Ex No:1 Date:	Assembly Language Programs

Aim:

The purpose of this experiment is to write and implement 8051 assembly language experiments using simulator.

Software Required:

Edsim51 simulator

Program Logic:

An Assembly language program consists of, among other things, a series of line of Assemble language instructions. An Assembly language instruction consists of mnemonic, optionally followed by one or two operands. The operands are the data items being manipulated and the mnemonics are the commands to the CPU, telling it what to do with those items.

1. **Looping:**

Repeating a sequence of instructions a certain number of time is called a loop. The loop is one of the most widely used actions that any microprocessor performs. In this instruction, the register is decremented; if it is not zero, it jumps to the target address referred by the label. Prior to the start of the loop the register isloaded with the counter for the number of repetitions. In this instruction both the register decrement and the decision to jump are combined into a single instruction.

A loop inside another loop is called nested loop.

2. **Unconditional jump instruction:**

The unconditional jump is a jump in which control is transferred unconditionally to the target location. In 8051 there is two unconditional jumps:

- **LJMP(long jump):**

LJMP is an unconditional long jump. It is a 3-byte instruction in which the first byte is a opcode, secondand third bytes represent the 16-bit address of the target location. The 2-byte target address allows a jump toany memory location from 0000 to FFFFH.

- **SYMP(Short jump):**

SJMP is an unconditional short jump. It is a 2-byte instruction in which the first byte is a opcode and second bye is the relative address of the target location. The target address ranges from 00 to FFH.

Procedure:

Step-1: Open the Edsim51 simulator.

Step-2: Type the program and save it with ".asm" or ".hex" extension.

Step-3: Run the program and rectify the result.

Step-4: Observe the output.

Step-5: Repeate the above steps for all the program.

Program:

1.Sample of Assembly language program:

```
        ORG 0H              ;start (origin) at location 0
        MOV R5, #25H        ;load 25H into R5
        MOV R7,#34H         ;load 34H into R7
        MOV A, #0           ;load0 into A
        ADD A, R5           ;add contents of R5 to A
                            ;now A = A + R5
        ADD A, R7           ;add contents of R7 to A
                            ;now A = A + R7
        ADD A, #12H         ;add to A value 12H
                            ;now A = A + 12H
HERE:   SJMP HERE           ;Stay in this loop
        END                 ;end of asm source file
```

2.Multipy 25 by 10 using the technique of repeated addition:

```
        MOV A, #0           ;A=0, clear ACC
        MOV R2, #10         ;the multiplier is placed in R2
AGAIN:  ADD A, #25          ;add the multiplicand to the ACC
        DJNZ R2, AGAIN      ;repeat until R2=0 (10 times)
        MOV R5,A            ;save A in R5 ;R5-FAH
```

3.Program to add first 10 natural number:

```
        MOV A, 0            ;A0,clear Acc
        MOV R2, #10         ;load counter value in R2
        MOV R0, #0          ;initialize R0 to zero
AGAIN:  INC R0              ;increment R0 to hold the natural numbers
        ADD A,R0            ;add first number to ACC
        DJNZ R2, AGAIN      ;repeat until R2=0(10 times)
        MOV 46H, A          ;save the result (37H) in RAM location 46H
```

4.Program to load the accumulator with the value of 55H and complement the ACC 700 times:

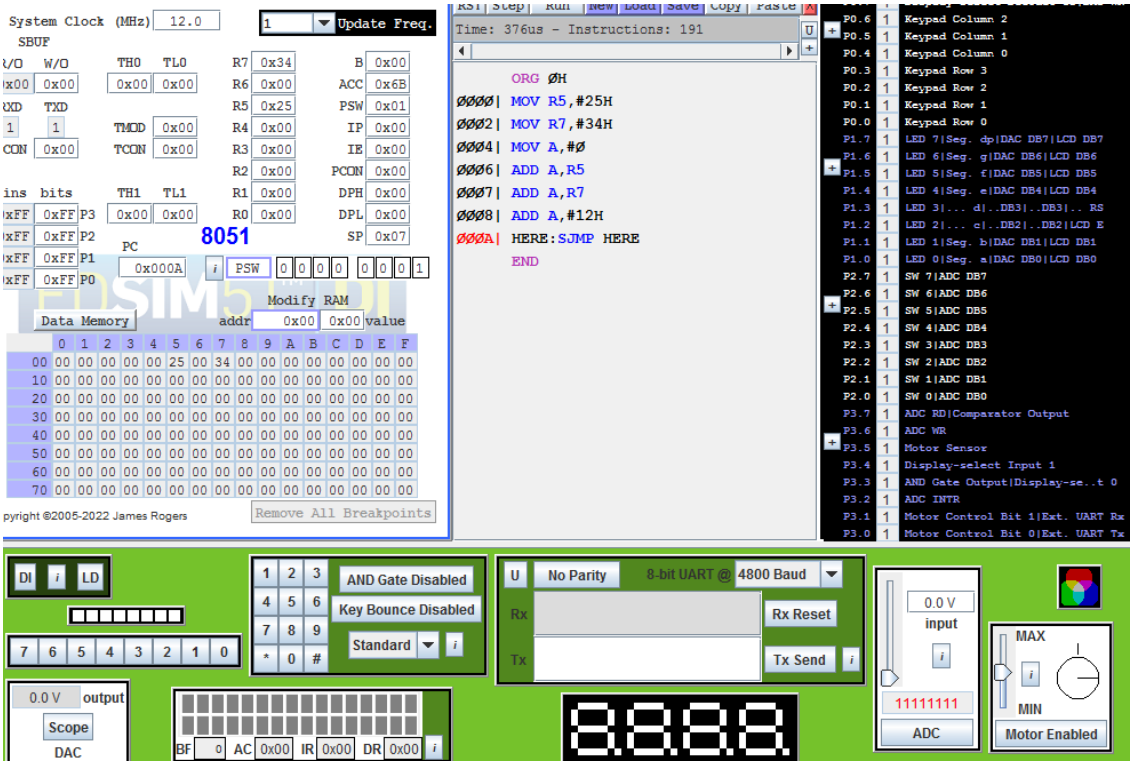
```
        MOV A,#55H          ;A=55H
        MOV R3,#10          ;R3=10, the outer loop count
NEXT:   MOV R2,#70          ;R2=70,the inner loop count
AGAIN:  CPL A               ;complement A register
        DJNZ R2,AGAIN       ;repeat it 70 times(inner loop)
        DJNZ R3,NEXT
```

