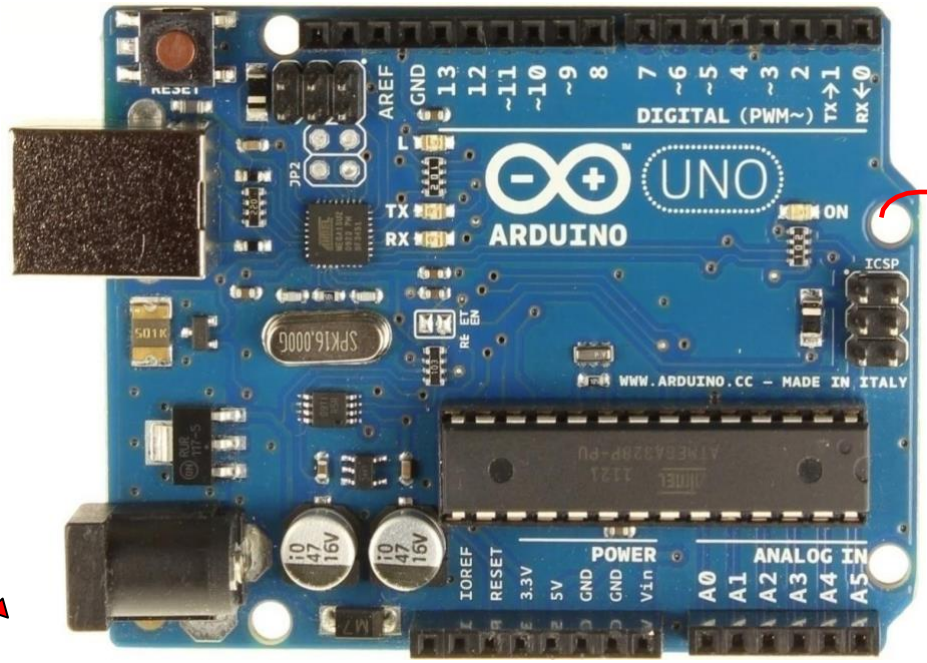
Introduction to Arduino

The Arduino Uno is a microcontroller board based on the ATmega328. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an 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.

Digital pins (0 to 13)

Pin 0 – RX
Pin 1 - TX

USB Connector



Power

Power supplies

Analog pins (A0 to A5)

Features of Uno board

Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 Ma
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328) of which 0.5 KB used by boot loader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Clock Speed	16 MHz

Power

- The Arduino Uno can be powered via the USB connection or with an external power supply. The power source is selected automatically.
- External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector.\
- The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable.

If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

The power pins are as follows:

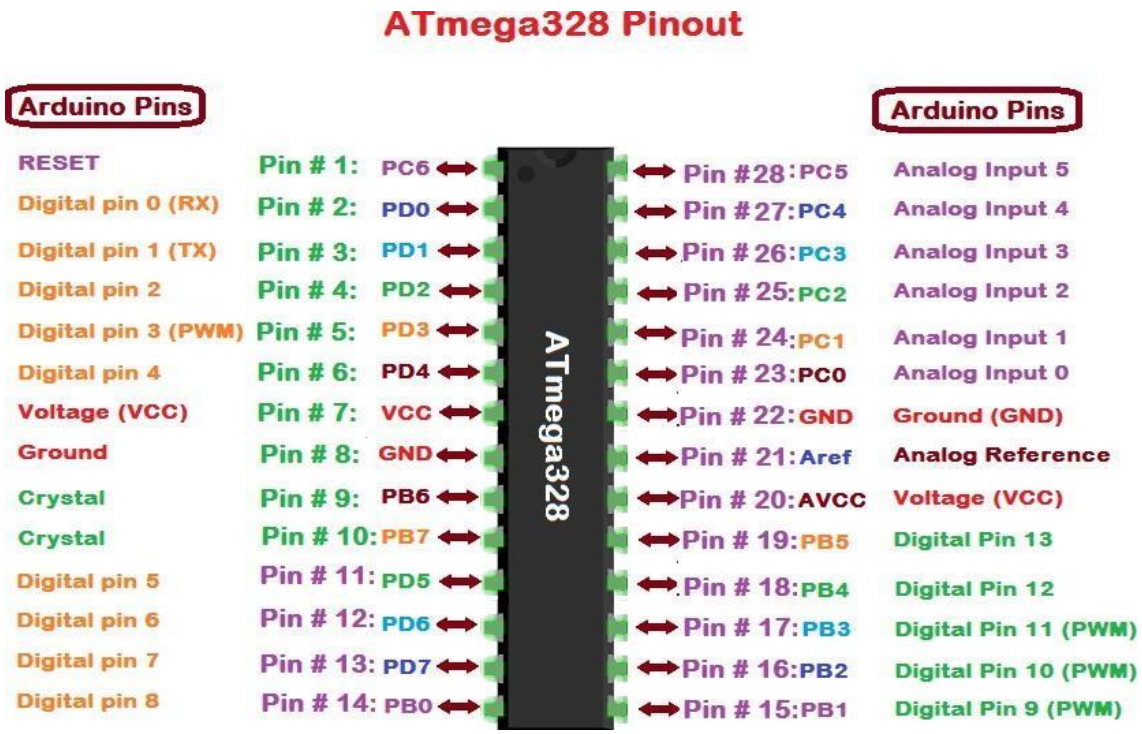
- **VIN.** The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.
- **5V.**This pin outputs a regulated 5V from the regulator on the board. The board can be supplied with power either from the DC power jack (7 - 12V), the USB connector (5V), or the VIN pin of the board (7-12V).
- **3V3.** A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.
- **GND.** Ground pins.

Each of the 14 digital pins (pins 0 to 13) on the Uno can be used as an input or output, using pinMode(), digitalWrite(), and digitalRead() functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) . In addition, some pins have specialized functions:

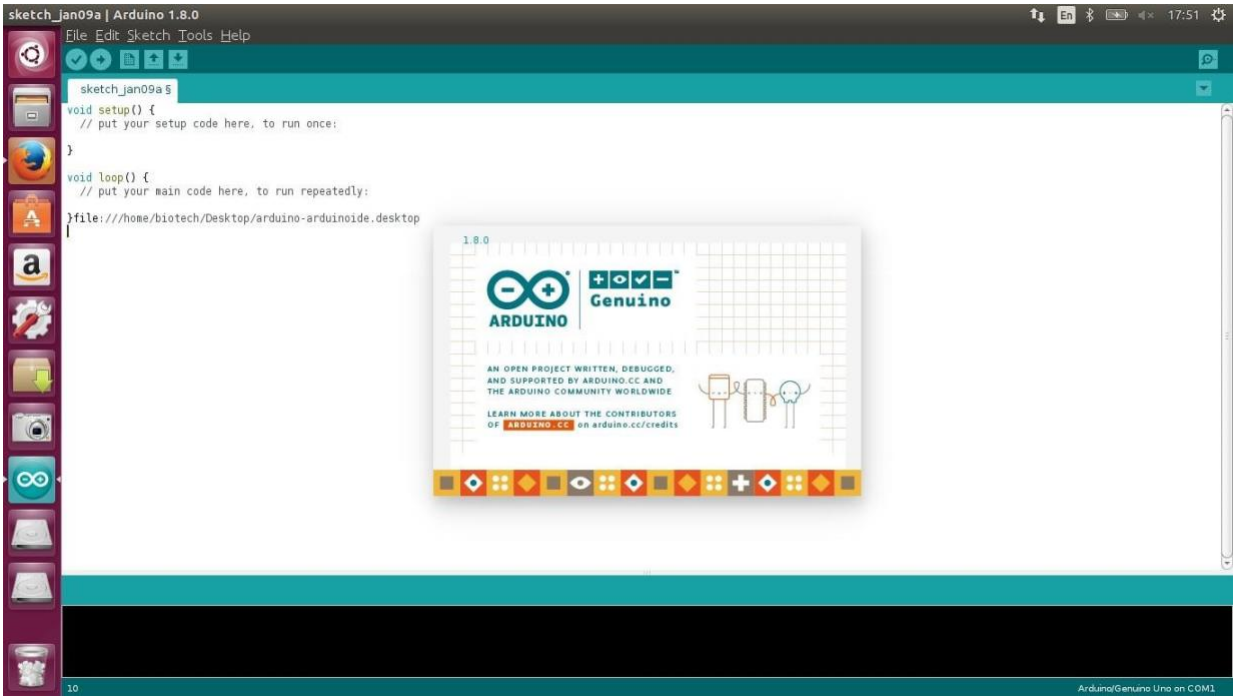
- **Serial: 0 (RX) and 1 (TX).** Used to receive (RX) and transmit (TX) TTL serial data. These pins are connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip.
- **External Interrupts: 2 and 3.** These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value.
- **PWM: 3, 5, 6, 9, 10, and 11.** Provide 8-bit PWM output with the analogWrite() function.
- **SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK).** These pins support SPI communication using the SPI library.
- **LED: 13.** There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.

The Uno has 6 analog inputs, labeled A0 through A5, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though it is possible to change the upper end of their range using the AREF pin and the analogReference() function.

ATmega328-Arduino Pin Mapping



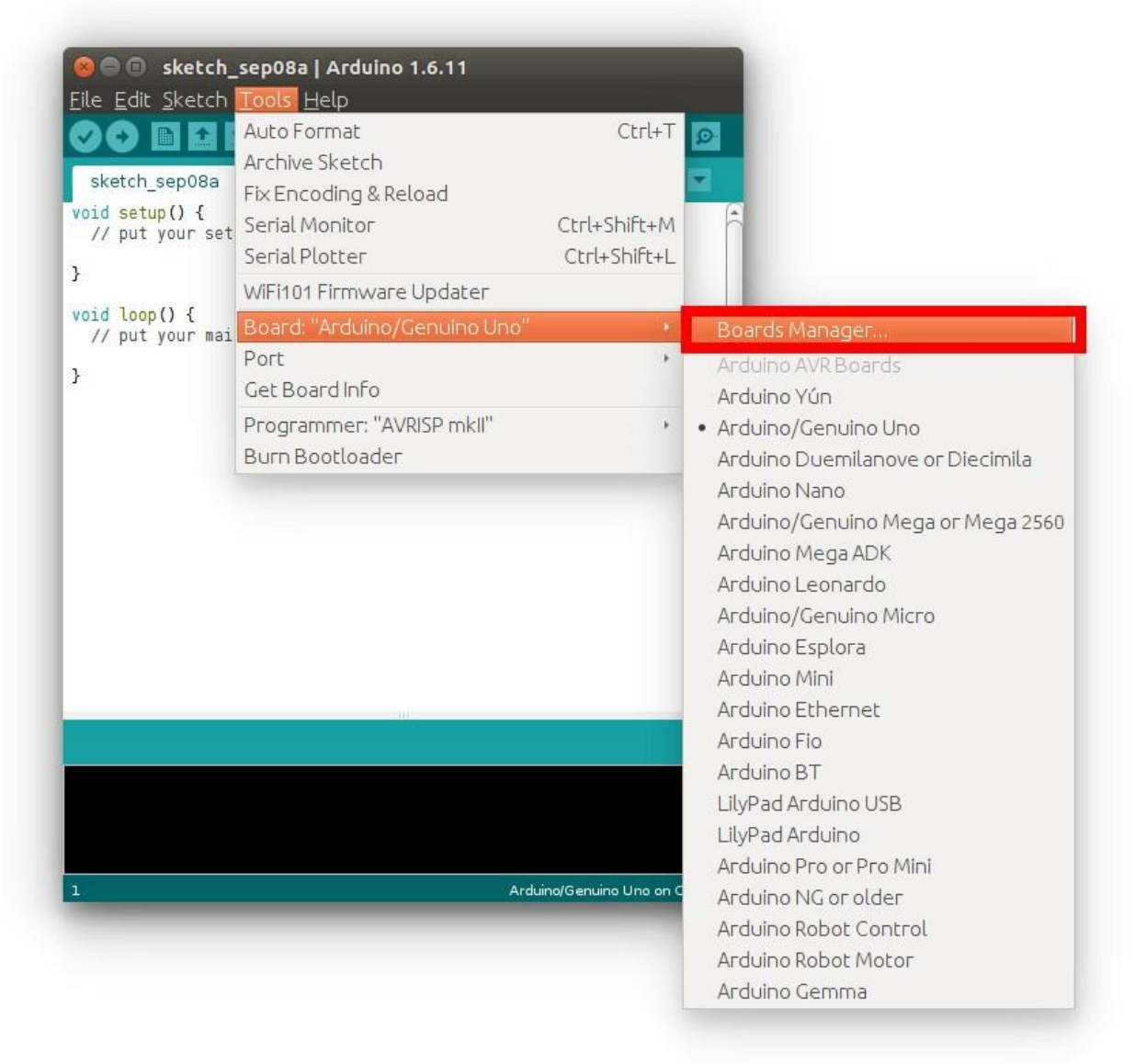
Arduino IDE



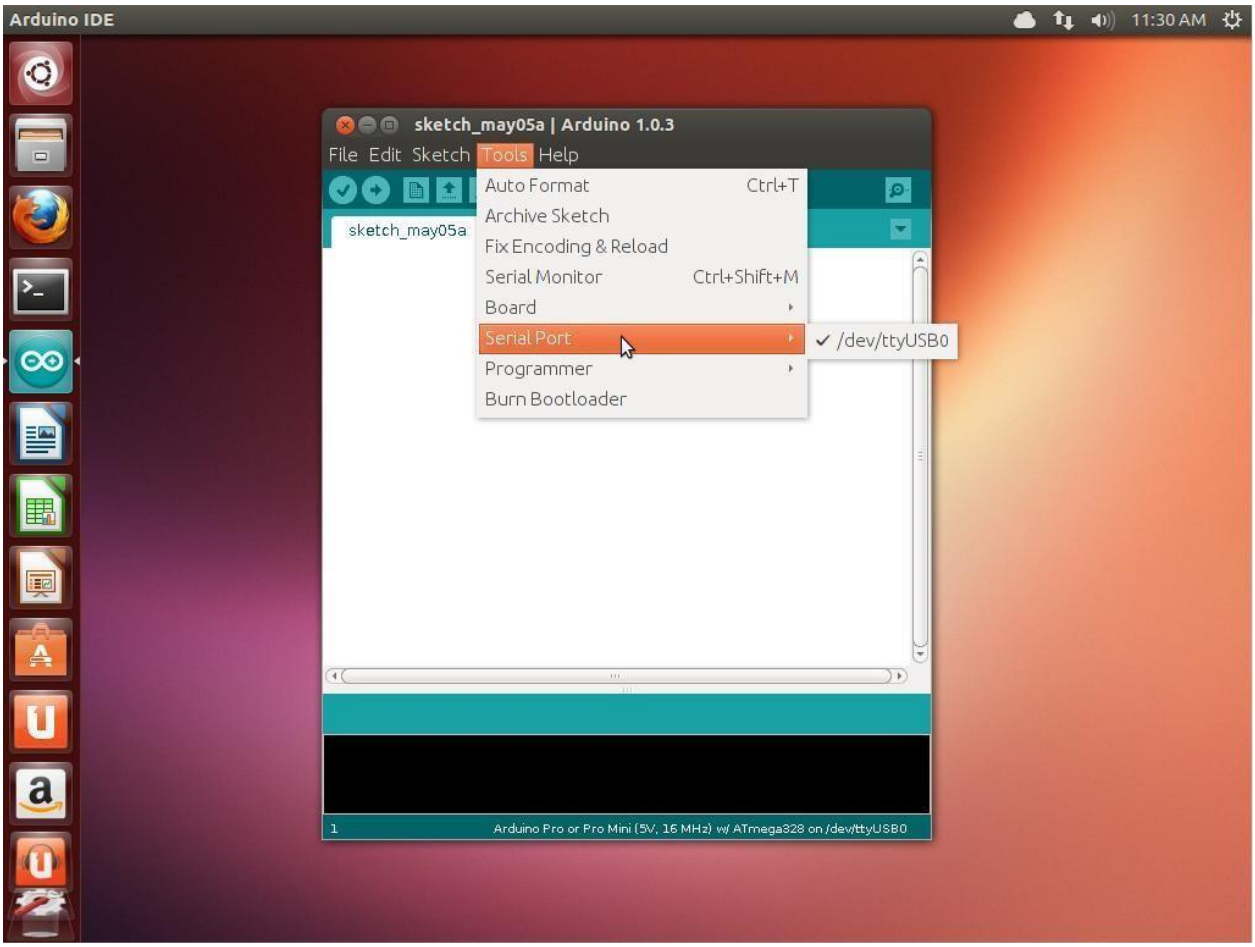
The programs written for Arduino are called sketches. For the sketch to work on the Arduino Uno, there are two hardware related settings need to be done in the Arduino IDE

- Board
- SerialPort

For selecting the board, go to the Tools tab and select Board. From the menu select Uno.



- When you connect your Arduino Uno to the USB port of your laptop, it will be mapped as a serial port.
- In the Arduino IDE, select the Serial Port as the port to which the Arduino is mapped.



- The basic structure of the Arduino sketch is fairly simple and has two required functions:
- void setup()

  {

  Statements;

  }

  void loop()

  {

  Statements;

  }
- **setup()** is the preparation, **loop()** is the execution. Both functions are required for the program to work.
- The setup function should follow the declaration of any variables at the very beginning

of the program.

- It is the first function to run in the program, is run only once, and is used to set pin Mode or initialize serial communication.
- The loop function follows next and includes the code to be executed continuously – reading inputs, triggering outputs, etc. This function is the core of all Arduino programs and does the bulk of the work.
- **setup():** The setup() function is called once when your program starts. Use it to initialize pin modes, or begin serial. It must be included in a program even if there are no statements to run.
- void setup()

  {

  pinMode(pin, OUTPUT); // sets the 'pin' as output

  }
- **loop():** After calling the setup() function, the loop() function does precisely what its name suggests, and loops consecutively, allowing the program to change, respond, and control the Arduino board.
- void loop()

  {

  digitalWrite(pin, HIGH); // turns 'pin' on delay(1000); // pauses for one second digitalWrite(pin, LOW); // turns 'pin' off delay(1000); // pauses for one second

  }
- **pinMode(pin, mode):** Used in void setup() to configure a specified pin to behave either as an INPUT or an OUTPUT.
- pinMode(pin, OUTPUT); // sets ‘pin’ to output
- pinMode(pin, INPUT); // set ‘pin’ to input
- Pins configured as OUTPUT can provide 40 mA (milliamps) of current to other devices/circuits. This is enough current to brightly light up an LED (don't forget the

series resistor), but not enough current to run most relays, solenoids, or motors.

- Short circuits on Arduino pins and excessive current can damage or destroy the output pin, or damage the entire Atmega chip. It is often a good idea to connect an OUTPUT pin to an external device in series with a 470Ω or 1KΩ resistor.
- **digitalRead(pin):** Reads the value from a specified digital pin with the result either HIGH or LOW. The pin can be specified as either a variable or constant (0-13).
- value = digitalRead(Pin); // sets 'value' equal to the input pin.
- **digitalWrite(pin, value):** Outputs either logic level HIGH or LOW at (turns on or off) a specified digital pin. The pin can be specified as either a variable or constant (0-13).
- digitalWrite(pin, HIGH); // sets 'pin' to high.
- **analogRead(pin):** Reads the value from a specified analog pin with a 10-bit resolution. This function only works on the analog in pins (0-5). The resulting integer values range from 0 to 1023.
- value = analogRead(pin); // sets 'value' equal to 'pin'.
- **analogWrite(pin, value):** Writes a pseudo-analog value using hardware enabled pulse width modulation (PWM) to an output pin marked PWM. On Uno, this function works on pins 3, 5, 6, 9, 10, and 11. The value can be specified as a variable or constant with a value from 0-255.
- analogWrite(pin, value); // writes 'value' to analog 'pin'.
- **delay(ms):** Pauses a program for the amount of time as specified in milliseconds, where 1000 equals 1 second.
- delay(1000); // waits for one second.
- **Serial.begin(baud rate):** Opens serial port and sets the baud rate for serial data transmission. The typical baud rate for communicating with the computer is 9600 although other speeds are supported.
- void setup()

  {

Serial.begin(9600); // opens serial port

} // sets data rate to 9600 bps.

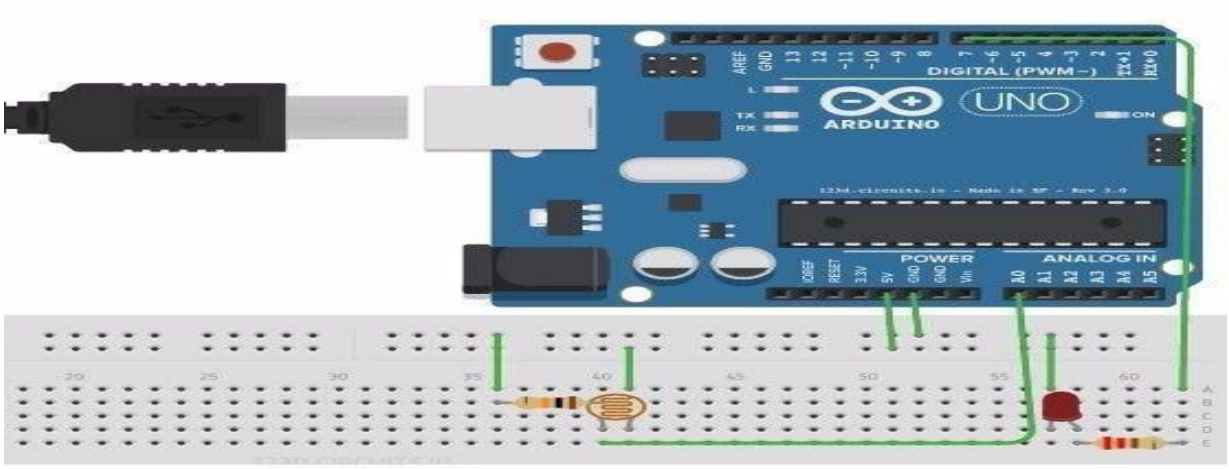
- **Serial.println(data):** Prints data to the serial port, followed by an automatic carriage return and line feed. This command takes the same form as Serial.print(), but is easier for reading data on the Serial Monitor.
- Serial.println(analogValue); // sends the analogValue'

TASK 1

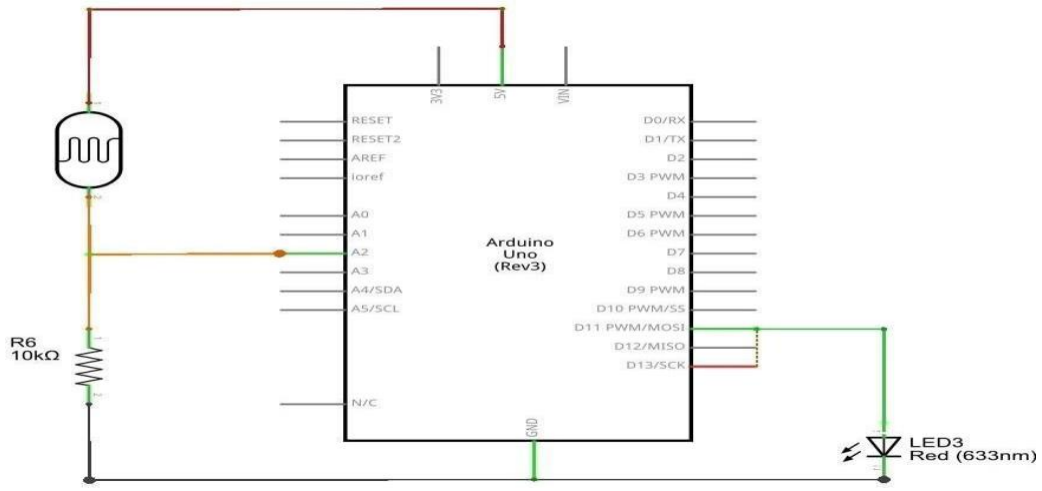
TASK 1 Aim: Write an ARDUINO Program for Smart street light management system.

Components Required: Arduino UNO, LED, LDR, connecting wires, bread board, Computer system with Arduino IDE.

Circuit:



Schematic Diagram:

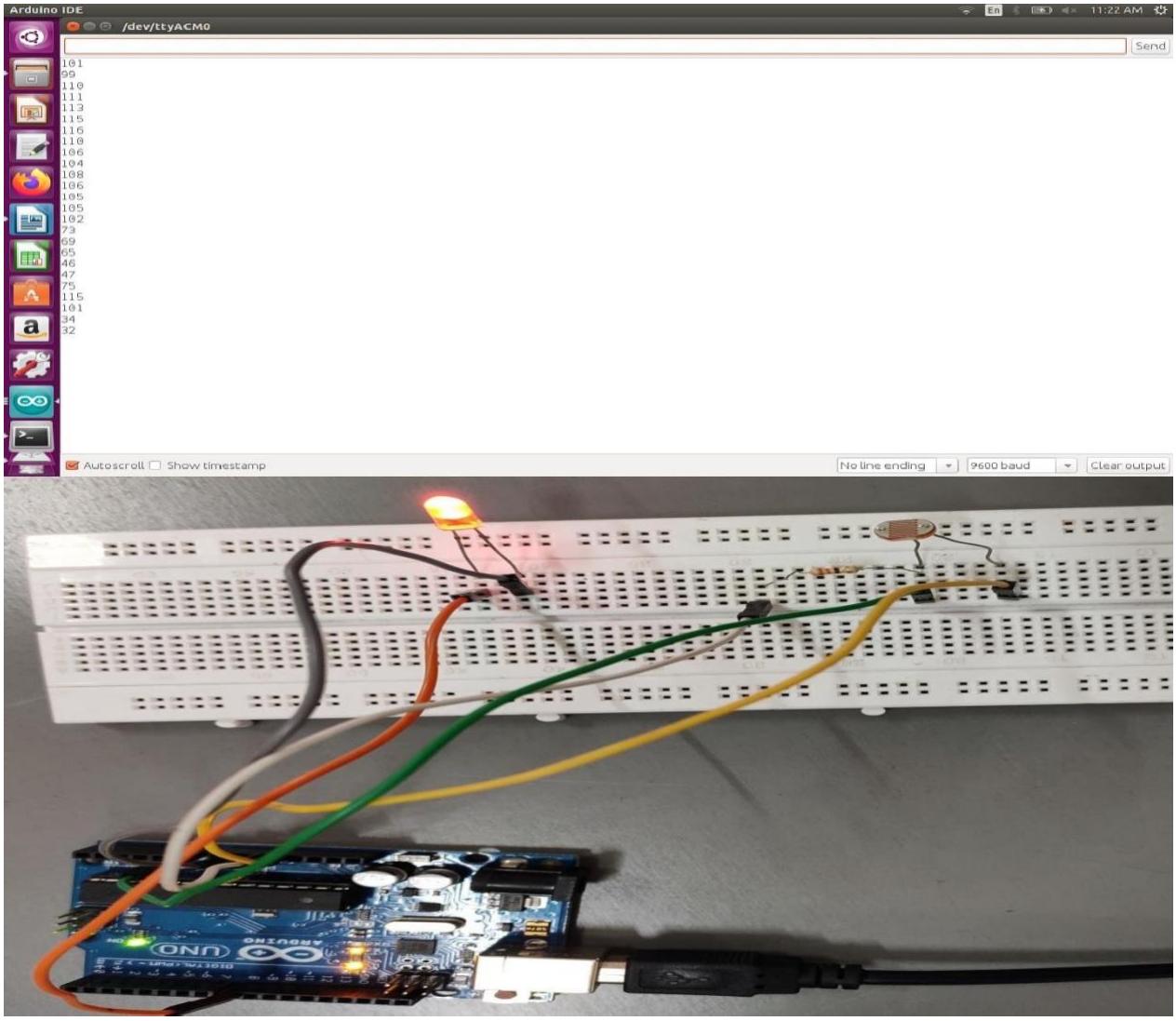


LDR is Light Dependent Resistor. LDRs are made from semiconductor materials to enable them to have their light-sensitive properties. These LDRs or PHOTO RESISTORS works on the principle of “Photo Conductivity”. Now what this principle says is, whenever light falls on the surface of the LDR (in this case) the conductance of the element increases or in other words, the resistance of the LDR falls when the light falls on the surface of the LDR. This property of the decrease in resistance for the LDR is achieved because it is a property of semiconductor material used on the surface.

Program:

```
void setup() {
pinMode(A0,INPUT);
Serial.begin(9600);
pinMode(8,OUTPUT);
}
void loop() {
// Serial.print("LDR = ");
 int i=analogRead(A0);
 Serial.println(i);
if(i>60)
digitalWrite(8,1);
else
digitalWrite(8,0);
delay(500);
}
```

Output:


Arduino IDE
11:22 AM
/dev/ttyACM0
Send
101
99
110
111
113
115
116
110
106
104
108
106
105
105
102
73
69
65
46
47
75
115
101
34
32
Autoscroll
Show timestamp
No line ending
9600 baud
Clear output

TASK 2

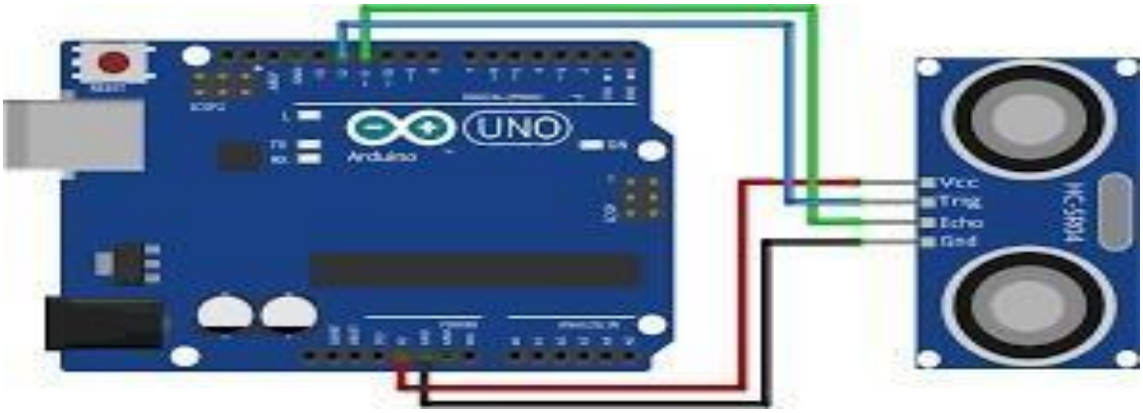
Ultrasonic Sensor

Task 2: Write an ARDUINO Program for Smart reserve break monitoring system.

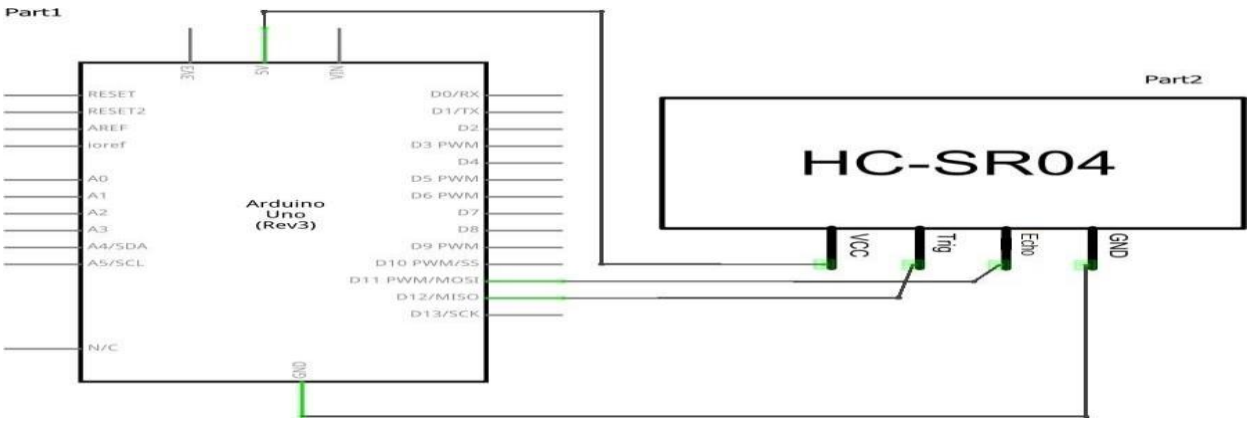
Aim: To Write an Arduino program for measuring distance using ultrasonic sensor

Components Required: Arduino UNO, Ultrasonic sensor, connecting wires, bread board, Computer system with Arduino IDE.

Circuit:



Schematic Diagram:



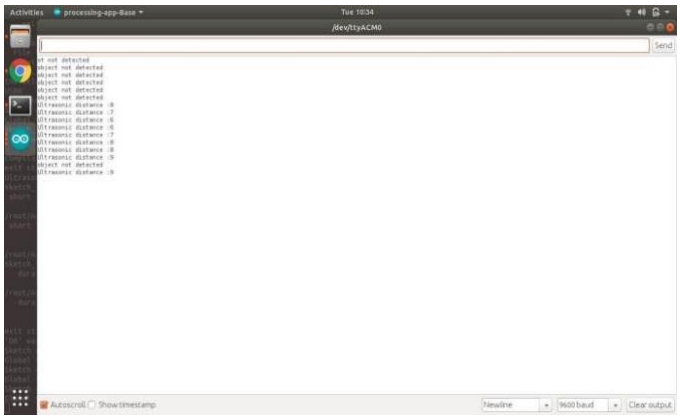
The HC-SR04 ultrasonic sensor uses SONAR to determine the distance of an object just like the bats do. It offers excellent non-contact range detection with high accuracy and stable readings in an easy-to-use package from 2 cm to 400 cm or 1” to 13 feet.

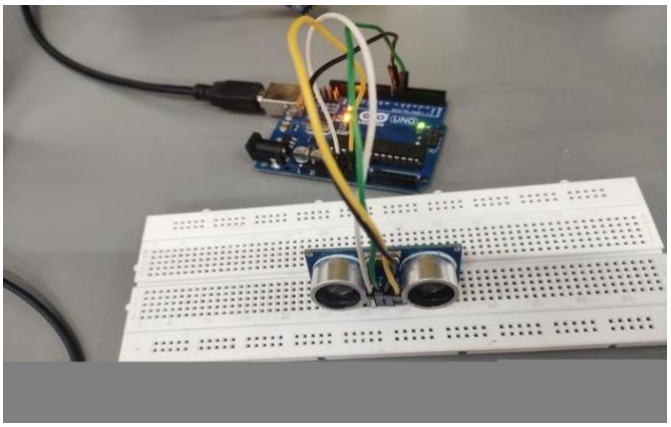
The operation is not affected by sunlight or black material, although acoustically, soft materials like cloth can be difficult to detect. It comes complete with ultrasonic transmitter and receiver module.

Program:

```
#include <Ultrasonic.h>
Ultrasonic ultrasonic(11,12);//Init an Ultrasonic object
int Distance;
void setup() {
Serial.begin(9600);
}
void loop()
{
 Distance=ultrasonic.Ranging(CM);//get the current result;
 delay(100);
 Serial.print("the distance is ");
 Serial.println(Distance);
 delay(1000);
}
```

Output





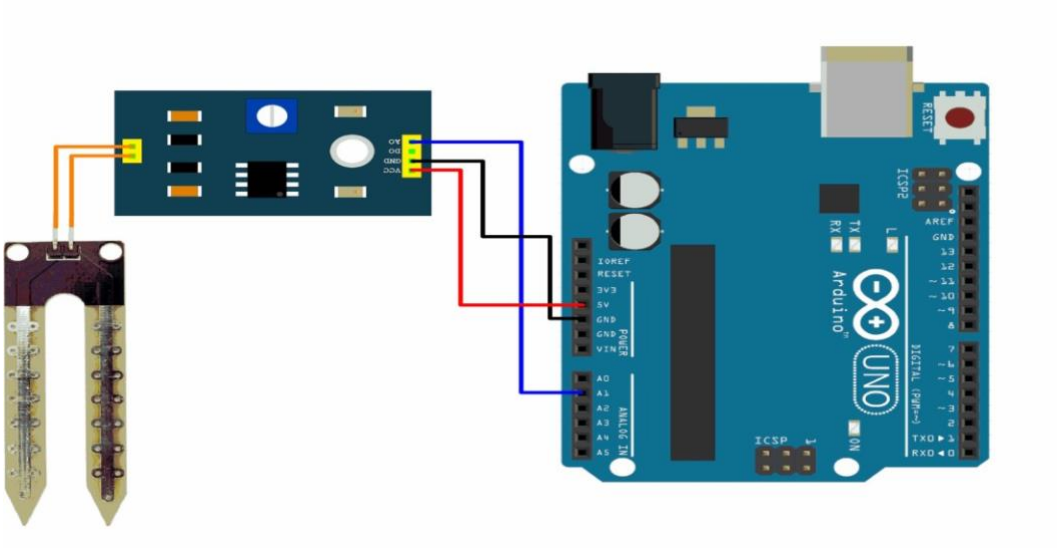
TASK 3

Soil Moisture Sensor

TASK 3: Write an ARDUINO Program for Smart soil state monitoring system.
Aim: To write an Arduino program for measuring wetness of soil using soil moisture sensor

Components Required: Arduino UNO, soil moisture sensor, connecting wires, bread board, Computer system with Arduino IDE.

Circuit:



Soil moisture is basically the content of water present in the soil. This can be measured using a soil moisture sensor which consists of two conducting probes that act as a probe. It can measure the moisture content in the soil based on the change in resistance between the two conducting plates.

The resistance between the two conducting plates varies in an inverse manner with the amount of moisture present in the soil.

Here, the analog output of soil moisture sensor is processed using ADC. The moisture content in terms of percentage is displayed on the serial monitor.

The output of the soil moisture sensor changes in the range of ADC value from 0 to 1023.

This can be represented as moisture value in terms of percentage using formula given below.

$$AnalogOutput = \frac{ADCValue}{1023}$$

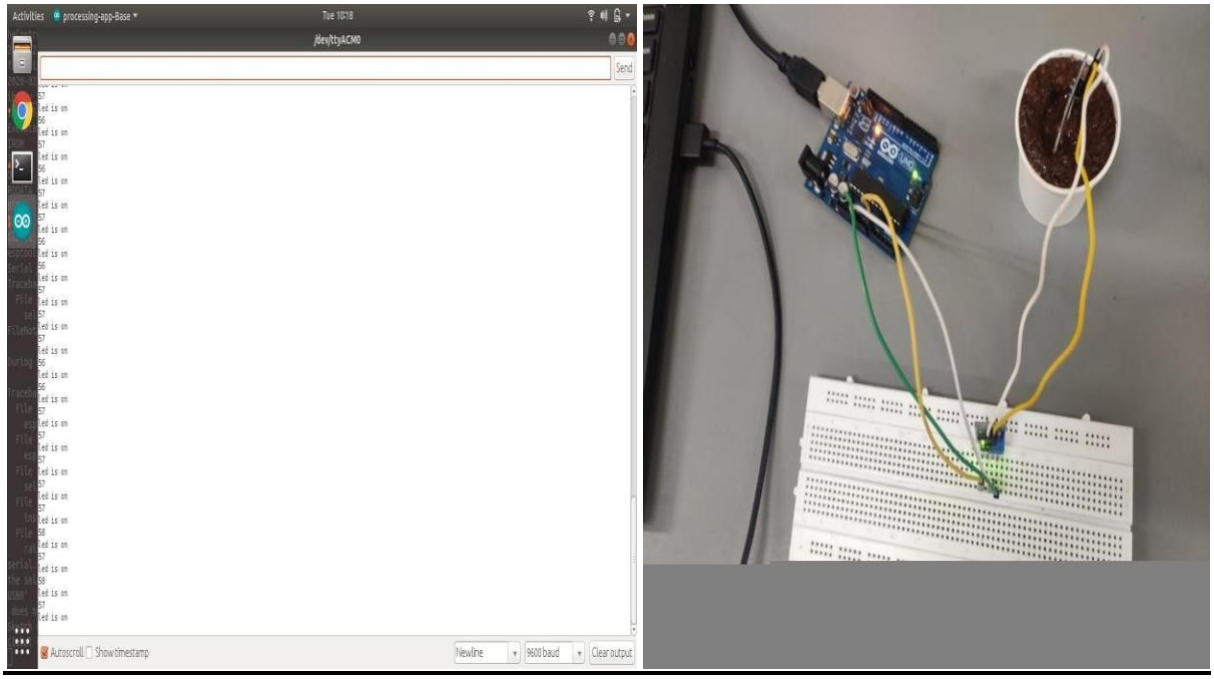
Moisture in percentage = 100 – (Analog output * 100)

For zero moisture, we get maximum value of 10-bit ADC, i.e. 1023. This, in turn, gives 0% moisture.

Program:

```
#include<SoftwareSerial.h>
void setup() {

  // put your setup code here, to run once:

pinMode(A4,INPUT);
pinMode(13,OUTPUT);

Serial.begin(9600);

}

void loop() {

int sensorvalue=analogRead(A4);

Serial.println(sensorvalue);

if(sensorvalue<500)

{

  Serial.println("led is on");

  digitalWrite(13,HIGH);

  delay(1000);

}

else
{
  digitalWrite(13,LOW);

  delay(1000);

}
```

Output:


Activities
processing-app-Base
Tue 10:18
/dev/ttyACM0
Send
Autoscroll
Show timestamp
Newline
9600 baud
Clear output

TASK 4

DC Motor and Relay

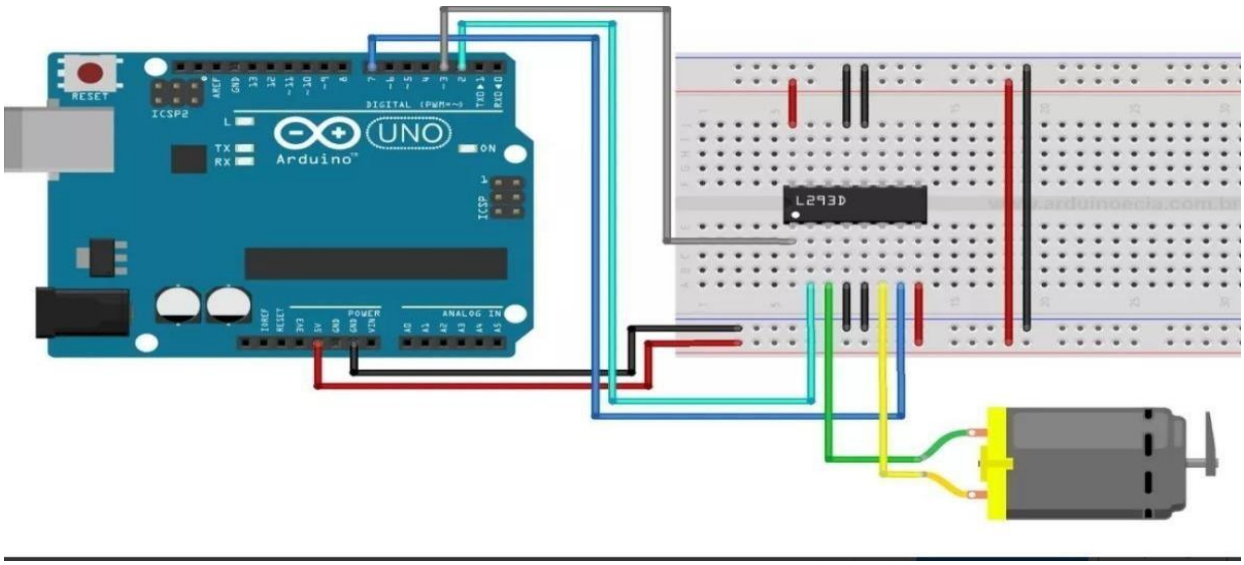
Task 4 :Write an ARDUINO Program for Micro controller-Actuator Interface.

a)DC Motor
b)Relay

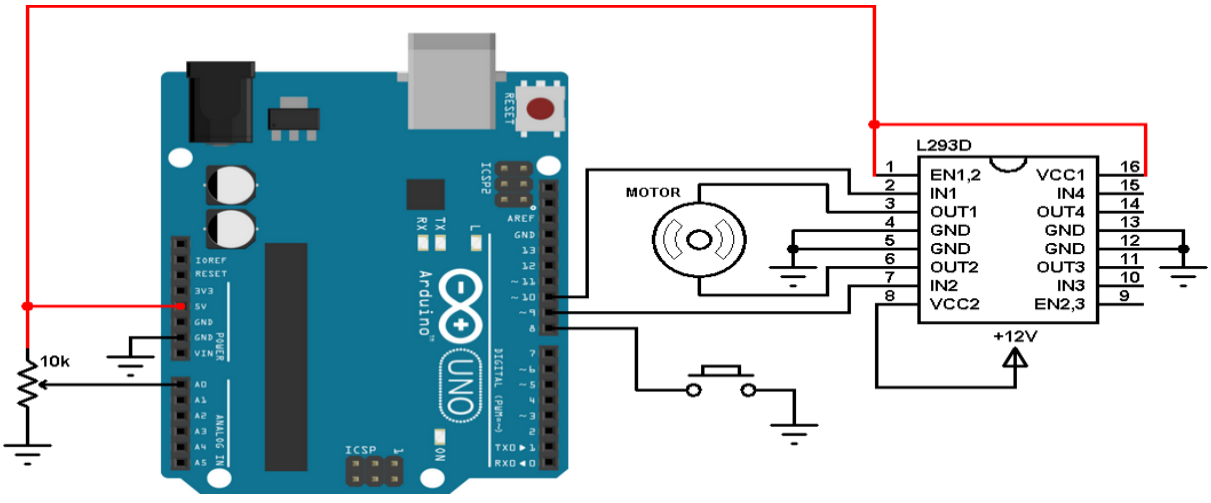
Aim: To write an Arduino program for rotating motor in clockwise and anti clock wisedirections.

Components Required: Arduino UNO, DC Motor, L293D IC, connecting wires, bread board, and Computer system with Arduino IDE.

Circuit:



Schematic Diagram:



Truth Table for Motor:

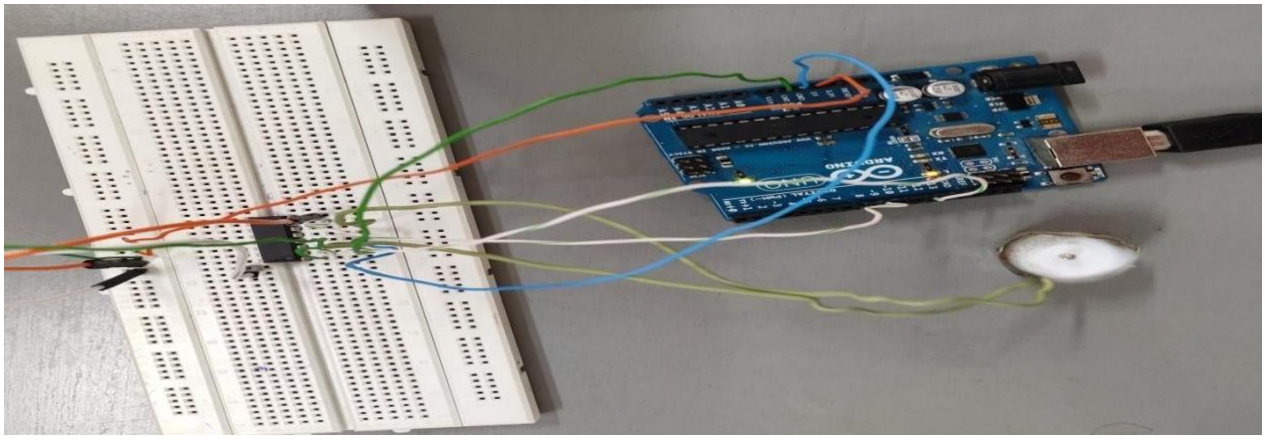
Enable	Input1	Input2	Motor action
H	H	L	Clockwise rotation
H	L	H	Counter-clockwise rotation
H	L	L	Stop
H	H	H	Stop

Program:

```
void setup(){
pinMode(10, OUTPUT);
pinMode(9, OUTPUT);
}
void loop(){
digitalWrite(10, LOW);
digitalWrite(9, HIGH);
delay(3000);
digitalWrite(9,LOW);
digitalWrite(10, HIGH);
delay(3000);
digitalWrite(9, LOW);
digitalWrite(10, LOW);
delay(3000);
}
```

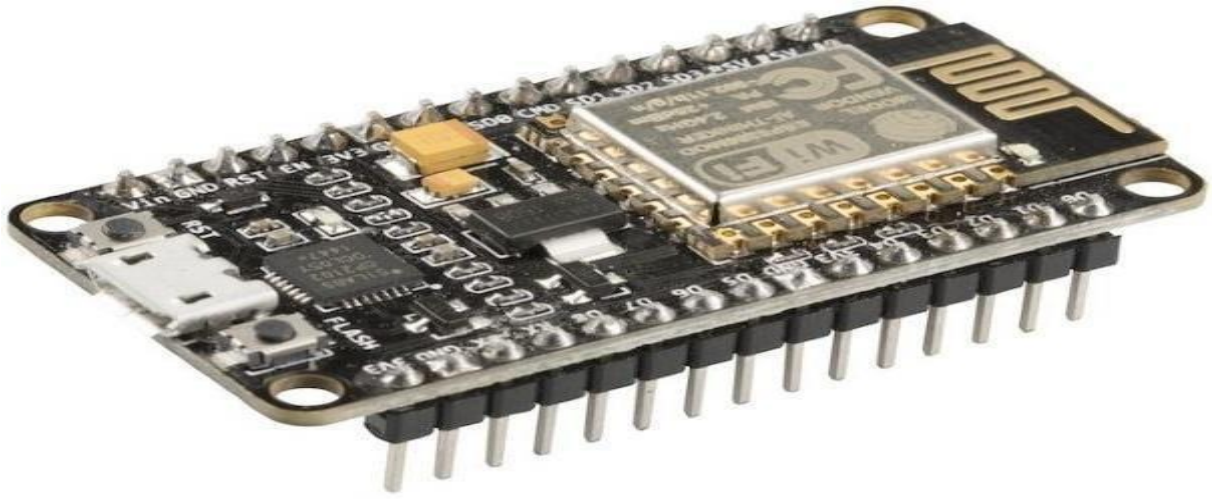
Output:

Motor rotates in clock wise and anti clockwise directions and finally stops

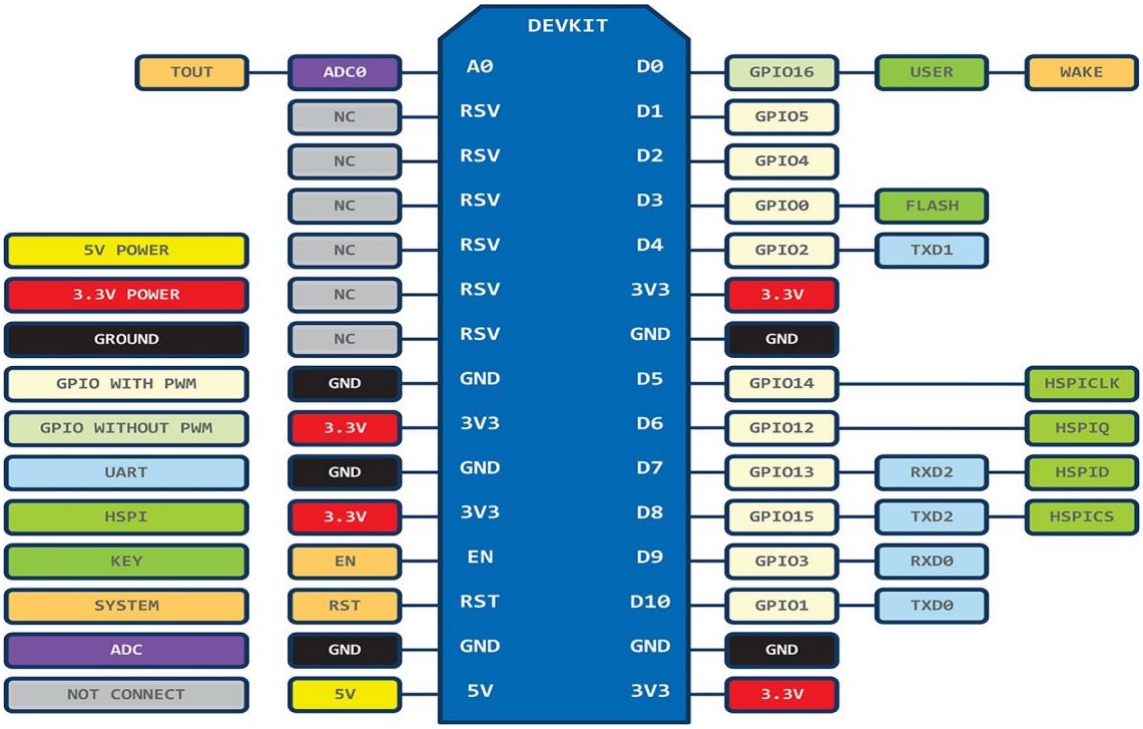


4(b)Remote Relay
Introduction to Node MCU

"**NodeMCU** is an open source IoT platform. It includes firmware which runs on the ESP8266 Wi-Fi SoC from Espressif Systems, and hardware which is based on the ESP-12 module. The term "NodeMCU" by default refers to the firmware rather than the development kits"



Pin Layout



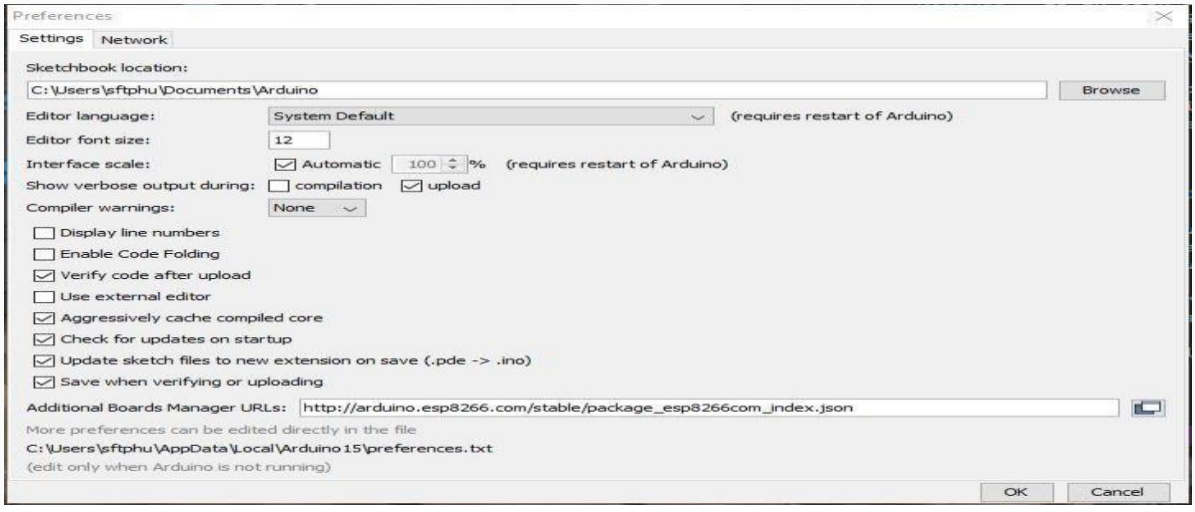
mapping list between NodeMCU pins and GPIO below:

- D0 = GPIO16
- D1 = GPIO5
- D2 = GPIO4
- D3 = GPIO0
- D4 = GPIO2
- D5 = GPIO14
- D6 = GPIO12
- D7 = GPIO13
- D8 = GPIO15
- D9 = GPIO3
- D10 = GPIO1
- LED_BUILTIN = GPIO16 (auxiliary constant for the board LED, not a board pin)

Setting up in Arduino IDE

Here is a simple walk-through for setting up the device in your Arduino IDE. Basically, we need to add the board to the list of available boards and choose it (together with the appropriate port and speed setup) for NodeMCU development. Using this approach, it is important to note the Lua firmware is overwritten. You can get it back if needed, so no issues.

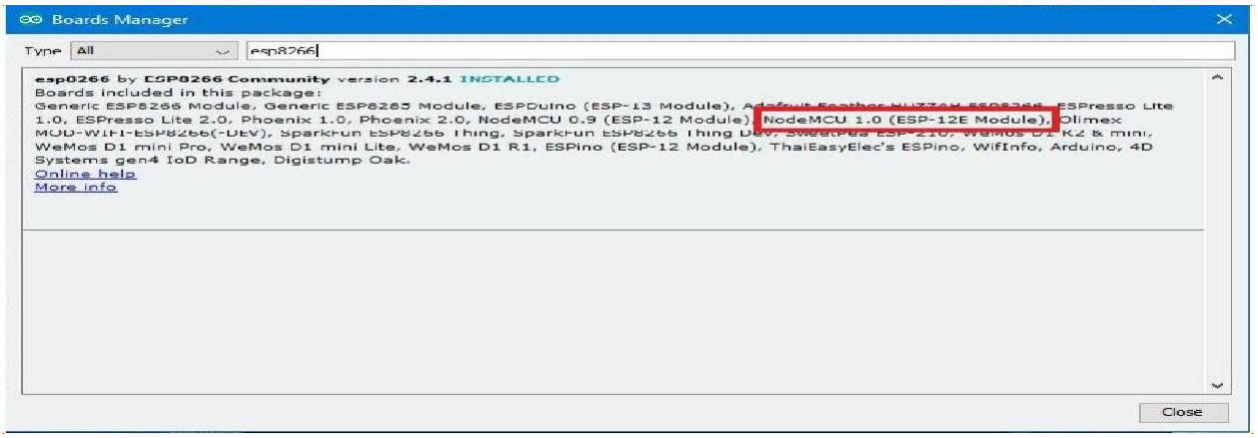
- We assume you have the Arduino IDE installed. If not, head to Arduino Home, navigate to the 'Software' section, download and install the latest version. You can also install the portable version and can use the setup in another IDE of choice (such Visual Studio Code), but we will not focus on this here
- Open 'Arduino IDE' and click **File - Preferences**



- Copy the URL below into the section **Additional boards Manager** and click OK to close the Preferences tab

  http://arduino.esp8266.com/stable/package_esp8266com_index.json

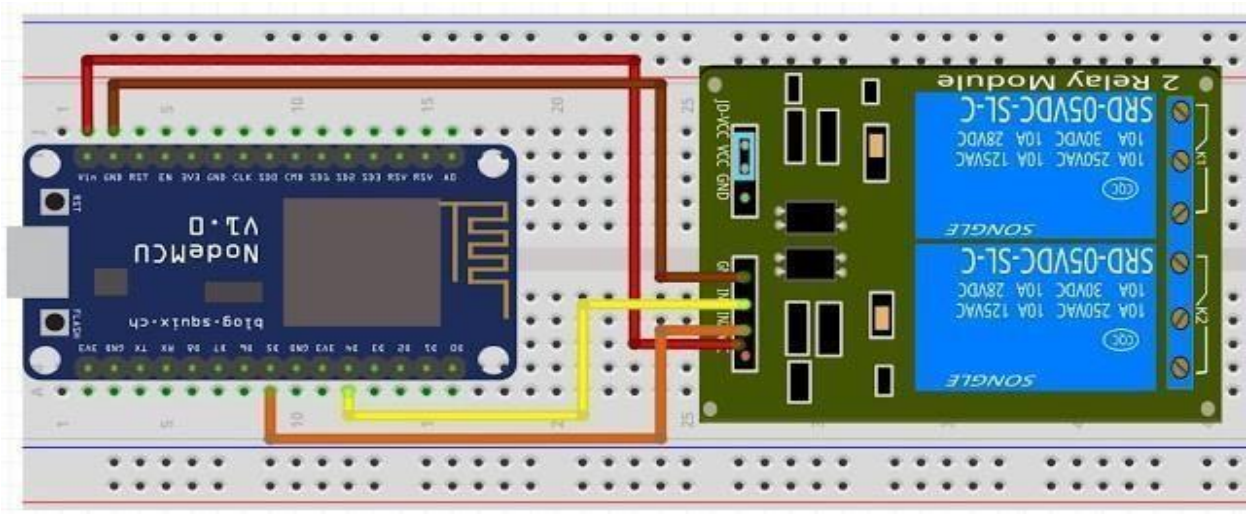
- Goto **Tools - Board ... - Board Manager**
- Locate the **esp8266 by esp8266 community** entry and install the software for Arduino (notice the NodeMCU entry in this board list)



Aim: To Remote relay operation using NodeMCU.

Components Required: NodeMCU, Two Channel Relay, USB Cable,Connecting wires,Computer with Arduino IDE.

Circuit:

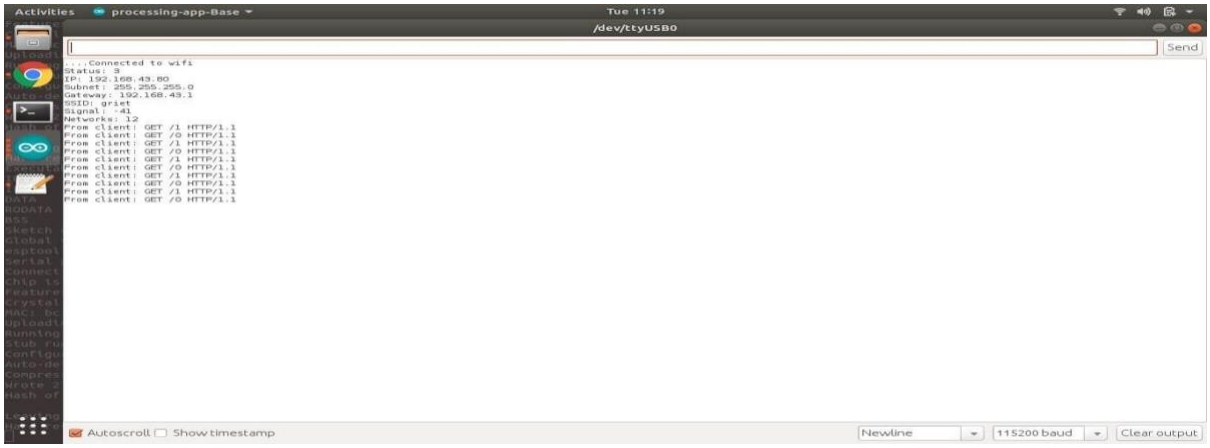


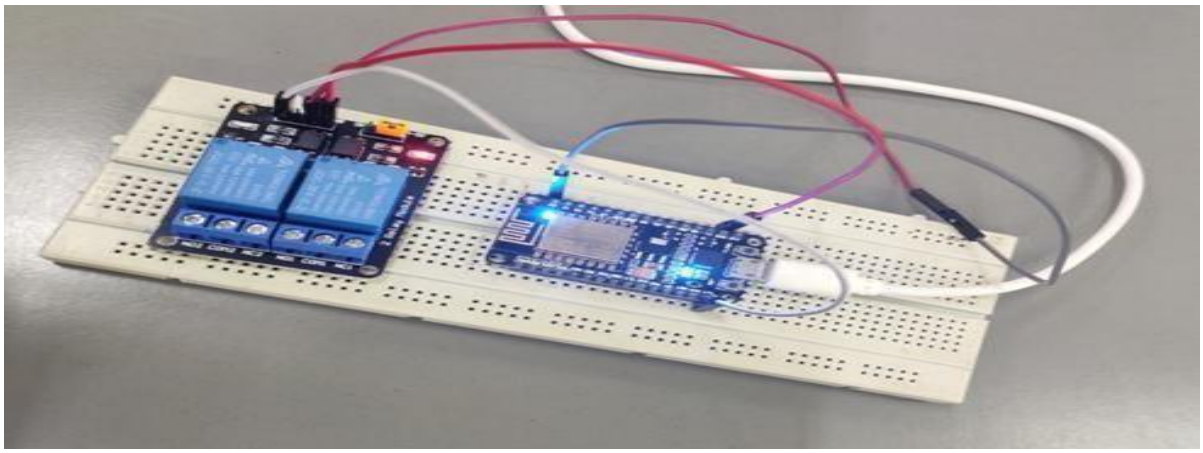
Program:

```
#include <ESP8266WiFi.h>

byte ledPin = 2;
char ssid[] = "griet";              // SSID of your home WiFi
char pass[] = "griet123";           // password of your home WiFi
WiFiServer server(9000);

IPAddress ip(192, 168,43, 80);          // IP address of the server
IPAddress gateway(192,168,43,1);        // gateway of your network
IPAddress subnet(255,255,255,0);        // subnet mask of your network

void setup()

{
  pinMode(D0,OUTPUT);
  pinMode(D1,OUTPUT);
  Serial.begin(115200);                 // only for debug

  WiFi.config(ip, gateway, subnet);     // Rather than the DHCP IP given by the phone hotspot
this forces the ip to stay on 192.168.43.80
  WiFi.begin(ssid, pass);               // connects to the WiFi router
  while (WiFi.status() != WL_CONNECTED)

{
    Serial.print(".");
    delay(500);
  }
  server.begin();                       // starts the server
  Serial.println("Connected to wifi");
  Serial.print("Status: ");

  Serial.println(WiFi.status()); // some parameters of the device when connected the network
  Serial.print("IP: ");

  Serial.println(WiFi.localIP());
  Serial.print("Subnet: ");

Serial.println(WiFi.subnetMask());
  Serial.print("Gateway: ");

Serial.println(WiFi.gatewayIP());
  Serial.print("SSID: ");

Serial.println(WiFi.SSID());
```

```
  Serial.print("Signal: ");
Serial.println(WiFi.RSSI());
  Serial.print("Networks: ");
Serial.println(WiFi.scanNetworks());
  pinMode(ledPin, OUTPUT);
}

void loop ()

{
  WiFiClient client = server.available();
  if (client)

 {
    if (client.connected())
{
      String request = client.readStringUntil('\r');    // receives the message from the client
      Serial.print("From client: ");
      Serial.println(request);
      client.flush();
      client.print("HTTP/1.1 200 OK\r\n");
      client.flush();
      short int i=0;
      for(i=0;request[i]!='/';i++);
      digitalWrite(D0,49-request[++i]);
    }

  }

}
```

Output:


Activities
processing-app-Base
Tue 11:19
/dev/ttyUSB0
Send
...Connected to wifi
Status: 3
IP: 192.168.43.80
Subnet: 255.255.255.0
Gateway: 192.168.43.1
SSID: griet
Signal: -41
Networks: 12
From client: GET /1 HTTP/1.1
From client: GET /0 HTTP/1.1
From client: GET /1 HTTP/1.1
From client: GET /0 HTTP/1.1
From client: GET /1 HTTP/1.1
From client: GET /0 HTTP/1.1
From client: GET /1 HTTP/1.1
From client: GET /0 HTTP/1.1
From client: GET /1 HTTP/1.1
From client: GET /0 HTTP/1.1
Autoscroll
Show timestamp
Newline
115200 baud
Clear output



TASK 5

Liquid Crystal Display(LCD)

TASK 5: Write an ARDUINO Program for Moving curtail display using LCD.

Introduction: LCD Shield mainly constitutes 16*2 LCD.A Liquid Crystal Display(LCD) is a flat panel display, electronic visual display or video display that uses the light modulating properties of liquid crystal

Pin-out:

LCD Pin	Arduino Pin
RS	8
EN	9
D4	4
D5	5
D6	6
D7	7

Header File:

```
#include<LiquidCrystal.h>
```

Functions

1. **LiquidCrystal lcd(rs, enable, d4, d5 ,d6, d7) :** Creates a variable of type LiquidCrystal.The display can be controlled using data lines.
2. **rs :** The number of Arduino pin that is connected to RS(register Select) pin on LCD.
3. **enable :** The number of Arduino pin that is connected to enable pin on LCD.
4. **d4, d5, d6, d7:** The number of arduino pin that is connected to corresponding data pins on LCD.
5. **lcd.begin(cols,rows) :** Specifies the dimensions of the display(usually cols=16,rows=2).
6. **lcd.clear() :** Clears the LCD screen and positions the cursor in the upper left corner.
7. **lcd.setCursor(col,row) :** Positions the LCD cursor at the given row and column of the LCD screen.
8. **lcd.write(data) :** Writes a character onto the LCD.
9. **lcd.print(data) :** Prints text on the LCD.
10. **lcd.display() :** Turns on the LCD display.
11. **lcd.nodisplay() :** Turns off the LCD display.
12. **lcd.scrollDisplayLeft() :** Scrolls the contents of the display one space to the left.
13. **lcd.scrollDisplayRight() :** Scrolls the contents of the display one space to the right.
14. **lcd.autoscroll() :** Turns on automatic scrolling of the LCD. If current text direction is left-to- right, the display scrolls to the left. . If current text direction is right-to-left, the display scrolls to the right.
15. **lcd.noAutoscroll() :** Turns off automatic scrolling of LCD.
16. **lcd.leftToRight() :** Sets the direction for text written to left-to-right, the default. This means that subsequent characters written to the display will go from left to right.
17. **lcd.rightToLeft() :** Sets the direction for text written to right-to-left. This means that subsequent characters written to the display will go from right to left.
18. **lcd.createChar(num,data) :** Creates a custom character for use on LCD.num specifies the character to be overridden. data specifies the custom character array name.

TASK: Write an ARDUINO Program for Moving curtail display using LCD .

Aim: To Write an Arduino program for Moving curtail display using LCD.

Components Required: ARDUINO UNO,LCD, USB Cable, Computer system with Arduino IDE.

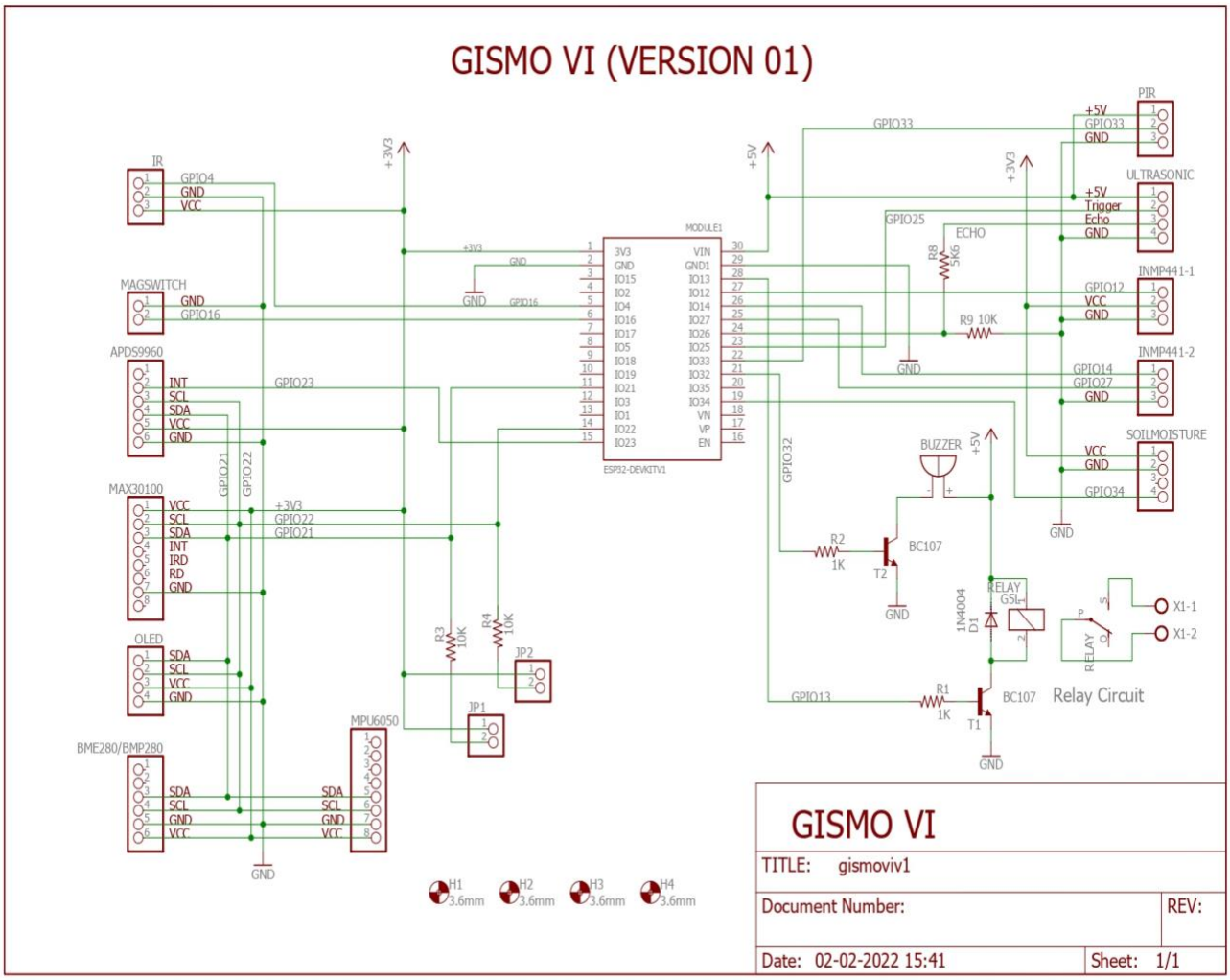
Program:

```
#include<LiquidCrystal.h>
LiquidCrystal lcd(8,9,4,5,6,7);
int i=0;
char ch;
char s[]="Welcome to MC LAB!";
void setup()
{
  lcd.begin(16,2);
}
void loop()
{
  lcd.setCursor(0,0);
  ch='!';
  lcd.print(ch);
  delay(1000);
  lcd.clear();
  lcd.setCursor(3,0);
  for(i=0;s[i]!='\0';i++)
  {
  lcd.print(s[i]);
  if(i==10)
  lcd.setCursor(5,1);
  }
  delay(2000);
  lcd.clear();}
```

Output:


LCD Keypad Shield
Welcome to
MC LAB!
SELECT
LEFT
UP
DOWN
RIGHT
RST
ROBOT
THE FUTURE

GISMO VI Schematic Diagram



03-02-2022 09:04 C:\Users\user\Desktop\Hardware Design\GISMO VI\gismoviv1.sch (Sheet: 1/1)

TASK 6

TASK 6: Write an ARDUINO Program for Sensor /Actuator Interfacing using ESP32.(MPU6050)

Aim: To Write an Arduino program for Sensor /Activator Interfacing using ESP32.

Components Required: GISMO VI Board, MPU 6050,PIR Sensor,Cable.

Program:

```
#include "I2Cdev.h"
#include "MPU6050.h"
#include "Wire.h"
MPU6050 mpu;
float baseline[3];
float features[3];
float motion_threshold = 0.7;
void setup() {
Wire.begin();
Serial.begin(115200);
mpu.initialize();
calibrate();
}
void loop() {
float ax,ay,az;
mpu_read(&ax,&ay,&az);
ax=ax-baseline[0];
ay=ay-baseline[1];
az=az-baseline[2];
mpu_record();
delay(2000);
}
void mpu_read(float *ax,float *ay,float *az) {
int16_t _ax, _ay, _az, _gx, _gy, _gz;
mpu.getMotion6(&_ax, &_ay, &_az, &_gx, &_gy, &_gz);
```

```
*ax = _ax/16384.0;
*ay = _ay/16384.0;
*az = _az/16384.0;
}
void calibrate(){
float ax,ay,az;
for(int i=0;i < 10;i++){
mpu_read(&ax,&ay,&az);
delay(100);
}
baseline[0]=ax;
baseline[1]=ay;
baseline[2]=az;
}
void mpu_record(){
float ax,ay,az;
float aax,aay,aaz;
String position;
mpu_read(&ax,&ay,&az);
ax = ax - baseline[0];
ay = ay - baseline[1];
az = az - baseline[2];
features[0] = ax;
features[1] = ay;
features[2] = az;
Serial.print(features[0]);
Serial.print(" ");
Serial.print(features[1]);
Serial.print(" ");
Serial.println(features[2]);
```

```
aax=fabs(ax);
aay=fabs(ay);
aaz=fabs(az);
Serial.print(aax);
Serial.print(" ");
Serial.print(aay);
Serial.print(" ");
Serial.println(aaz);
position="Upright";
if(aax > motion_threshold)
{
if(ax > 0)
position="Left";
else
position="Right";
}
if(aay > motion_threshold)
{
if(ay > 0)
position="Backward";
else
position="Forward";
}
Serial.println(position);
}
```

OUTPUT: Gravitational Force along XYZ Axis.

Task 7

Mobile App for Simple Interface

Task 7

a) Write an ARDUINO Program to Develop a Mobile App for simple interface.

b)Design a Mobile App to work with data of a cloud.

a) DevelopaMobileappforsimpleUser Interface

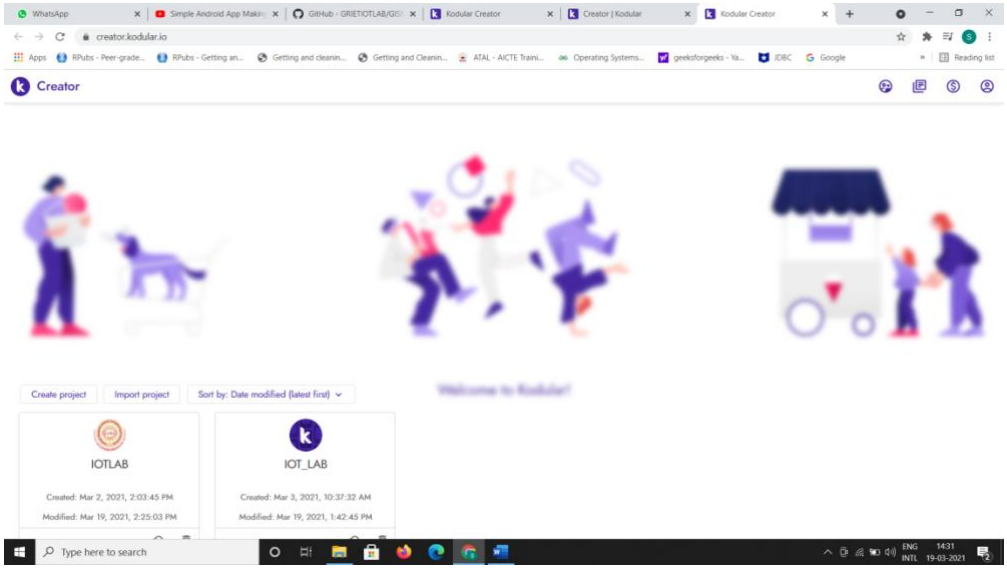
There are several steps to develop a mobile app. These are the following Steps:

Step 1: go to Kodular Creator website by clicking below link you can go the website.

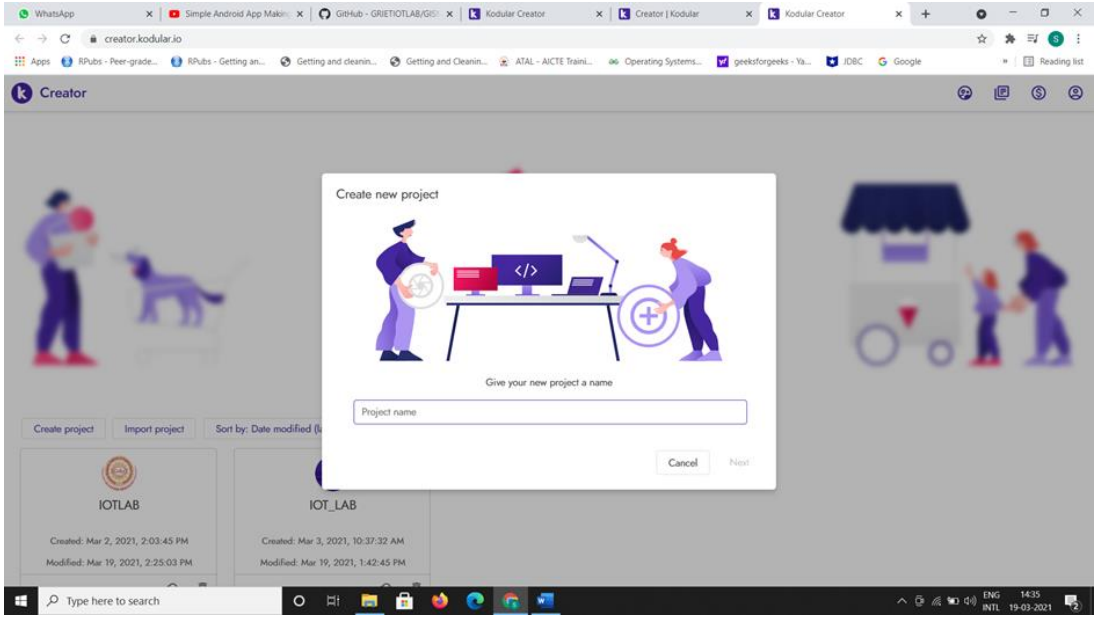
https://www.kodular.io/creator

Step 2 : Click on get Started button

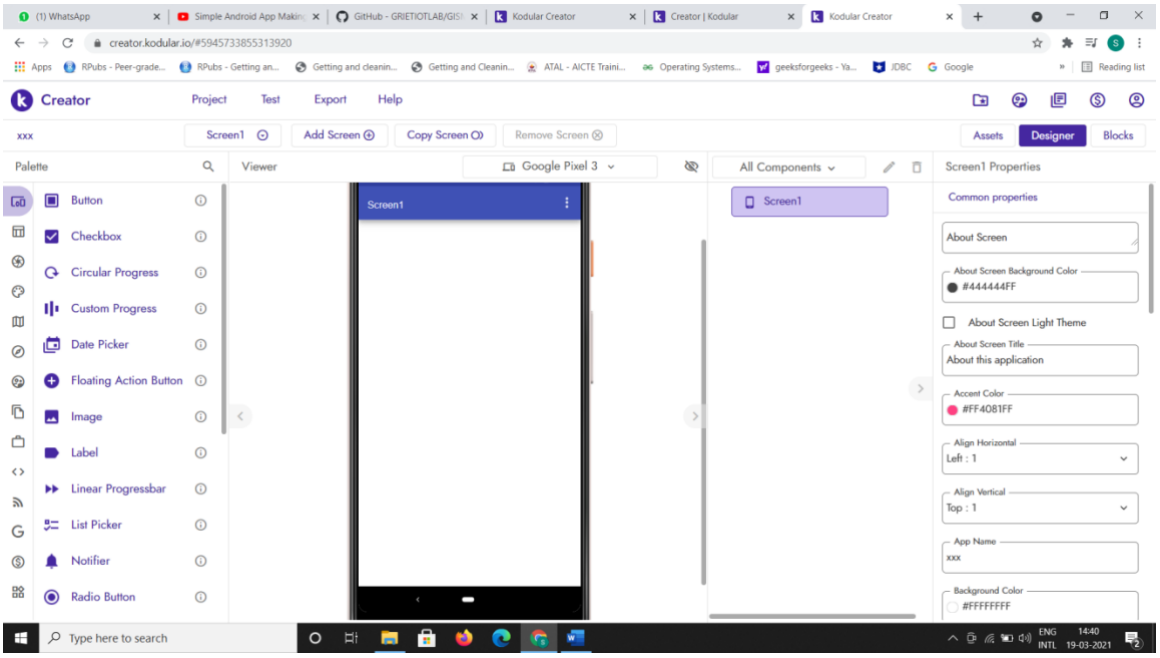
Step 3: then you will get a scree like this



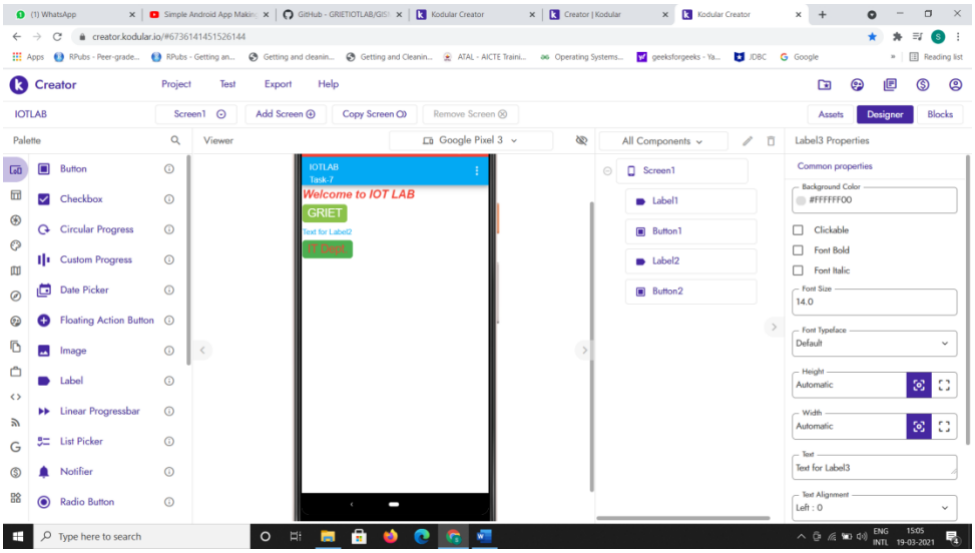
Step 4: Click on create project and give project name



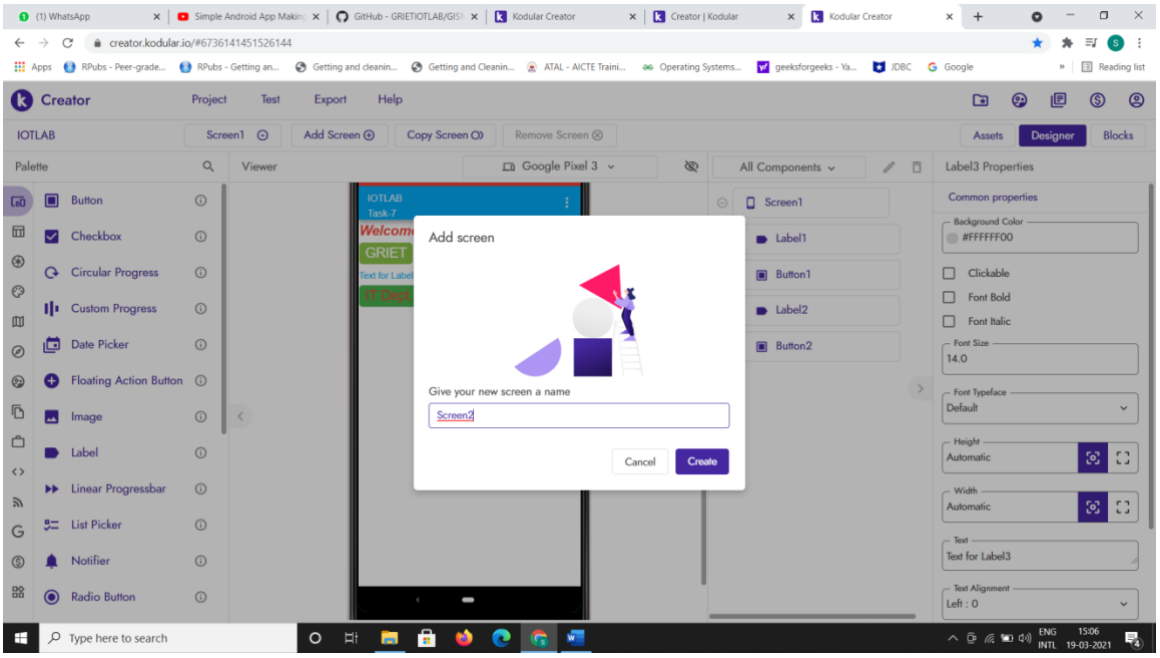
Step 5: Then click on next and finish. Then your project will be created and window will be opened like this



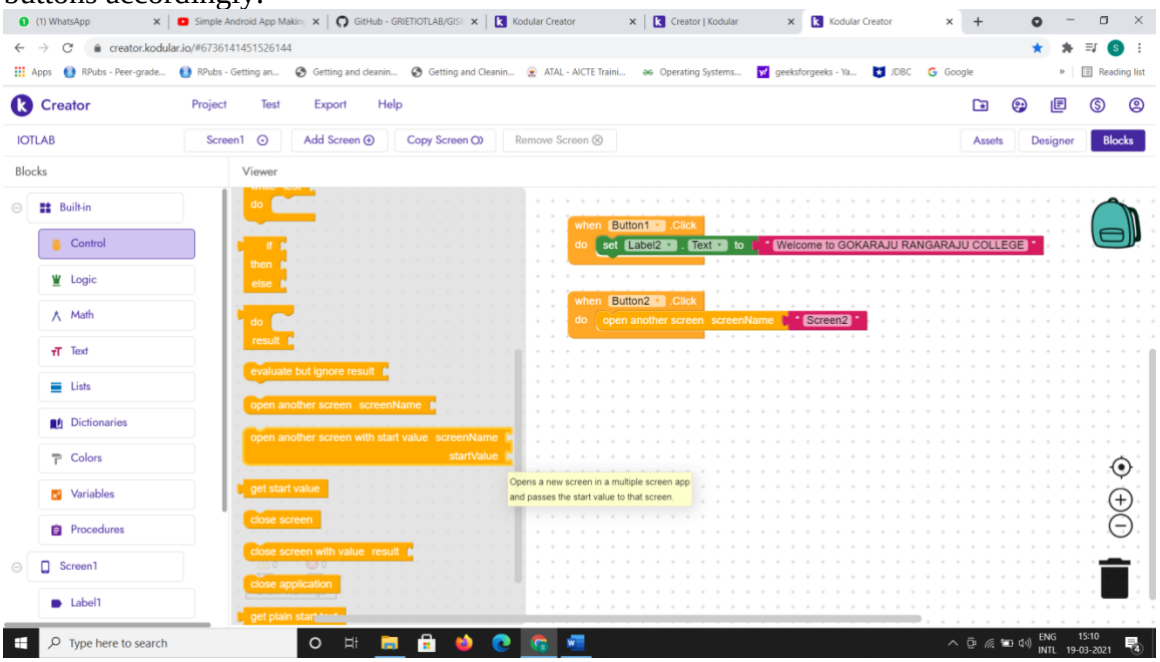
Step 6: then go to the left side pallet and click on User Interface then you will find all GUI items like buttons, lables, etc..as shown in the above figure.
Step 7: drag and drop the controls which you need and those controls will be visible in your app.
Here as a sample mobile app, drag and drop one label and add the text as "Welcome to IOT LAB" and you can apply colour,font etc…
Step 8: And take 2 button one as GRIET and the other is IT Dept and took two labels one will be printing the text when GRIET button is clicked in the same screen and other will print the text when IT Dept. button is clicked in another screen.



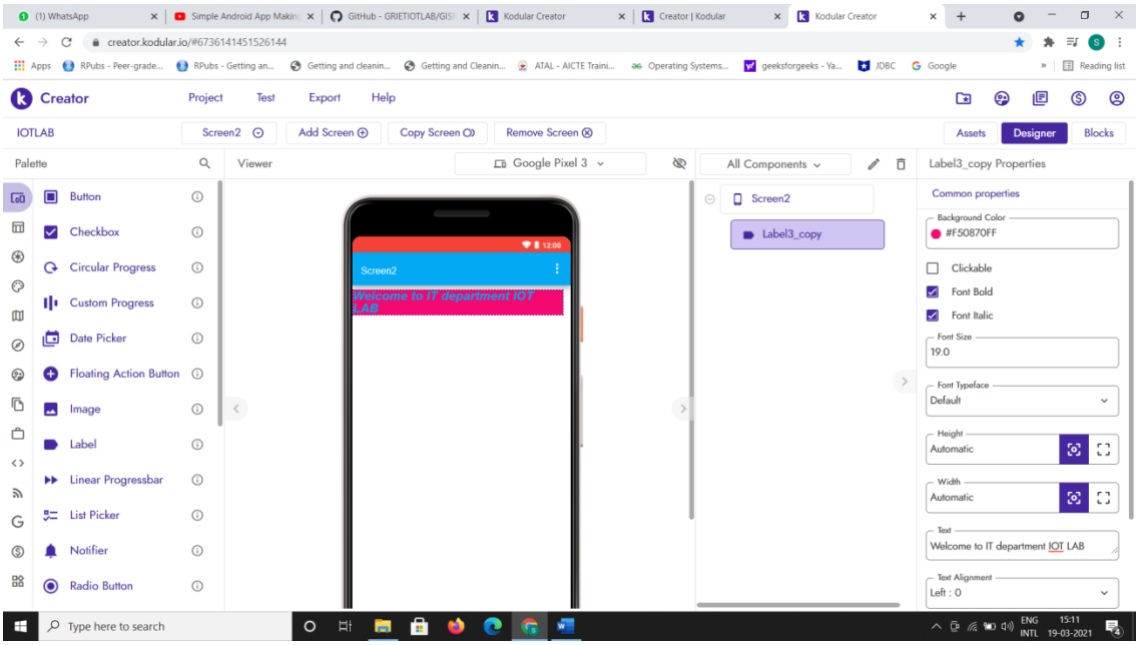
Step 9: click on Add Screen and click on create button then a new screen will be created.



Step 10: go to screen1 and go to Blocks tab on the right to write the code to get the text when you click the buttons accordingly.

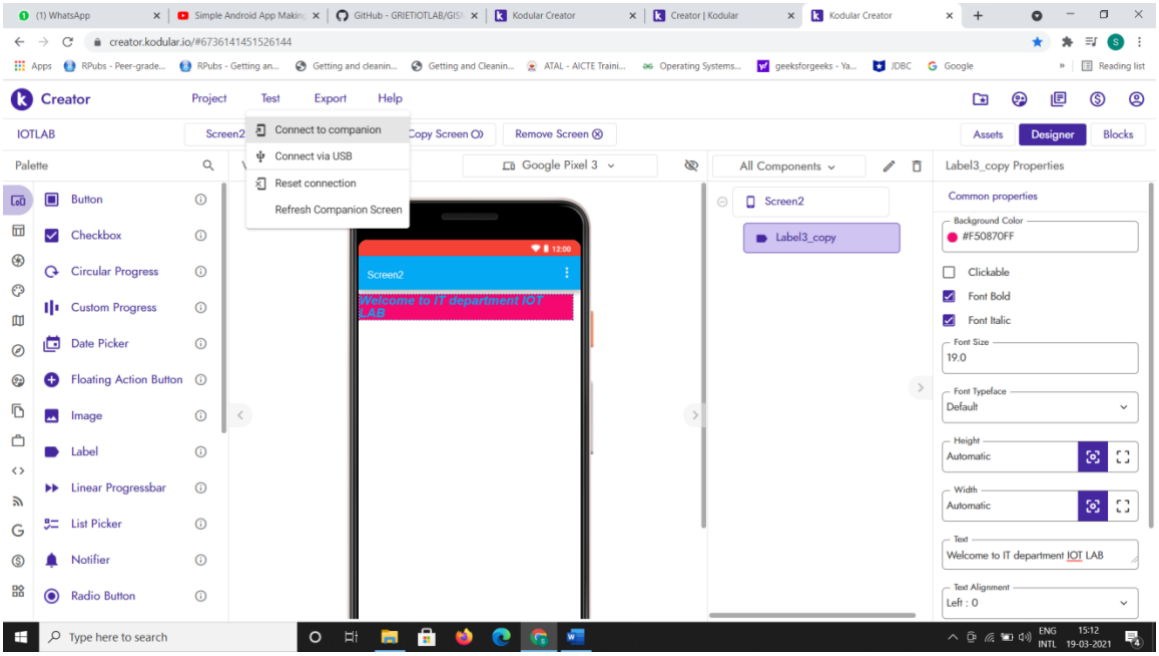


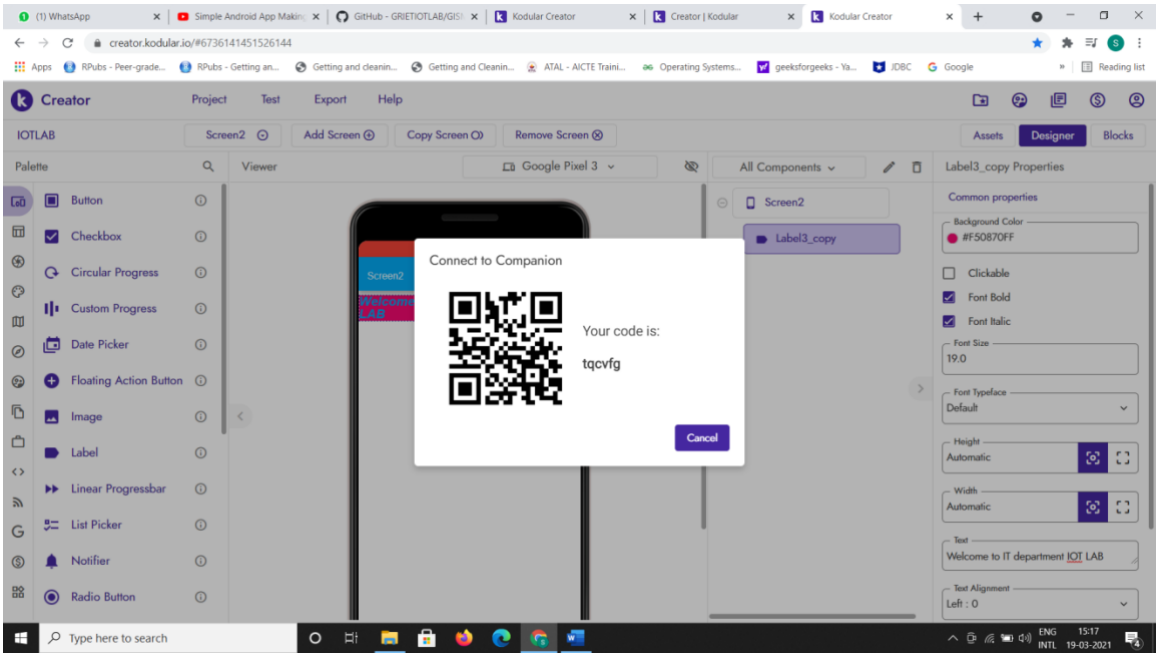
Step 11: go to screen2 and put a label and write the text that should be displayed when you click on IT Dept. button in screen.



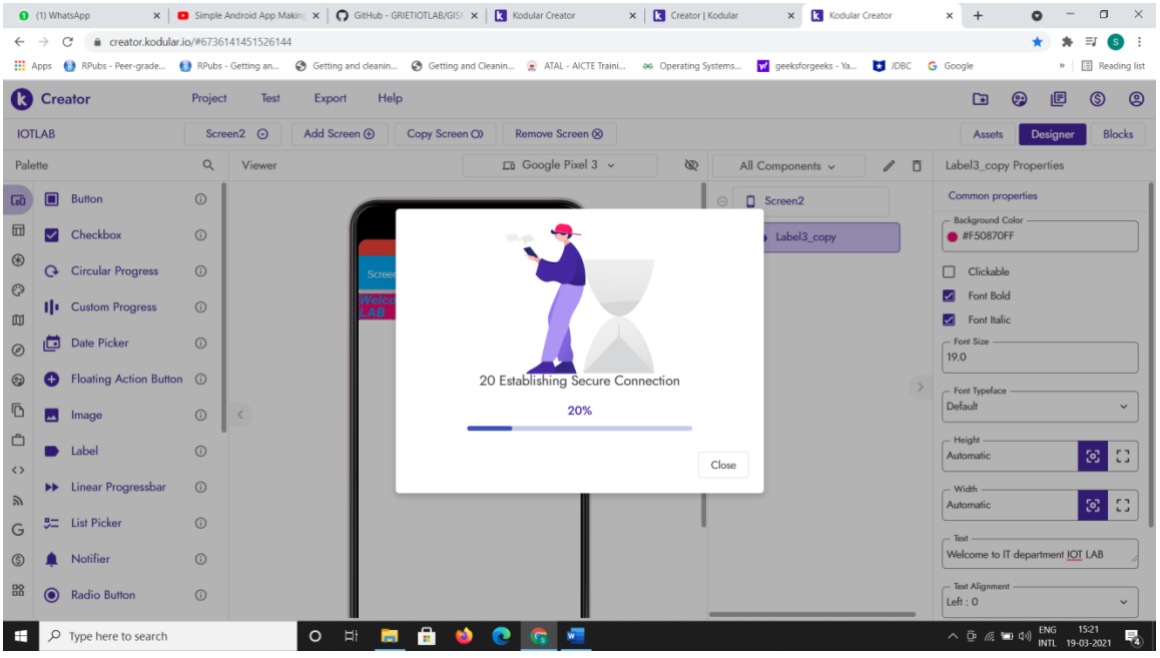
Step12: now test the mobile app, buy clicking test and connect to companion. Before this testing open google play store in your mobile and install the app named “Kodular Companion”.

Step 13: then open that app and scan the QR code which we get when we click on connect to companion in Kodular Creator.

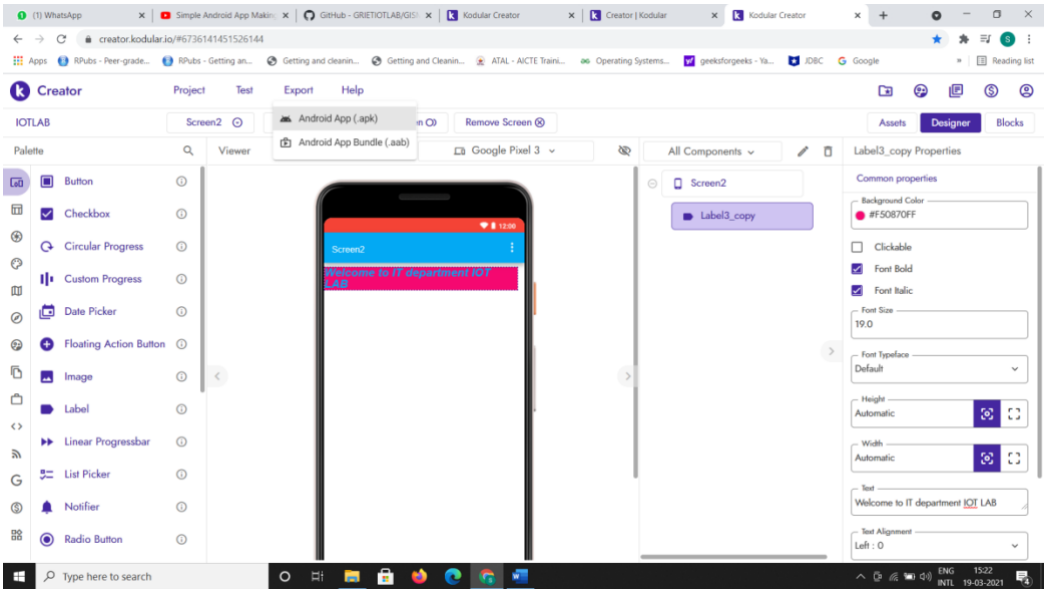


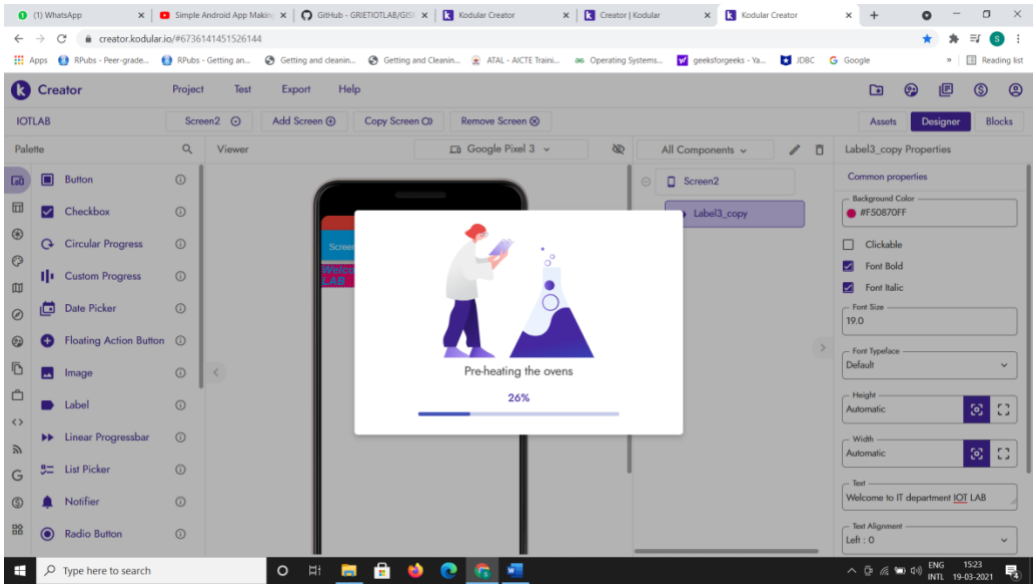


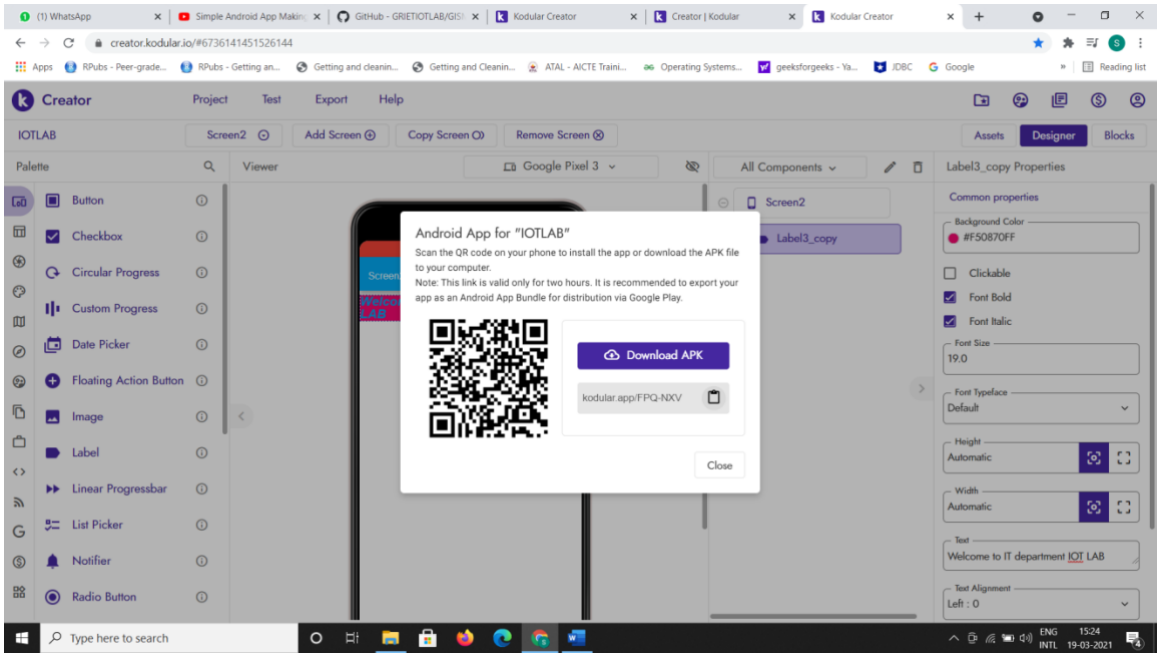
Step 14: If you scan the QR code in your mobile the app will be opened in your mobile. Now test the app whether it is working correctly or not.

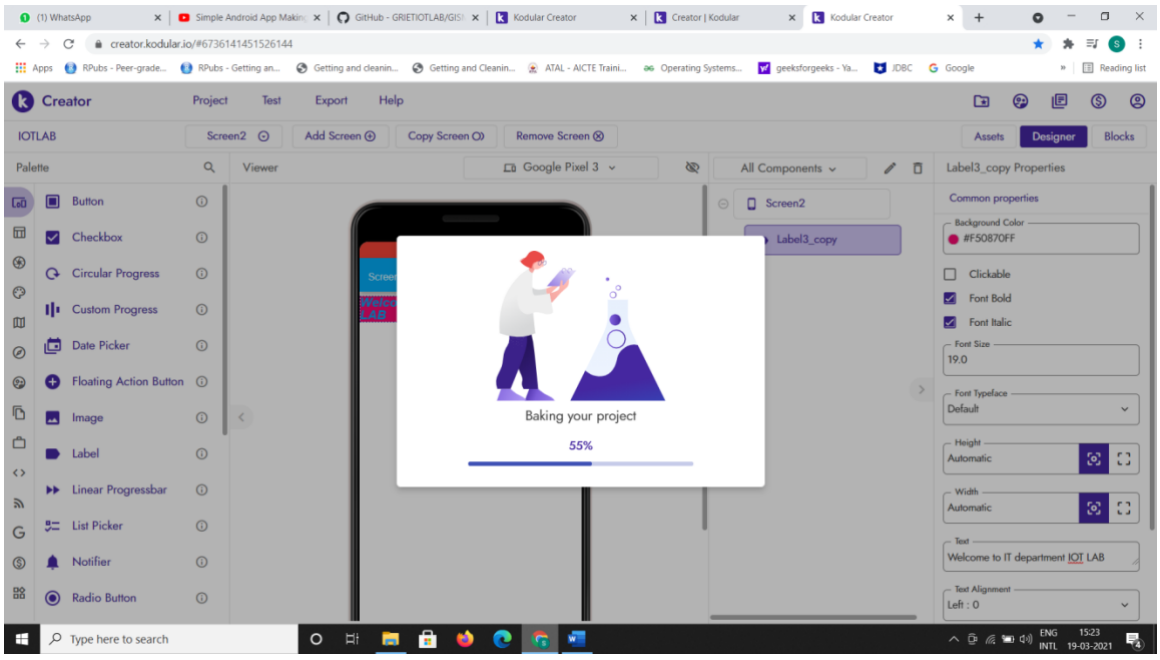


Step15: then to create permanent mobile app , go to Export option and click android app.


Creator
Project
Test
Export
Help
IOTLAB
Screen2
Android App (.apk)
Android App Bundle (.aab)
Remove Screen
Assets
Designer
Blocks
Palette
Viewer
Google Pixel 3
All Components
Label3_copy Properties
Button
Checkbox
Circular Progress
Custom Progress
Date Picker
Floating Action Button
Image
Label
Linear Progressbar
List Picker
Notifier
Radio Button
Label3_copy
Common properties
Background Color
#F50870FF
Clickable
Font Bold
Font Italic
Font Size
19.0
Font Typeface
Default
Height
Automatic
Width
Automatic
Text
Welcome to IT department IOT LAB
Text Alignment
Left : 0
Type here to search

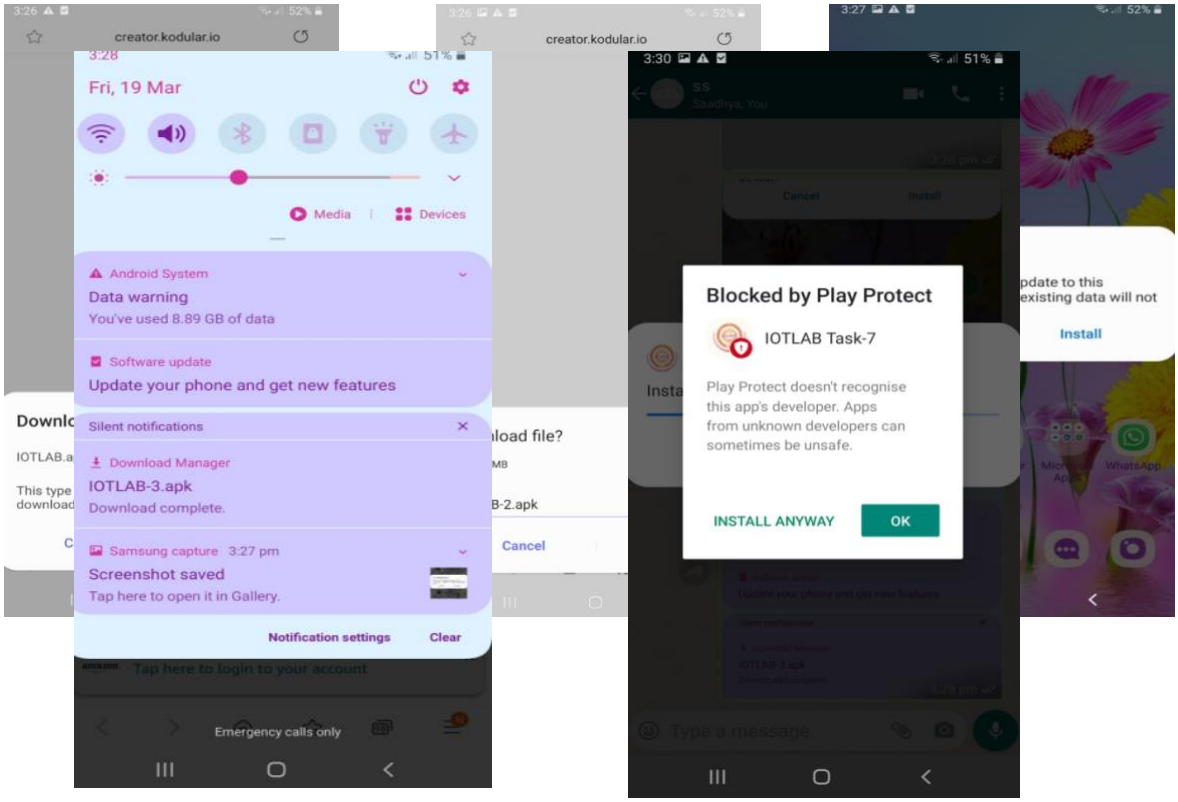

Creator
Project
Test
Export
Help
IOTLAB
Screen2
Add Screen
Copy Screen
Remove Screen
Assets
Designer
Blocks
Palette
Viewer
Google Pixel 3
All Components
Label3_copy Properties
Button
Checkbox
Circular Progress
Custom Progress
Date Picker
Floating Action Button
Image
Label
Linear Progressbar
List Picker
Notifier
Radio Button
Label3_copy
Common properties
Background Color
#F50870FF
Clickable
Font Bold
Font Italic
Font Size
19.0
Font Typeface
Default
Height
Automatic
Width
Automatic
Text
Welcome to IT department IOT LAB
Text Alignment
Left : 0
Pre-heating the ovens
26%
Type here to search





Step16: Now scan the QR code and also Click on download APK. Then download the app in mobile. Make sure that mobile and laptop should be connect to same wifi. Then you will get app in mobile.

Follow the screen shots..

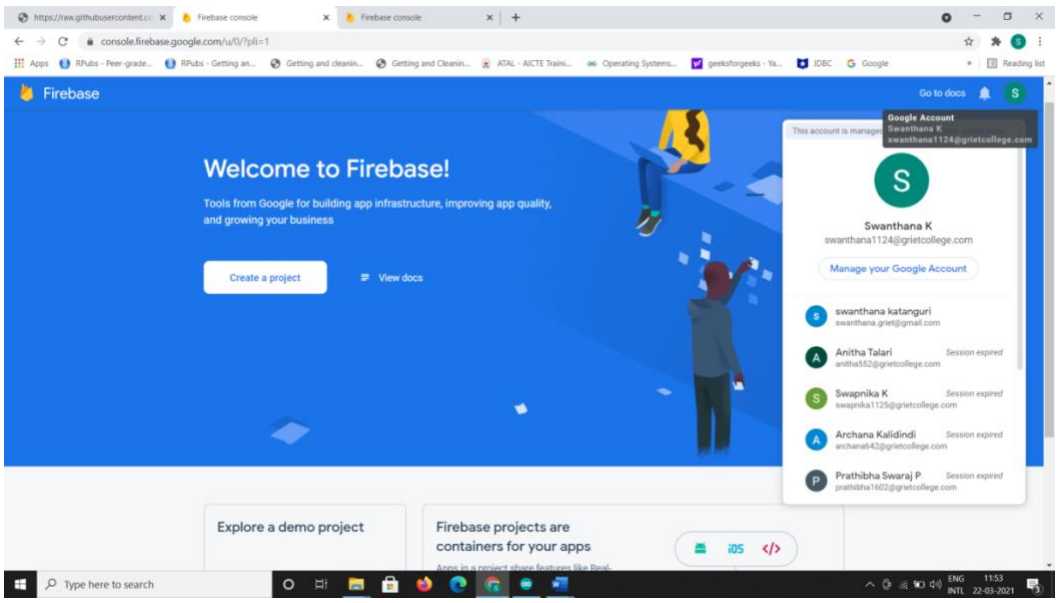


b) Design a Mobile app to work with data of a cloud

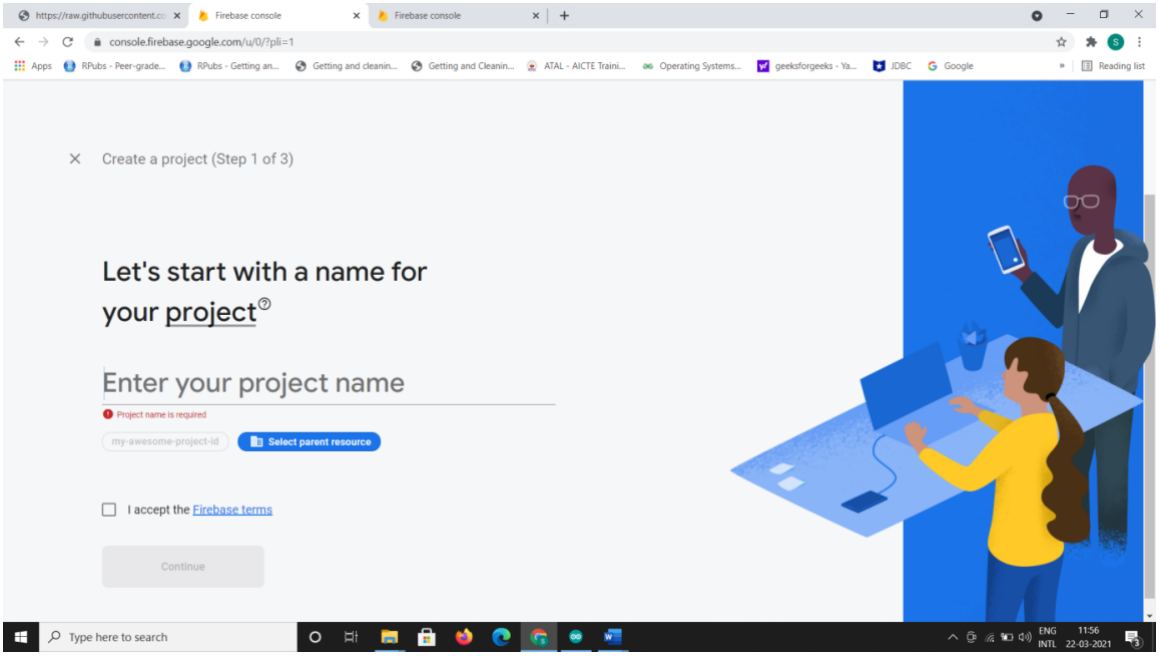
Program:

Follow the below steps to create a firebase(cloud) project.

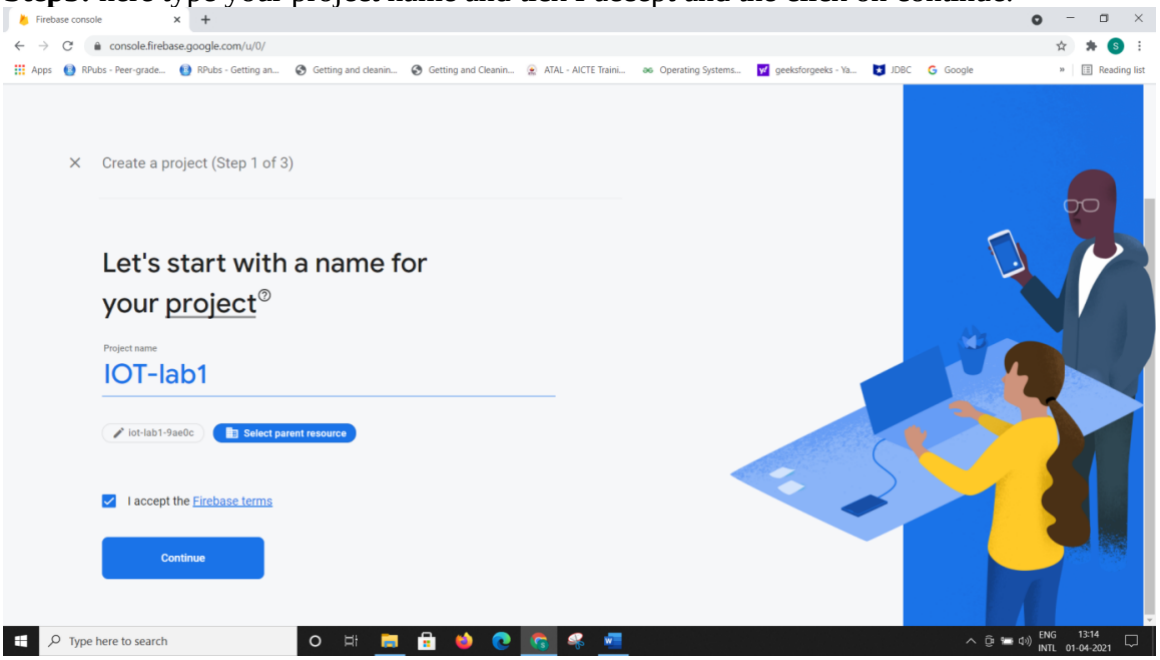
Step1:



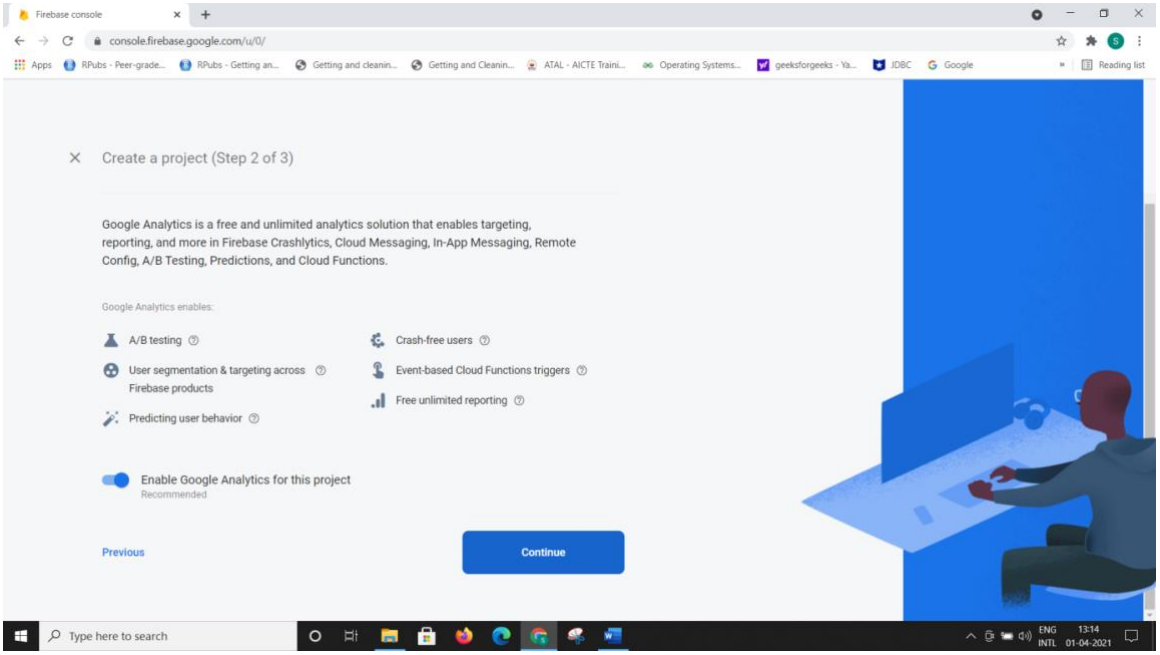
Step2: click on create project.



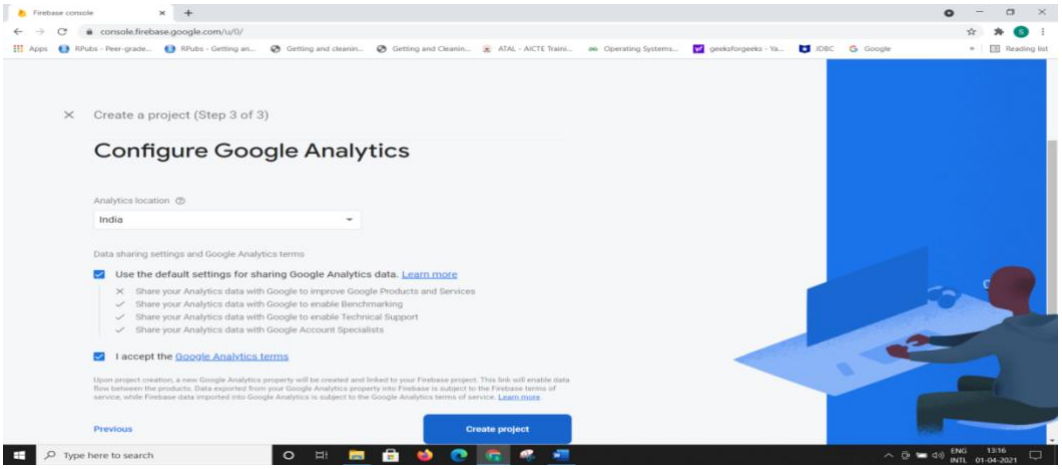
Step3: here type your project name and tick I accept and the click on continue.

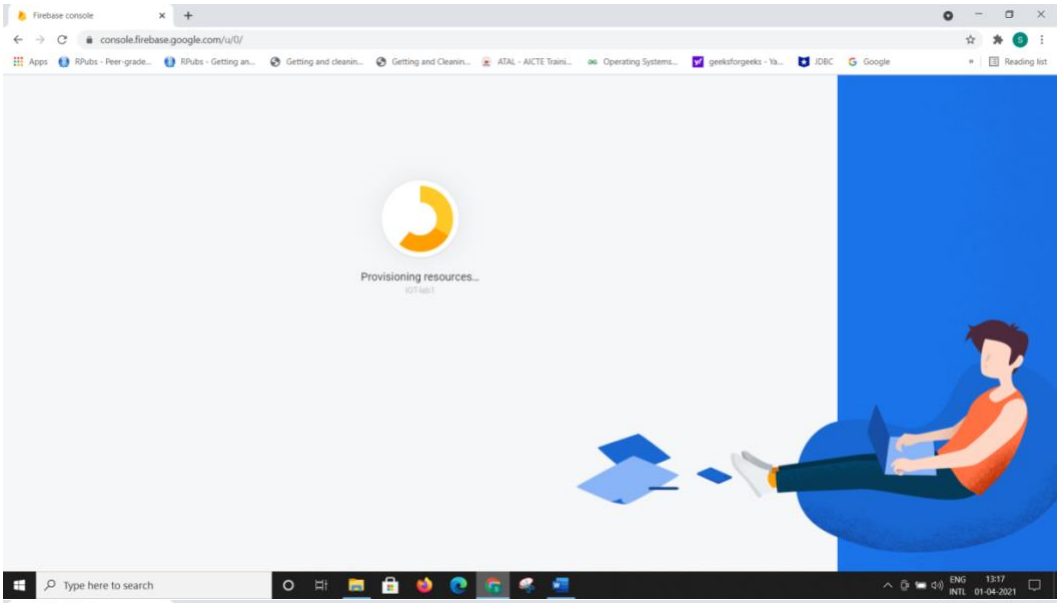


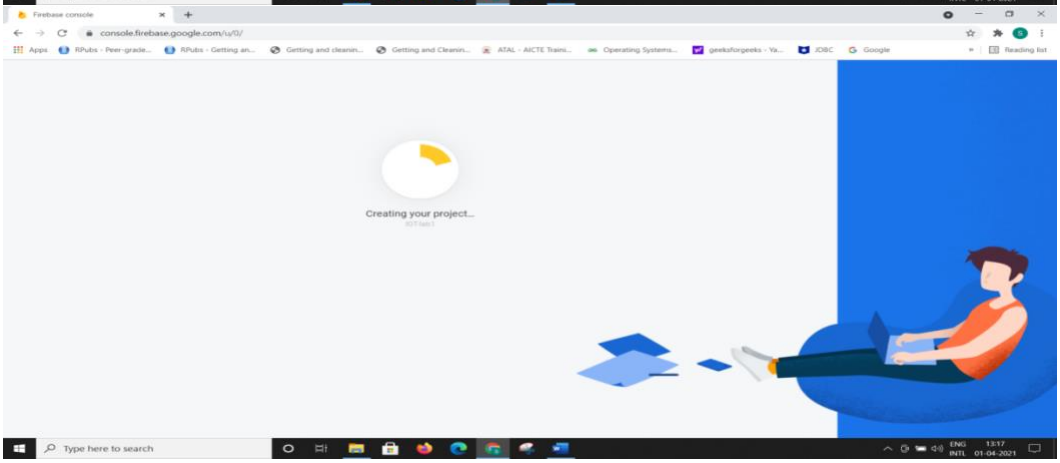
Step4:click on continue

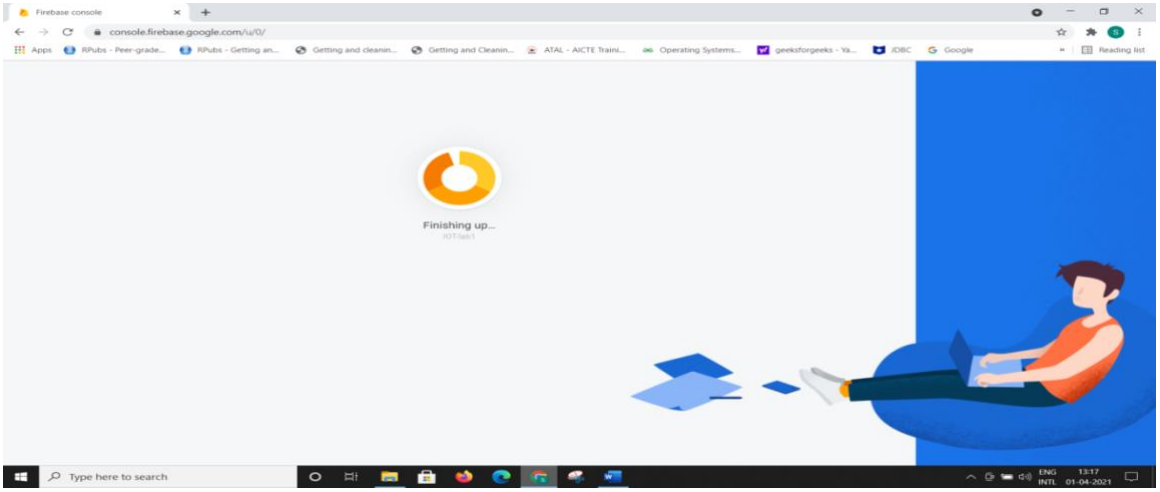


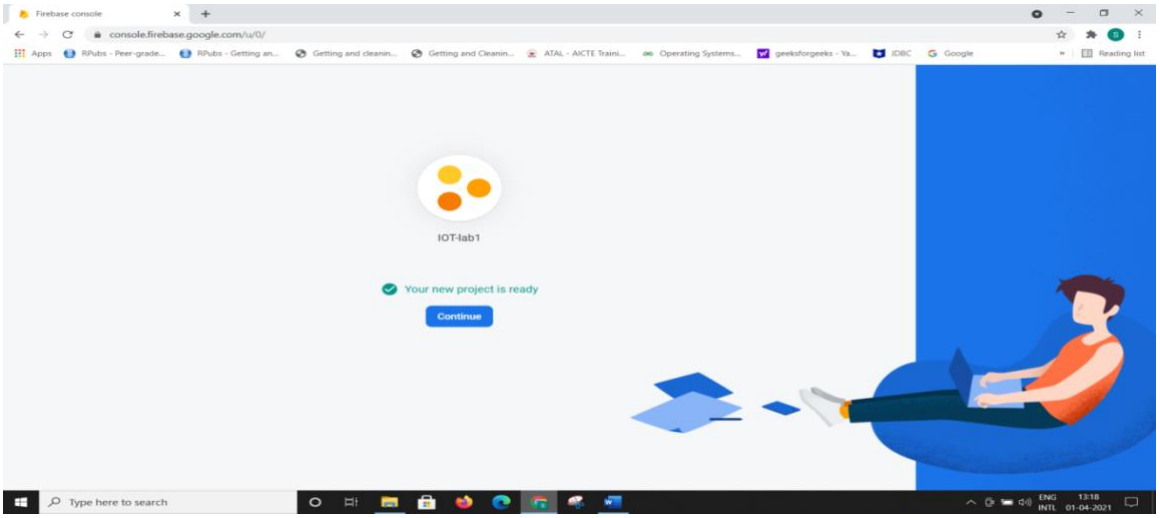
Step5:select the location as INDIA and tick the check box of I accept the terms and finally click on Create Project button.



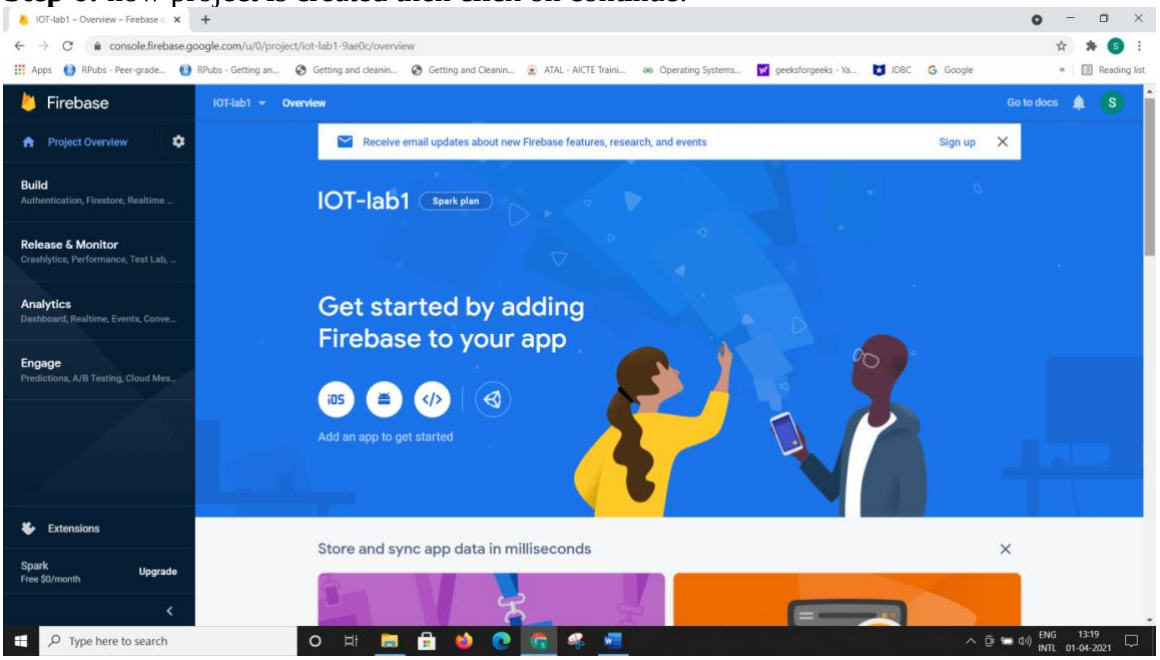

Firebase console
console.firebase.google.com/u/0/
Provisioning resources...
Type here to search


Firebase console
console.firebase.google.com/u/0/
Creating your project...
Type here to search

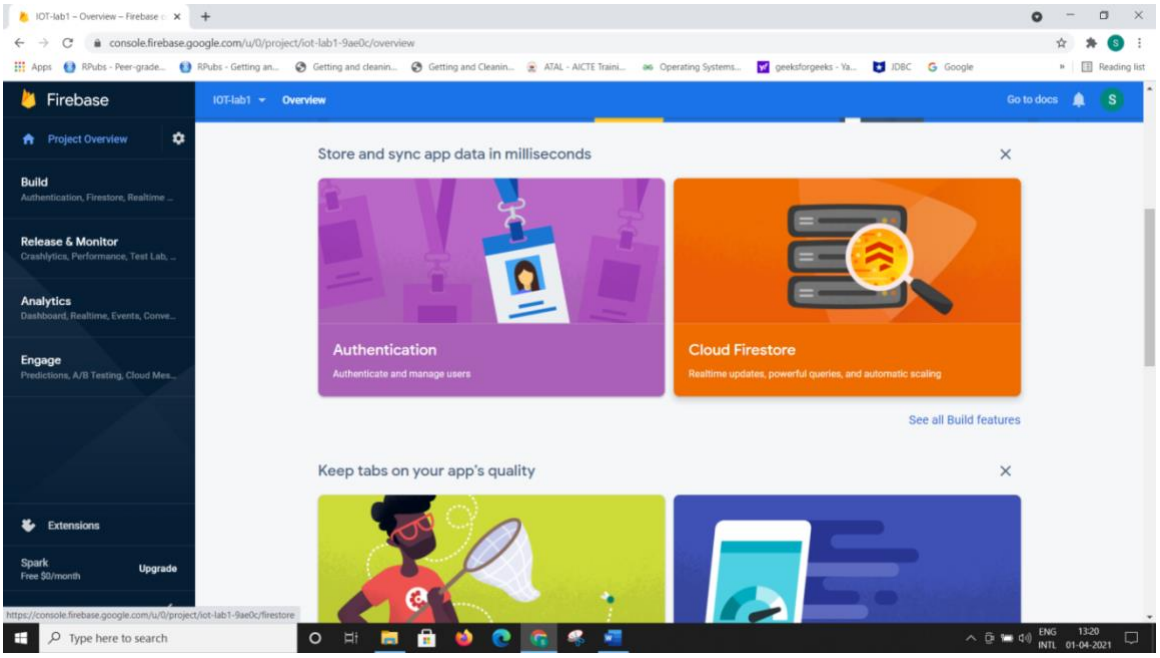

Firebase console
console.firebase.google.com/u/0/
Finishing up...
Type here to search

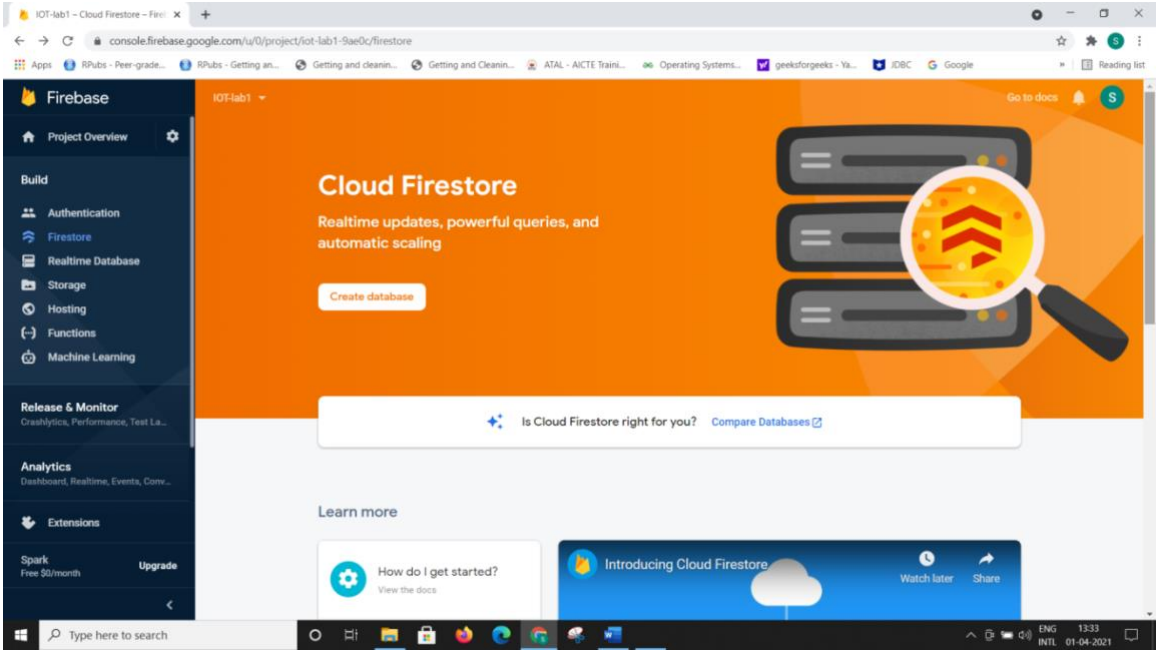


Step 6: now project is created then click on continue.

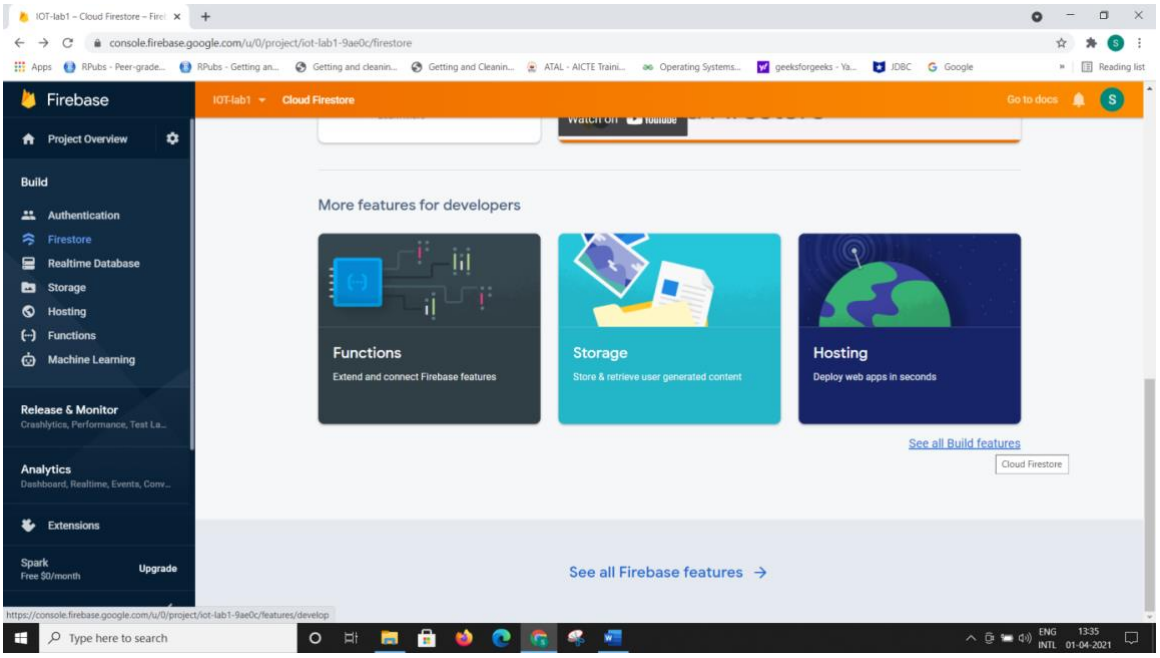


Step 7: Now move down the scrollbar and select cloud firestore

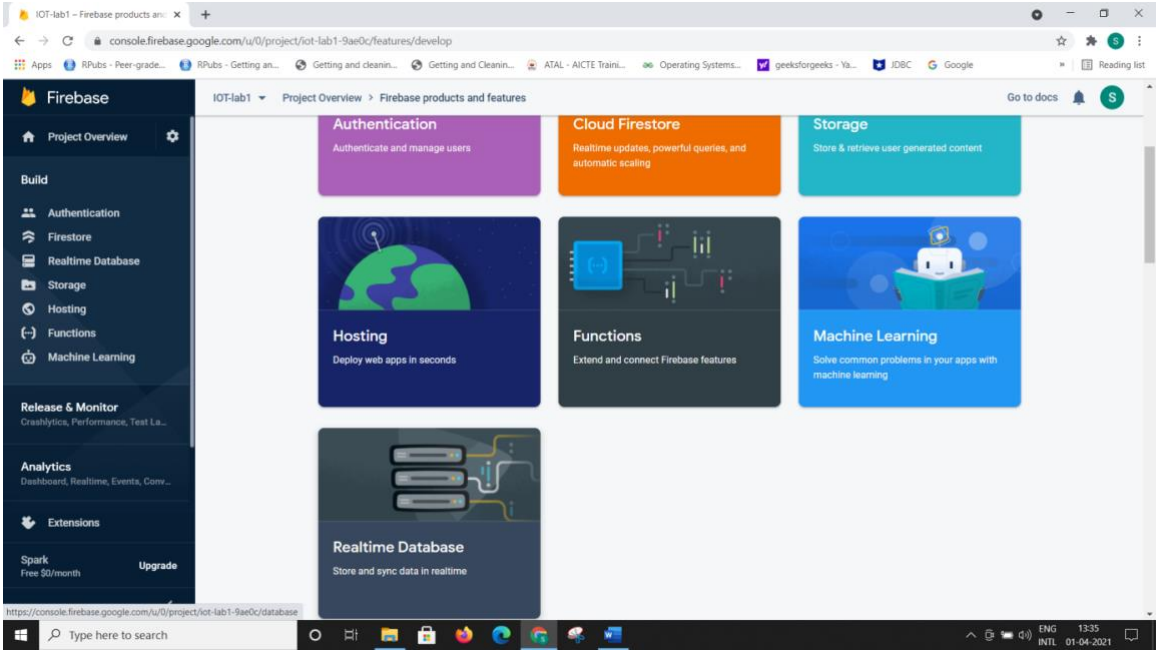




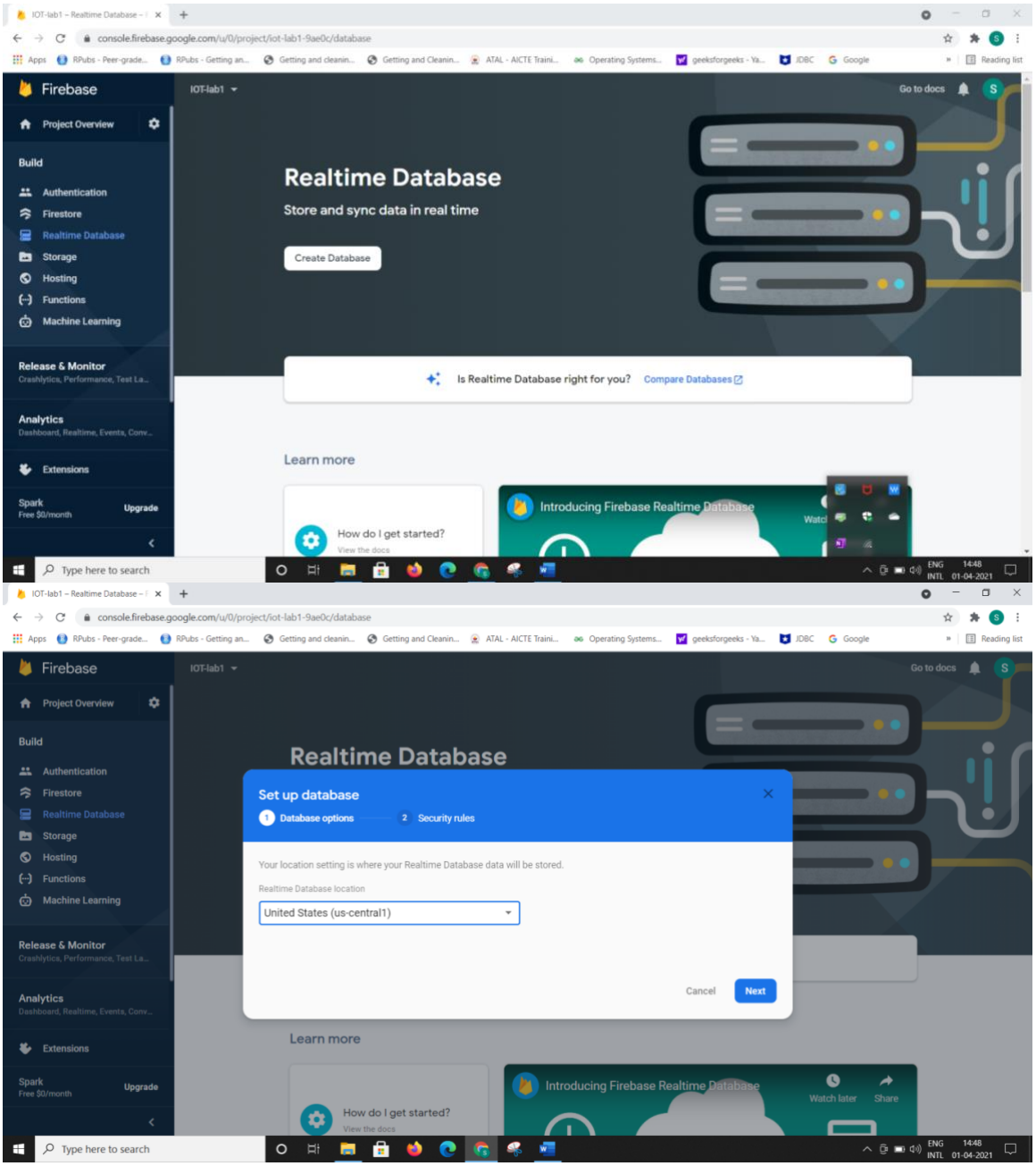
Now scroll down and select see all build features.



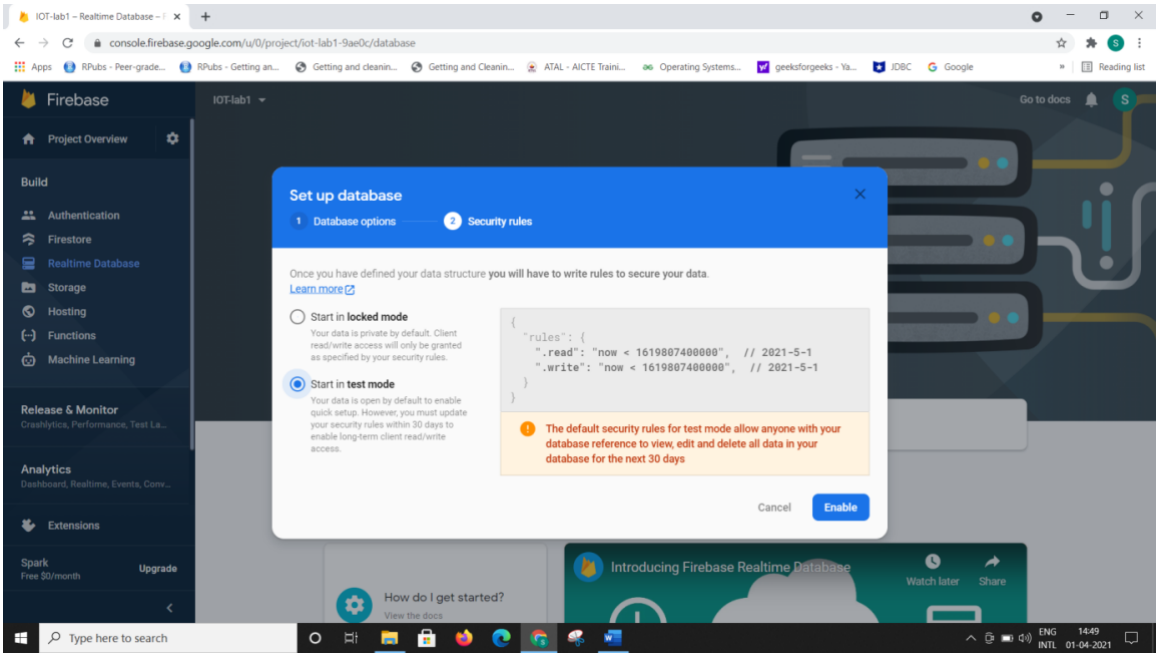
Now select realtime database



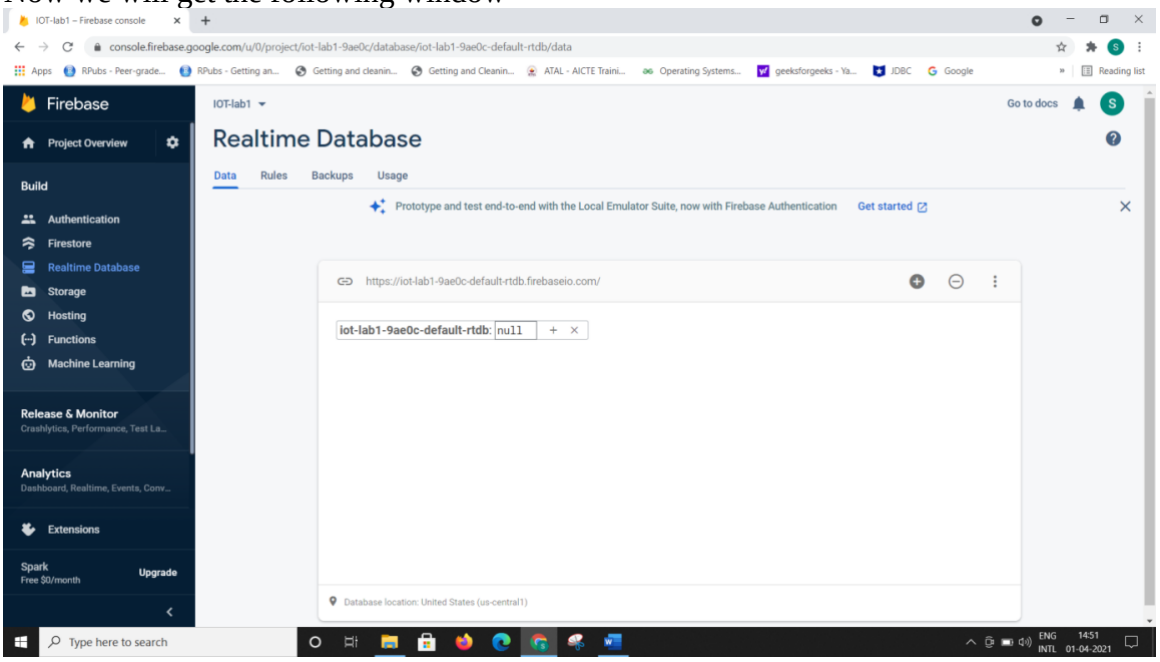
Now click on create database



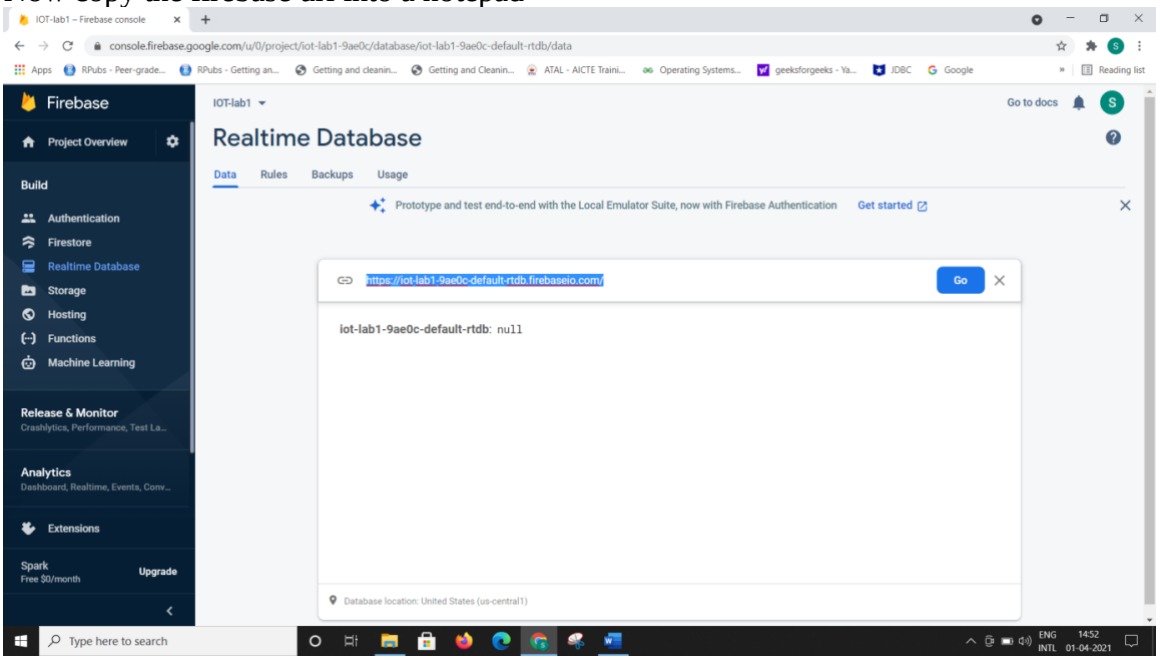
Now select start in test mode radio button and then click on Enable button.



Now we will get the following window

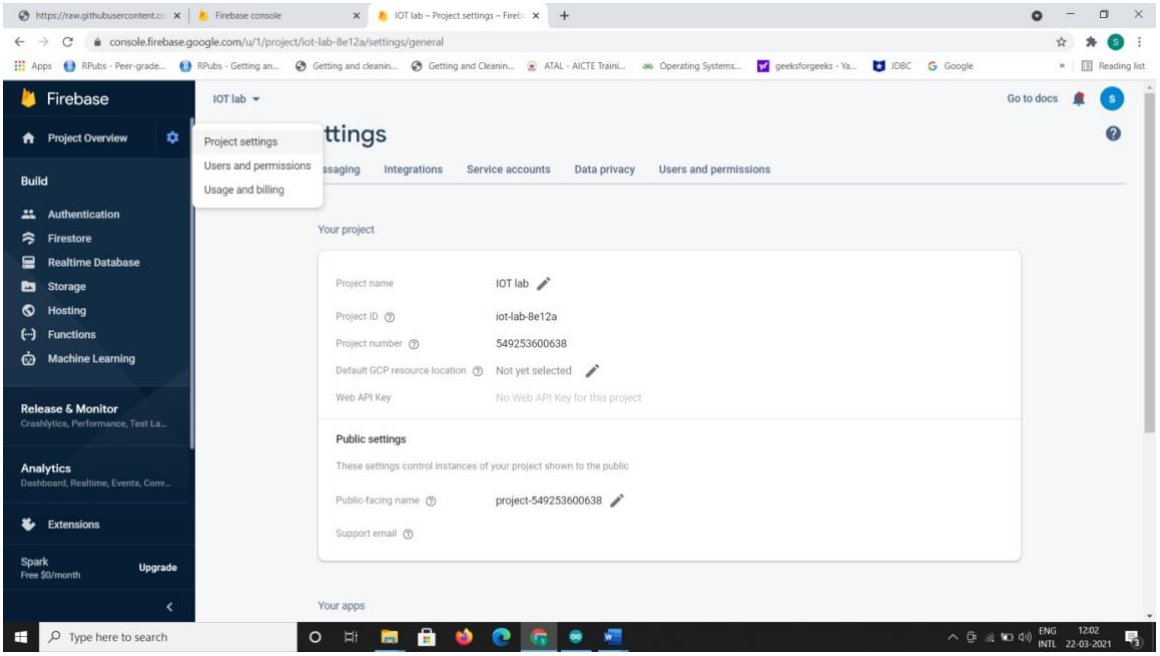


Now copy the firebase url into a notepad

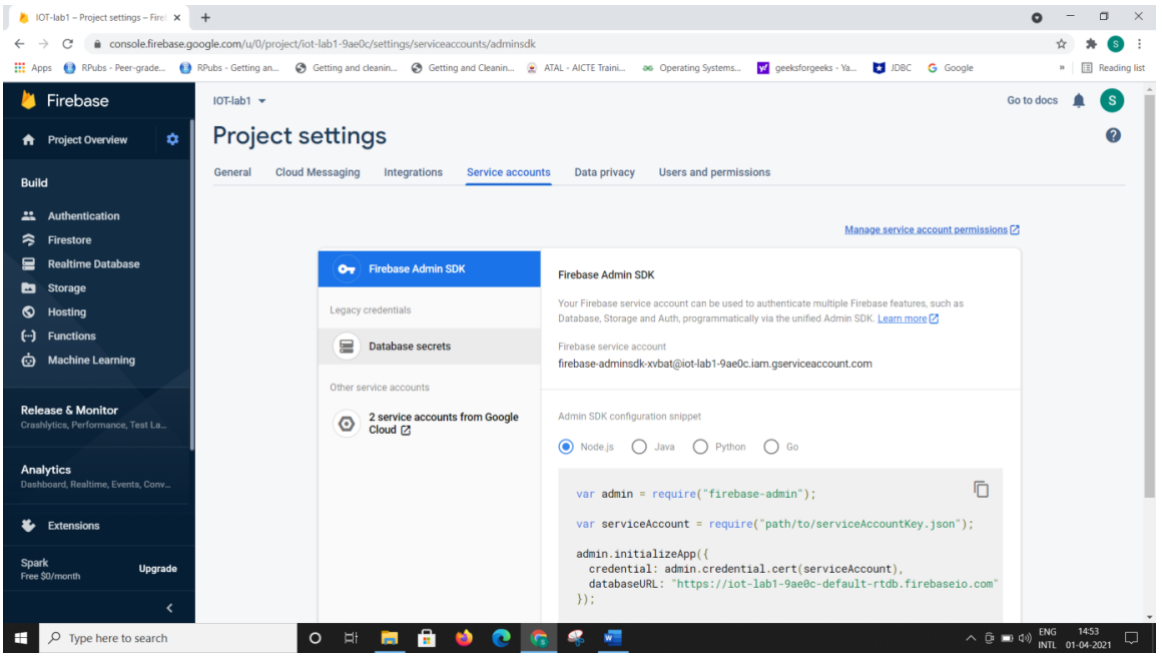


Step8: go to left side pallet and click on settings icon and select project settings.

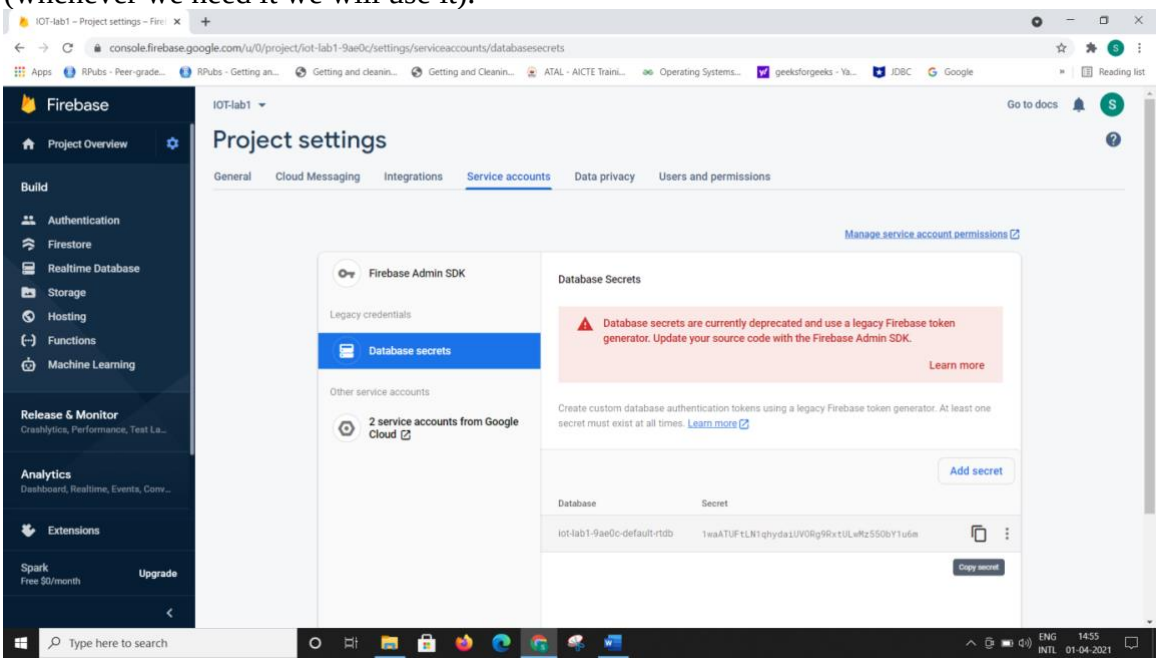
s



Now select service accounts



Click on database secrets which is in the left panel and there we find the secret key copy that key into the notepad (whenever we need it we will use it).



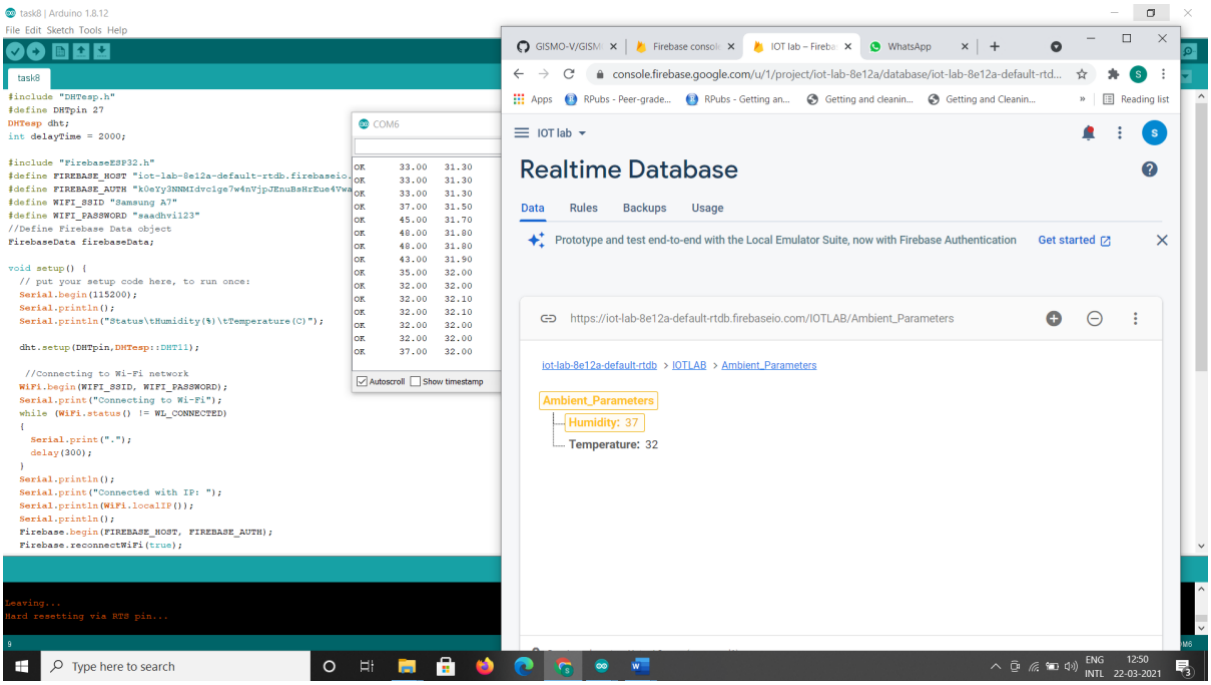
Now we are done with creation of a database in cloud (firebase) to store the data which comes from the board. For example, we will take an Arduino program to sense temperature and humidity using DHT11 sensor on the board.

The following is the program for sensing temperature and humidity

```
#include "DHTesp.h"
#define DHTpin 27
```

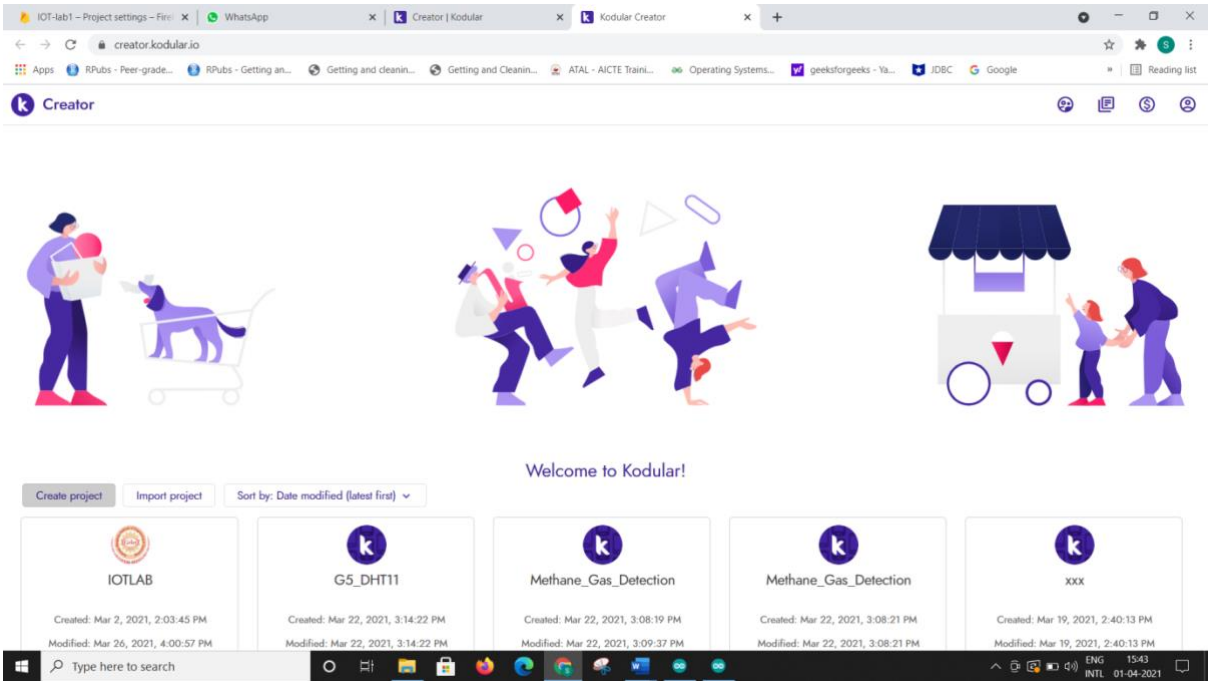
```
DHTespdht;
intdelayTime = 2000;
#include "FirebaseESP32.h"
#define FIREBASE_HOST "xxxxxxxxxxxxxxxxxxxxxxxxxxxx" //Do not include https:// in FIREBASE_HOST
#define FIREBASE_AUTH "xxxxxxxxxxxxxxxxxxxxxxxxxxxx"
#define WIFI_SSID "xxxxxxxxx"
#define WIFI_PASSWORD "xxxxxxxxxx"
//Define Firebase Data object
FirebaseDatafirebaseData;
void setup() {
  // put your setup code here, to run once:
Serial.begin(115200);
Serial.println();
Serial.println("Status\tHumidity(%)\tTemperature(C)");
dht.setup(DHTpin,DHTesp::DHT11);
   //Connecting to Wi-Fi network
WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
Serial.print("Connecting to Wi-Fi");
while (WiFi.status() != WL_CONNECTED)
  {
Serial.print(".");
delay(300);
  }
Serial.println();
Serial.print("Connected with IP: ");
Serial.println(WiFi.localIP());
Serial.println();
Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
Firebase.reconnectWiFi(true);
  }
void loop() {
  // put your main code here, to run repeatedly:
delay(delayTime);
float h = dht.getHumidity();
float t = dht.getTemperature();
Serial.print(dht.getStatusString());
Serial.print("\t");
Serial.print(h);
Firebase.setFloat(firebaseData,"IOTLAB/Ambient_Parameters/Humidity",h);
Serial.print("\t");
Serial.println(t);
Firebase.setFloat(firebaseData,"IOTLAB/Ambient_Parameters/Temperature",t);


}
```

Compile and dump the program on to the board and then you will see the output in serial monitor and the same updated values you will see in realtime database also. The output is as shown in the below picture.

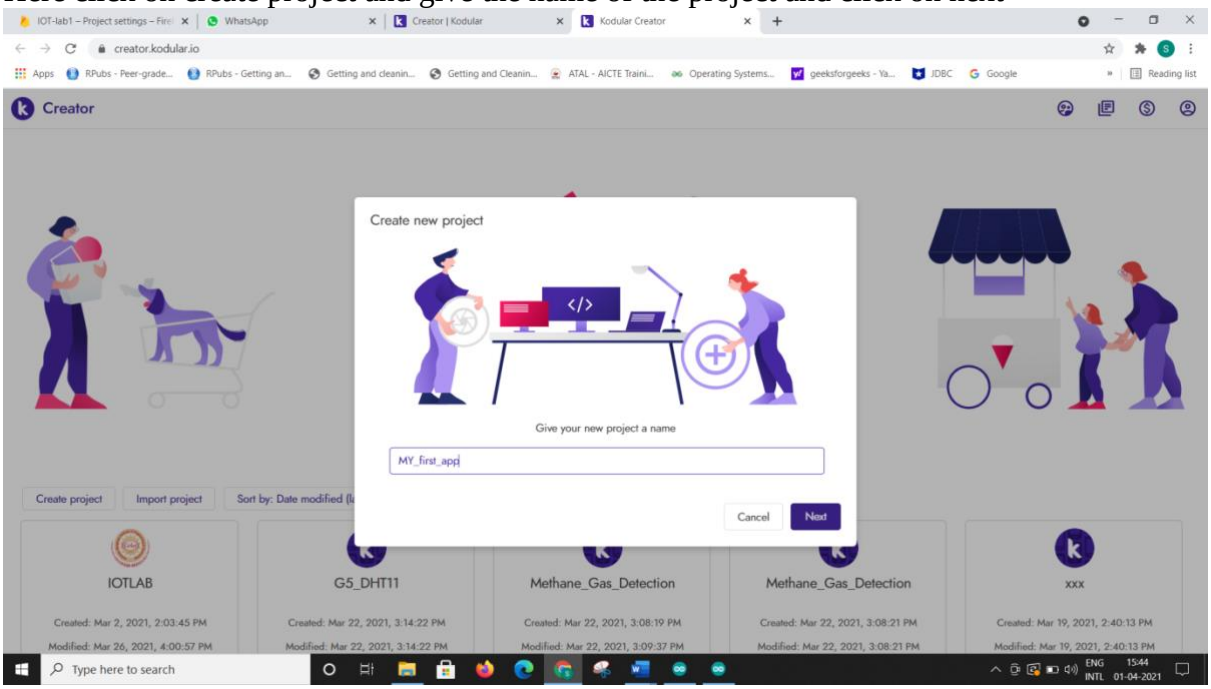


Now let us create a mobile app. We go with kodular creator.
The following steps explains how to develop a mobile app and display the data present in cloud.
Step1: Open kodular creator by typing in google or go to this link https://www.kodular.io/creator

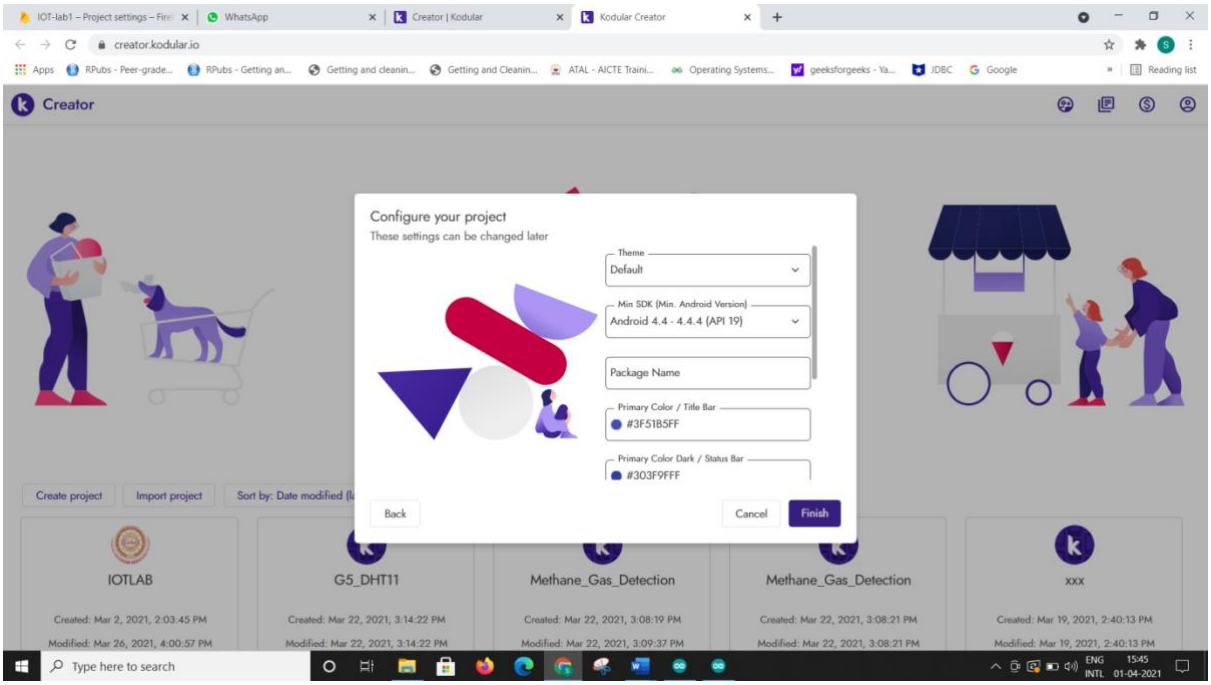
Now click on get started button



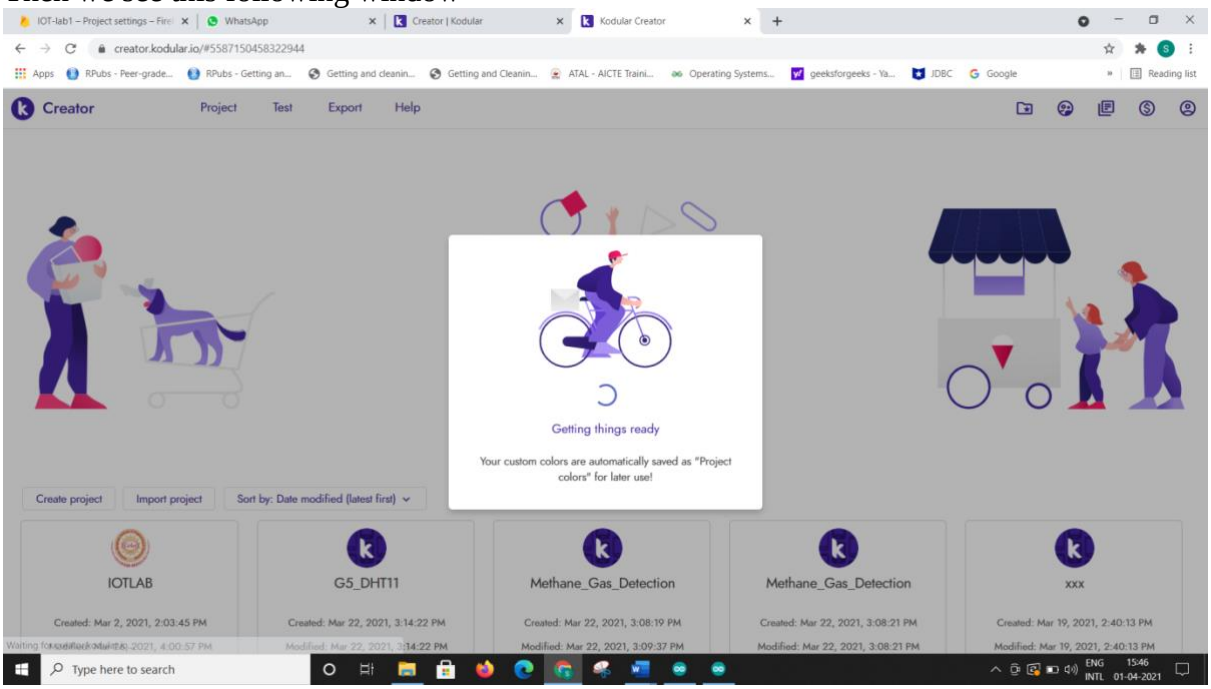
Here click on create project and give the name of the project and click on next



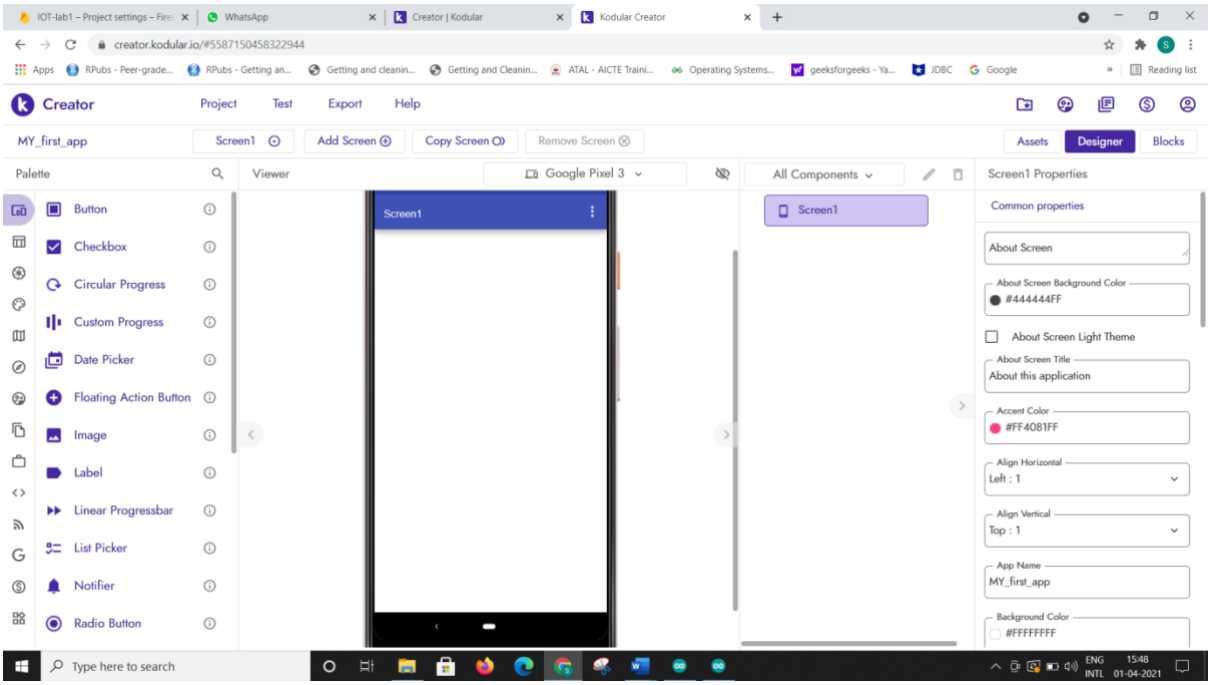
Then we will get this following window then just click on finish.



Then we see this following window



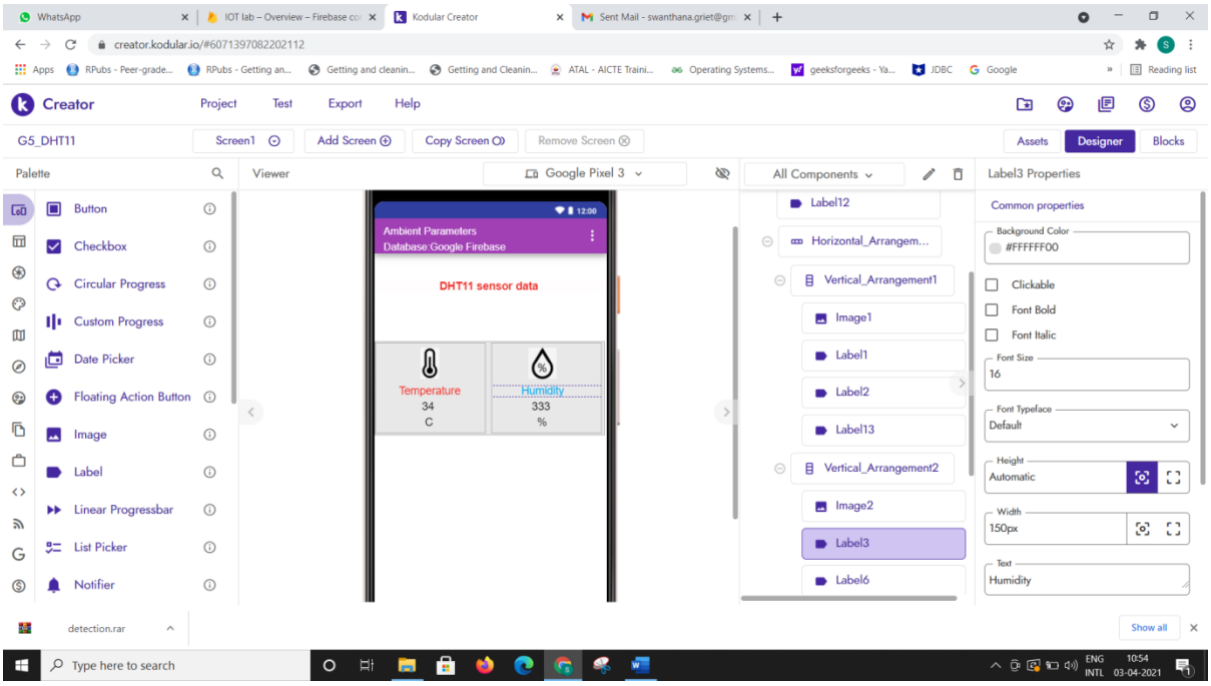
We get this following page to create your mobile app

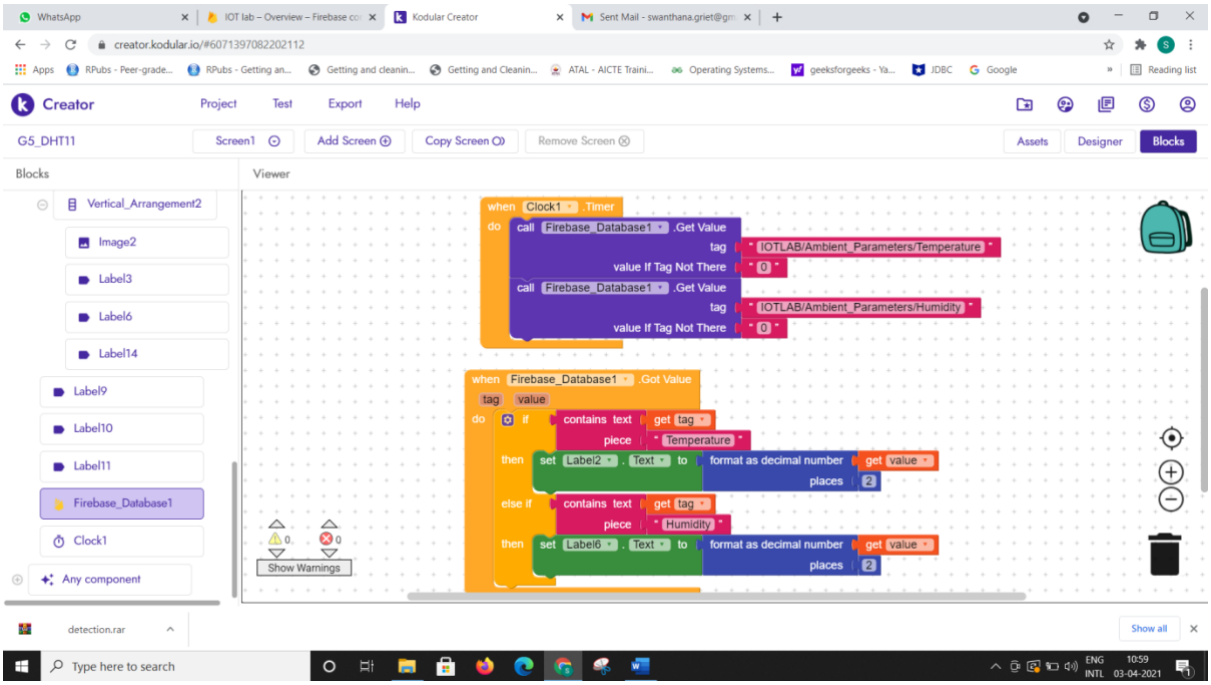


Here on the left side, we will see the Palette where we can find all the GUI widgets, we should just drag and drop the necessary widgets on to the mobile screen visible beside the palette.

Drag and drop the labels and name the text of the labels as Temperature and Humidity and the takemore labels to display the values of temperature and humidity by getting the values from the firebase (database). And also we can use alignments also like vertical or horizontal accordingly to your requirements, you should also drag and drop the firebase and clock from the Palette.

Now design part of the mobile app is done. Now move to Blocks to write the code by clicking Blocks on the right-side top of the window.





Now your mobile app is ready which is connected to firebase to get the ambient parameters that are updated for every clock second from the board.

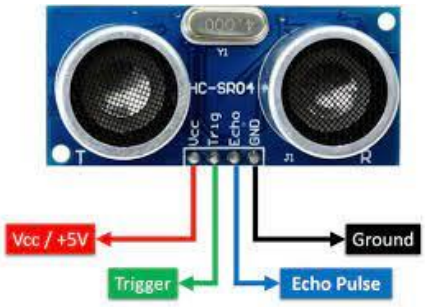
Task 8

(ULTRASONIC SENSOR)

Task 8: Internet enabled remote range indicator.
Aim :Write a program for interfacing theUltrasonic Sensor with Generic Sensor Board
Components Required:GISMO VI Board,Ultrasonic Sensor, Usb cable

PIN-OUT
Echo: Echo output
Trig: Trigger input

Ultrasonic Sensor signal	ESP32 pins
Trigger	GPIO25
Echo	GPIO26
Vcc	5V
GND	GND



Ultrasonic Sensor(HC-SR04)

Ultrasonic Sensor(HC-SR04)
The working principle of this module is simple. It sends an ultrasonic pulse out at 40kHz which travels through the air and if there is an obstacle or object, it will bounce back to the sensor. By calculating the travel time and the speed of sound, the distance can be calculated.

Ultrasonic sensors are a great solution for the detection of clear objects. The sensor head emits an ultrasonic wave and receives the wave reflected back from the target. ultrasonic / level sensors measure the distance to the target by measuring the time between the emission and reception.

For presence detection, ultrasonic sensors detect objects regardless of the color, surface, or material (unless the material is very soft like wool, as it would absorb sound.)

To detect transparent and other items where optical technologies may fail, ultrasonic sensors are a reliable choice.

Sound travels at approximately 340 meters per second. To measure the distance the sound has travelled we use the formula: **Distance = (Time x SpeedOfSound) / 2**. The "2" is in the formula because the sound has to travel back and forth. Every 1 Second equals **1000000 Microsecond.**

Program:

```
// defines pins numbers
constinttrigPin = 25;
constintechoPin = 26;
// defines variables
long duration;
int distance;

void setup()
{
        pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
        pinMode(echoPin, INPUT); // Sets the echoPin as an Input
        Serial.begin(115200); // Starts the serial communication
}
void loop()
 {
        // Clears the trigPin
        digitalWrite(trigPin, LOW);
        delayMicroseconds(2);
        // Sets the trigPin on HIGH state for 10 micro seconds
        digitalWrite(trigPin, HIGH);
        delayMicroseconds(10);
        digitalWrite(trigPin, LOW);
        // Reads the echoPin, returns the sound wave travel time in microseconds
        duration = pulseIn(echoPin, HIGH);
        // Calculating the distance
                distance= duration*0.034/2;
        // Prints the distance on the Serial Monitor
        Serial.print("Distance: ");// in centimeter
        Serial.println(distance);
        delay(1000);//delay is in miliseconds
}
```

OUTPUT:

Task-9

Task-9 Internet enabled smart room lighting system. (Relay using GISMO Board)

Aim: Internet enabled smart room lighting system. (Relay using GISMO Board)

Components Required: GISMO VI Board**,** Relay Sensor, Usb cable

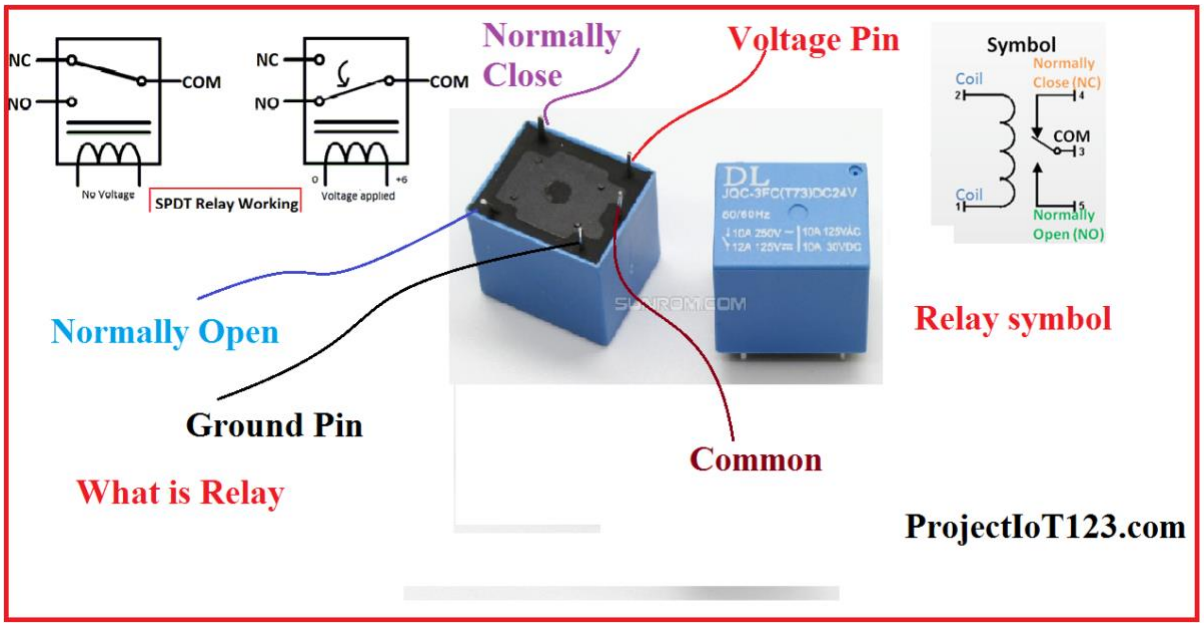
Relay Sensor	ESP32 pins
NC/NO with COM	GPIO13
GND	GND
Vcc	3.3V

Relay Sensor:

Relay works on the principle of **electromagnetic induction**. When the electromagnet is applied with some current it induces a magnetic field around it. ... In the relay Copper coil and the iron core acts as electromagnet. When the coil is applied with DC current it starts attracting the contact as shown.

When an electric **current** crosses the coil a magnetic field is formed over the armature that attracts it and activates the contacts. A control current applied to the relay coil makes possible to open, close or commute the contacts and controls the currents that circulate through external loads.

Relays are **switches that open and close circuits electromechanically or electronically**. Relays control one electrical circuit by opening and closing contacts in another circuit.



Note: Here we will use ultrasonic sensor and relay and buzzer. When any object comes closer to than the distance specified the relay will be HIGH and on the buzzer.

Program:

```
void setup()
{
  // put your setup code here, to run once:
  pinMode(13,OUTPUT);
}
void loop() {
  // put your main code here, to run repeatedly:
 digitalWrite(13,HIGH);
 delay(2000);
 digitalWrite(13,LOW);
 delay(2000);
}
```

Output :Relay will ON and OFF

Task 10

Task-10 Internet enabled Smart Garden Maintainence.

AIM: Internet enabled Smart Garden Maintainence.

Components Required:

PIN-OUT

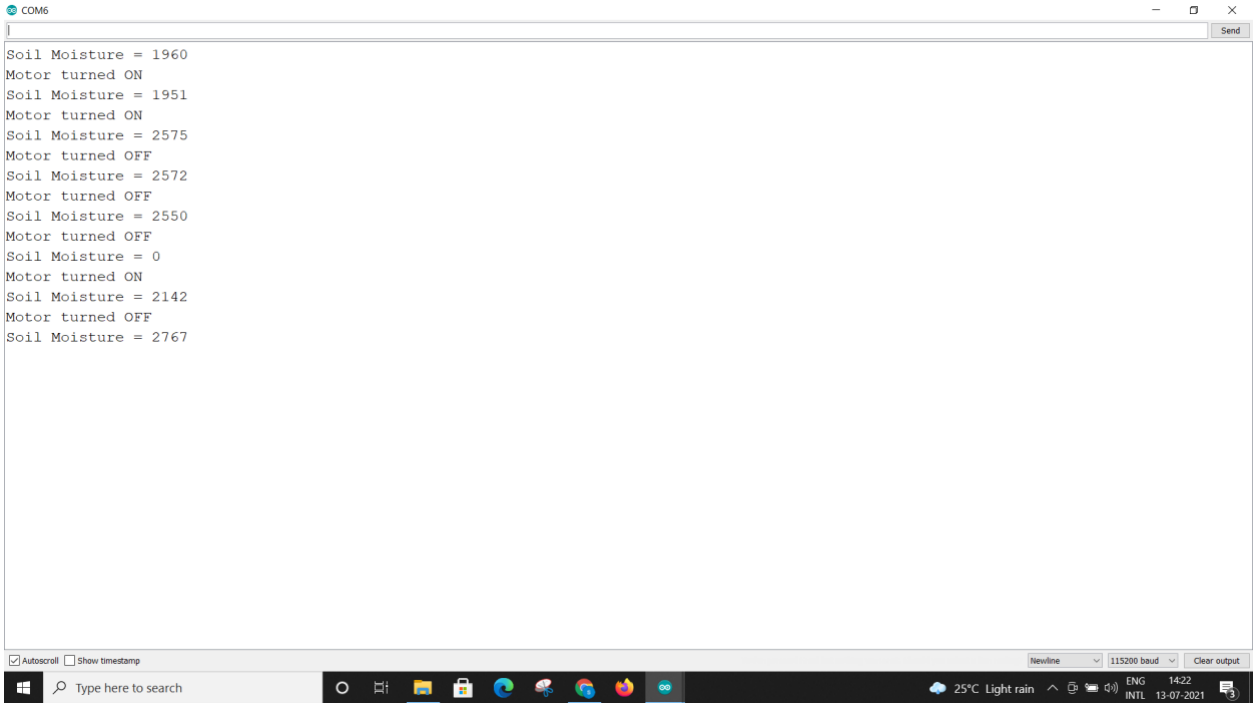
Soil Moisture sensor	ESP32 pins
AO	GPIO34
DO	NC
Vcc	3.3V
GND	GND

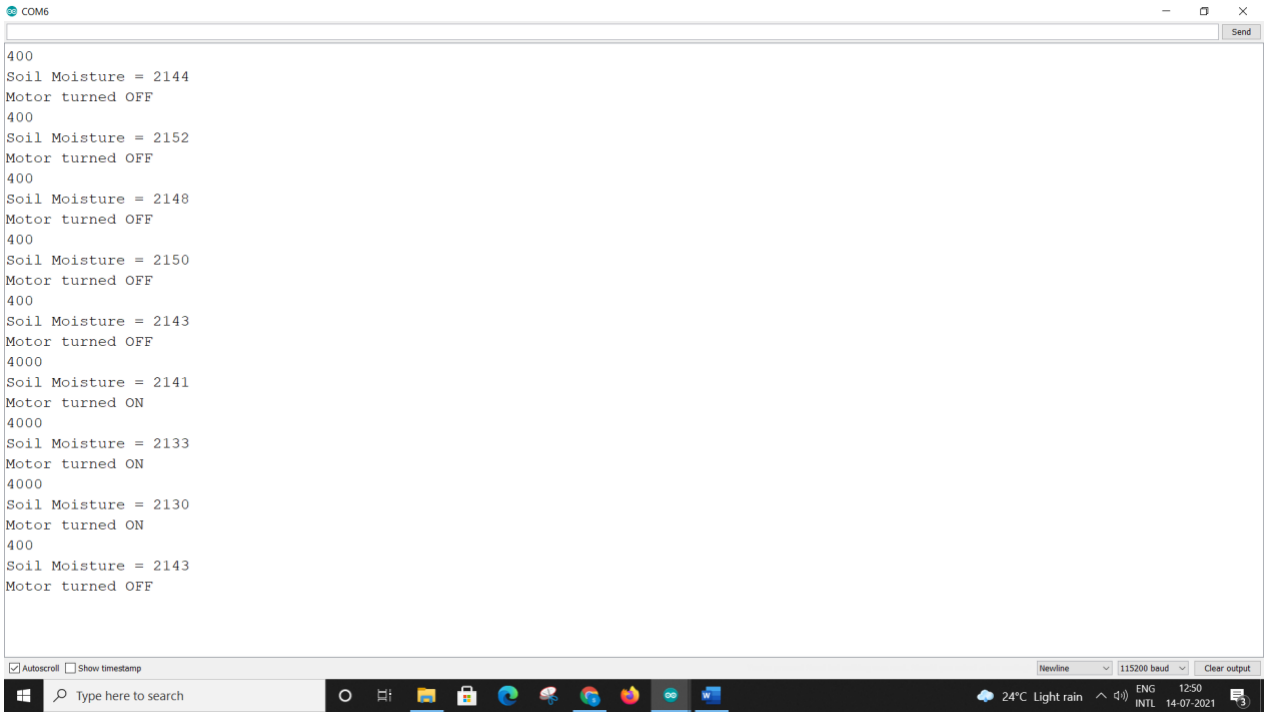
Program:

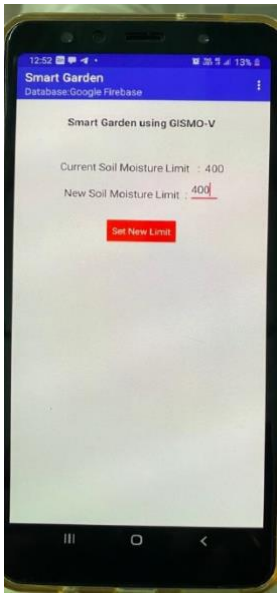
```
/* Soil Moisture sensor pin   : GPIO34
 * Relay pin                 : GPIO13
 *
 */
#include "FirebaseESP32.h"
#define FIREBASE_HOST "xxxxxxxxxxxxxxxxxxxxxxxxxxxxx" //Do not include https:// in FIREBASE_HOST
#define FIREBASE_AUTH "xxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
#define WIFI_SSID "XXXXXX"
#define WIFI_PASSWORD "XXXXXXXX"
//Define Firebase Data object
FirebaseData firebaseData;

#define smPin 34
#define relayPin 13
int smValue;
int smLimit = 3000;
int delayTime = 2000;
String smValueCloud;
void setup() {
  // put your setup code here, to run once:
  pinMode(relayPin,OUTPUT);
  Serial.begin(115200);

  //Connecting to Wi-Fi network
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  Serial.print("Connecting to Wi-Fi");
  while (WiFi.status() != WL_CONNECTED)
  {
    Serial.print(".");
    delay(300);
  }
  Serial.println();
```

```
  Serial.print("Connected with IP: ");
  Serial.println(WiFi.localIP());
  Serial.println();
  Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
  Firebase.reconnectWiFi(true);
  }

void loop() {
  // put your main code here, to run repeatedly:
String smCloudFull;
String smCloud;
if(Firebase.getString(firebaseData,"xxx/IOTLAB/Smart_Garden/SM_Threshold",smCloudFull)){
smCloud = smCloudFull.substring(2,smCloudFull.length()-2);
smLimit = smCloud.toInt();
Serial.println(smLimit);
}
smValue = 4095 - analogRead(smPin);
Serial.print("Soil Moisture = ");
Serial.println(smValue);
Firebase.setInt(firebaseData,"IOTLAB/Smart_Garden/Soil_Moisture",smValue);

if (smValue < smLimit){
digitalWrite(relayPin,HIGH);
Serial.println("Motor turned ON");
Firebase.setString(firebaseData,"IOTLAB/Smart_Garden/Motor_Status","ON");
}
else{
digitalWrite(relayPin,LOW);
Serial.println("Motor turned OFF");

Firebase.setString(firebaseData,"IOTLAB/Smart_Garden/Motor_Status","OFF");
}

delay(delayTime);
}
```

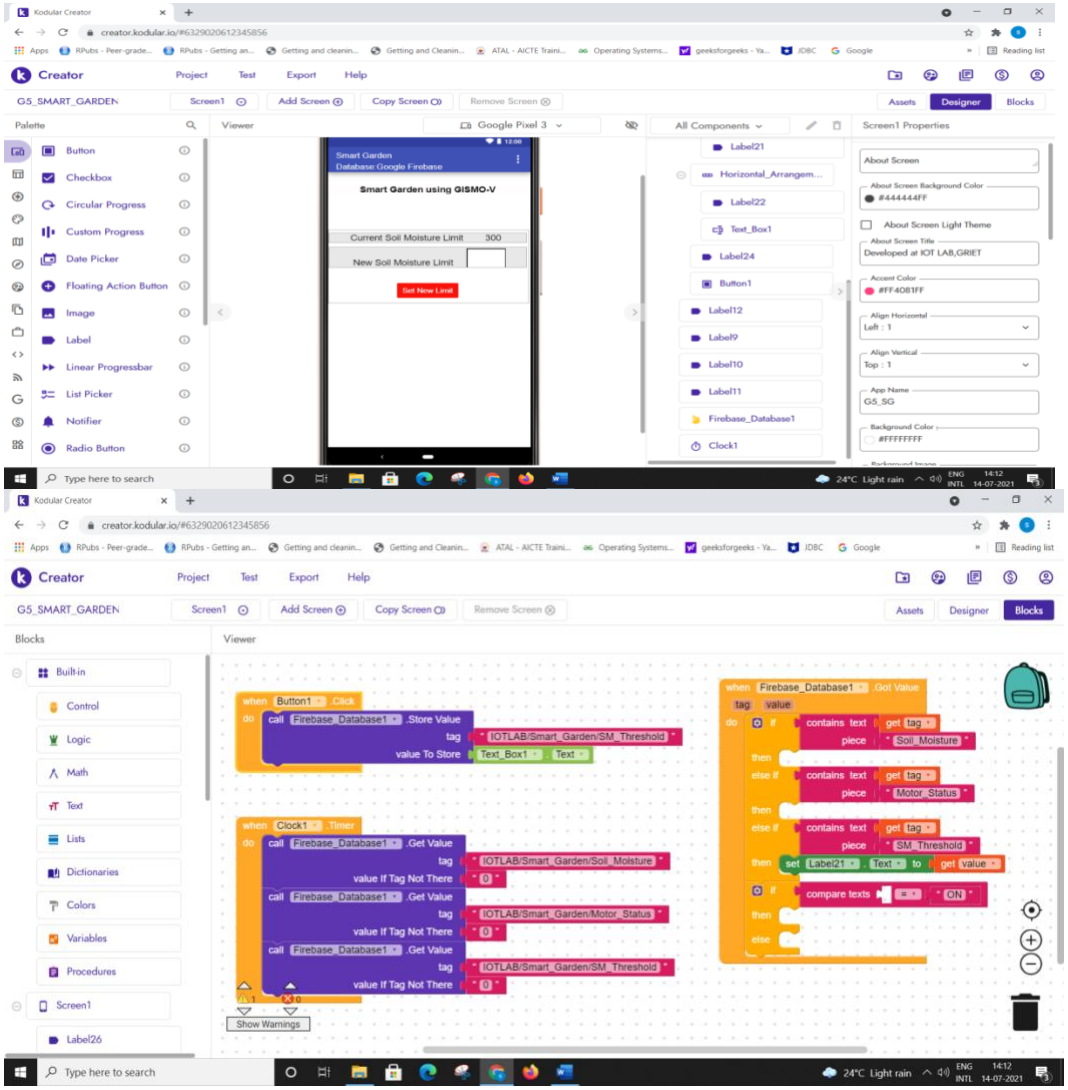
OUTPUT:


COM6
Send
Soil Moisture = 1960
Motor turned ON
Soil Moisture = 1951
Motor turned ON
Soil Moisture = 2575
Motor turned OFF
Soil Moisture = 2572
Motor turned OFF
Soil Moisture = 2550
Motor turned OFF
Soil Moisture = 0
Motor turned ON
Soil Moisture = 2142
Motor turned OFF
Soil Moisture = 2767
Autoscroll
Show timestamp
Newline
115200 baud
Clear output
Type here to search
25°C Light rain
ENG
INTL
14:22
13-07-2021


COM6
Send
400
Soil Moisture = 2144
Motor turned OFF
400
Soil Moisture = 2152
Motor turned OFF
400
Soil Moisture = 2148
Motor turned OFF
400
Soil Moisture = 2150
Motor turned OFF
400
Soil Moisture = 2143
Motor turned OFF
4000
Soil Moisture = 2141
Motor turned ON
4000
Soil Moisture = 2133
Motor turned ON
4000
Soil Moisture = 2130
Motor turned ON
400
Soil Moisture = 2143
Motor turned OFF
Autoscroll
Show timestamp
Newline
115200 baud
Clear output
Type here to search
24°C Light rain
ENG
INTL
12:50
14-07-2021



Here using a mobile app we can set the Soil Moisture limit, depending upon the seasons we can change the limit of the Soil Moisture.



Task=11

Task-11 Internet enabled Display of Ambient Prameter(BMP).

Aim: Internet enabled Display of Ambient Prameter(BMP).

Components: Required: GISMO VI ,BMP Sensor,Usb Cable

Pin-Out

DHT11 sensor	ESP32 pins
Data-pin	GPIO27
Vcc	3.3V
GND	GND

Functions:

1. dht.setup(DHTpin,DHTesp::DHT11)

DHT11(Temperature & Humidity sensor)

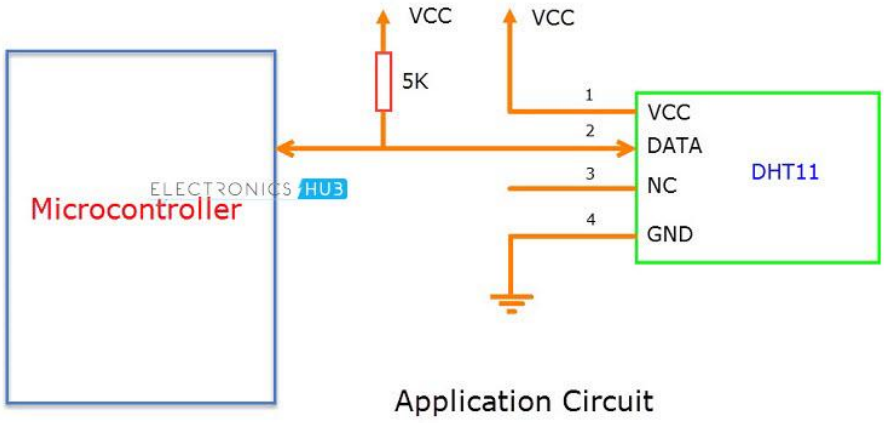
A **distributed hash table** (DHT) is a distributed system that provides a lookup service similar to a hash table: key-value pairs are stored in a DHT, and any participating node can efficiently retrieve the value associated with a given key.

The DHT11 is a **basic, ultra low-cost digital temperature** and humidity sensor. It uses a capacitive humidity sensor and a thermistor to measure the surrounding air, and spits out a digital signal on the data pin (no analog input pins needed). Its fairly simple to use, but requires careful timing to grab data. It can measure humidity from 20% to 90% RH (**relative humidity** (RH) is a measure of the water vapor content of air) and temperature from 0 to 50 degrees Celsius.

Working Principle of DHT11 Sensor

The **humidity sensing capacitor has two electrodes with a moisture holding substrate as a dielectric between them**. Change in the capacitance value occurs with the change in humidity levels. The IC measure, process this changed resistance values and change them into digital form.

DHT11 Output: Serial data. Temperature Range: 0°C to 50°C. Humidity Range: **20% to 90%.**



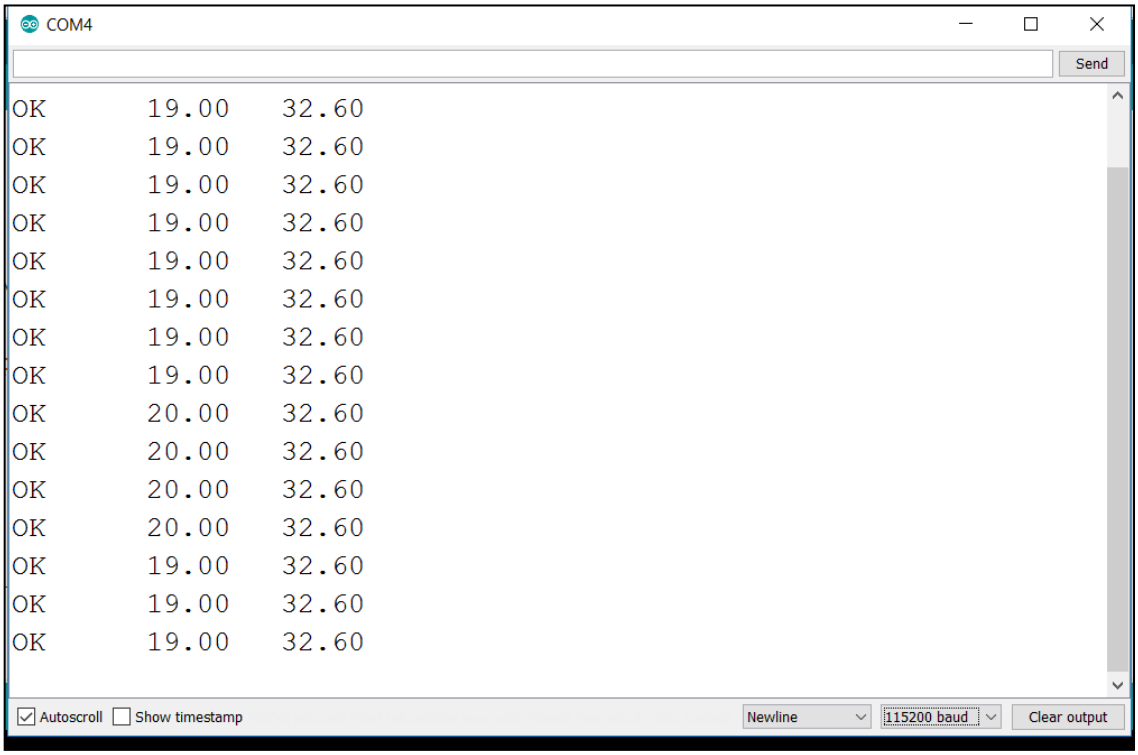
Application Circuit

Program:

```
#include "DHTesp.h"
```

```
#define DHTpin 27
DHTesp dht;
void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);
  Serial.println();
  Serial.println("Status\tHumidity(%)\tTemperature(C)");
  dht.setup(DHTpin,DHTesp::DHT11);
}

void loop() {
  // put your main code here, to run repeatedly:
 delay(2000);
  float h = dht.getHumidity();
  float t = dht.getTemperature();
  Serial.print(dht.getStatusString());
  Serial.print("\t");
  Serial.print(h);
  Serial.print("\t");
  Serial.println(t);
  }
```

OUTPUT:



Task-12

Task-12 Internet Enabled Home Safety and security system.

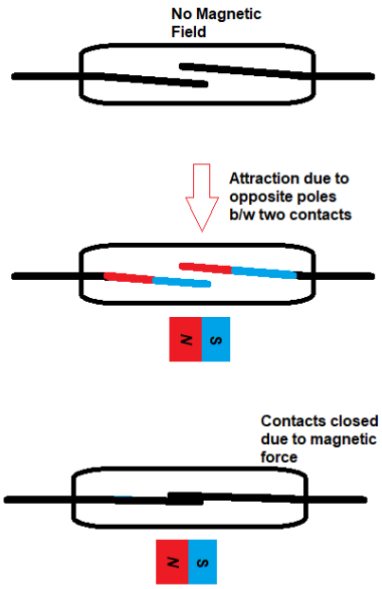
Aim: Internet Enabled Home Safety and security system.
Components Required:GISMO VI,Magnetic switch ,Usb Cable.

PIN-OUT

Magnetic Sensor	ESP32 pins
Sensor o/p contact1	GPIO16
Sensor o/p contac2	GND
Vcc	3.3V

Magnetic sensors **detect moving ferrous metal**. The simplest magnetic sensor consists of a wire coiled around a permanent magnet. A ferrous object approaching the sensor changes magnetic flux through the coil, generating a voltage at the coil terminals. ... Magnetic sensors measure speeds up to 600,000 rpm.

Magnetic Switch: The switching mechanism is comprised of **two ferromagnetic blades, separated by only a few microns**. When a magnet approaches these blades, the two blades pull toward one another. Once touching, the blades close the normally open (NO) contacts, allowing electricity to flow.
The use of INPUT_PULLUP with pinMode(). It **monitors the state of a switch by establishing serial communication between your Arduino and your computer over USB**.



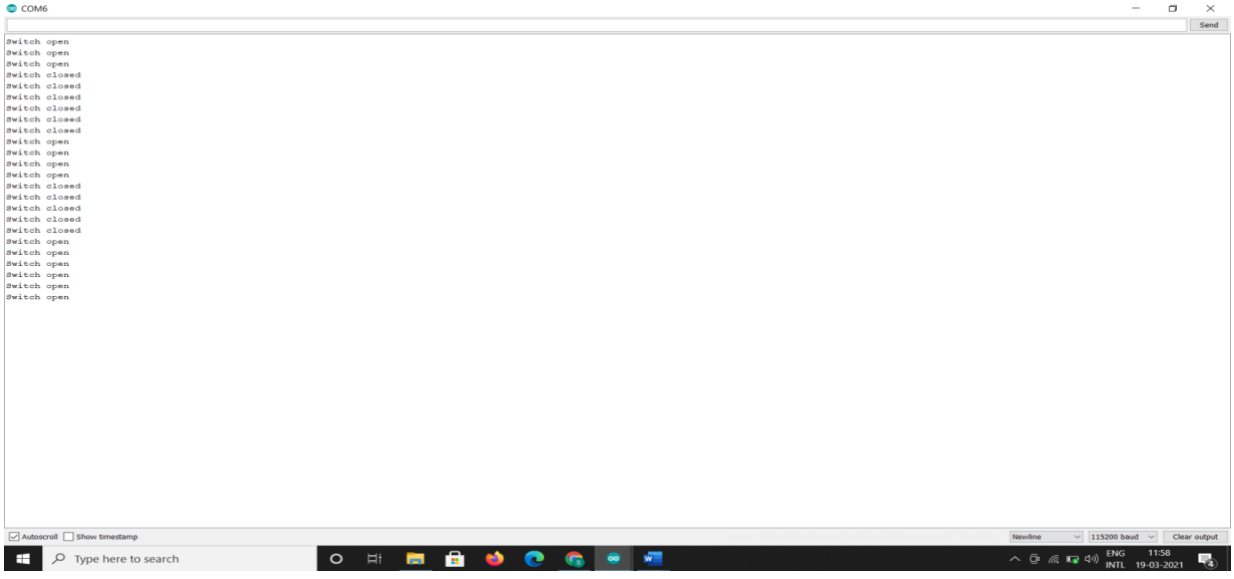
The INPUT_PULLUP also does **the same function as INPUT** however only differing in the aspect of base

voltage, where the INPUT argument pulls down the voltage of that port to 0V every time there is no voltage is detected across the port while the INPUT_PULLUP argument pulls up the voltage across the port to the maximum .

Program:

```
#define magSW 16
void setup()
 {
pinMode(magSW,INPUT_PULLUP);
Serial.begin(115200);
}

void loop()
 {
  int sw_status;
sw_status = digitalRead(magSW);
  if (sw_status)
Serial.println("Switch open");
  else
Serial.println("Switch closed");
delay(1000);
}
```

OUTPUT:



IOT based project ideas

1. Arduino/Raspberry Pi/ Node MCU based LED street lights with auto intensity control.
2. Arduino/Raspberry Pi/ Node MCU operated obstacle avoidance robot
3. Arduino/Raspberry Pi/ Node MCU based electrical appliance control using IR
4. Arduino/Raspberry Pi/ Node MCU based Solar street light
5. Remote controlled car using Arduino/Raspberry Pi/ Node MCU
6. DJ lights effect with respect to intensity of music using Arduino/Raspberry Pi/ Node MCU
7. Water level detector using Arduino/Raspberry Pi/ Node MCU
8. Mood lamp using Arduino/Raspberry Pi/ Node MCU
9. Controlling LEDs using remote control
10. Wireless transmission using rain sensor
11. Digital clock using Arduino
12. Controlling LED & Motor using Web Server.
13. Arduino/Raspberry Pi/ Node MCU based LED Dice
14. Automatic fan controller using Arduino/Raspberry Pi/ Node MCU.
15. Arduino Stop Watch.
16. Smart farming using Arduino/Raspberry Pi/ Node MCU.
17. Tone pitch follower using Arduino/Raspberry Pi/ Node MCU.
18. Battery tester using Arduino/Raspberry Pi/ Node MCU.
19. Arduino/Raspberry Pi/ Node MCU based LED bar graph code
20. Smart Garbage Collector.
21. LED Cube using Arduino/Raspberry Pi/ Node MCU
22. Arduino/Raspberry Pi/ Node MCU based vehicle parking counter
23. Arduino/Raspberry Pi/ Node MCU based RGB based door clock

24. Secret knock detecting door lock using Arduino/Raspberry Pi/ Node MCU.
25. Traffic light controller using Arduino/Raspberry Pi/ Node MCU.
26. Alcohol sensing display with alarm using Arduino/Raspberry Pi/ Node MCU.
27. Motion based automatic door opener using Arduino/Raspberry Pi/ Node MCU.
28. Automated plant watering system
29. Bridge Health Monitoring System
30. Wireless hand gesture controlled robotic arm.
31. Arduino/Raspberry Pi/ Node MCU based health care kit.
32. RFID attendance system.
33. Hand gesture controlled laptop using Arduino/Raspberry Pi/ Node MCU.
34. Home automation using voice control.
35. Face Recognition using Raspberry Pi.
36. Raspberry Pi vehicle number plate recognition.
37. Camera based surveillance system using raspberry Pi.
38. Internet enabled smoke alarm using node mcu.
39. Path following vehicle.
40. Alcohol detection system.
41. Accessing devices using hand gesture.
42. Smart alert system for monitoring cardiac patients.
43. Revitalizing solar energy.
44. Baby monitoring system.
45. SNAG evasion Automation.
46. IOT Pill bottle.
47. Automatic fan regulator control using temperature.
48. Water quality monitoring system.

49. Gas leakage detection.
50. GSM based home security alaram system.
51. Surveillance robot control through mobile app.
52. Smart parking monitoring in shopping malls.