5.Multiply ECH by 25H using the technique of repeated addition:

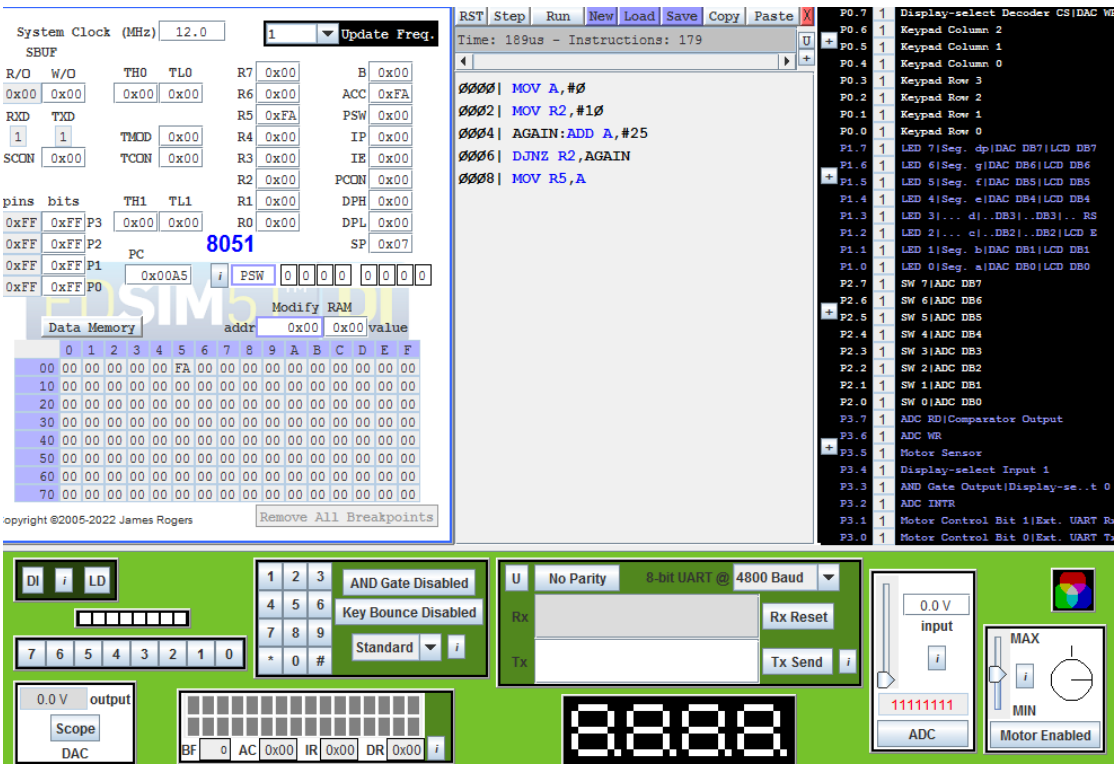
```
        MOV R1,#0          ;R1=0,this 1s the register to store the MSB
        MOV A,#0           ;clear ACC
        MOV R0,#25H        ;the multiplier is placed in R0
AGAIN:  ADD A,#0ECH        ;add the multiplicand to the Acc
        JNC HERE           ;if no carry, then repeat the addition
        INC R1             ;increment R1 for each carry generated
HERE:   DJNZ R0,AGAIN      ;repeat until R0=0
        MOV R0,A           ;the LSB of the product is moved to R0
                           ;the MSB of the product is in R1
                           ; now R1=22H and R0=1CH
```

Output:

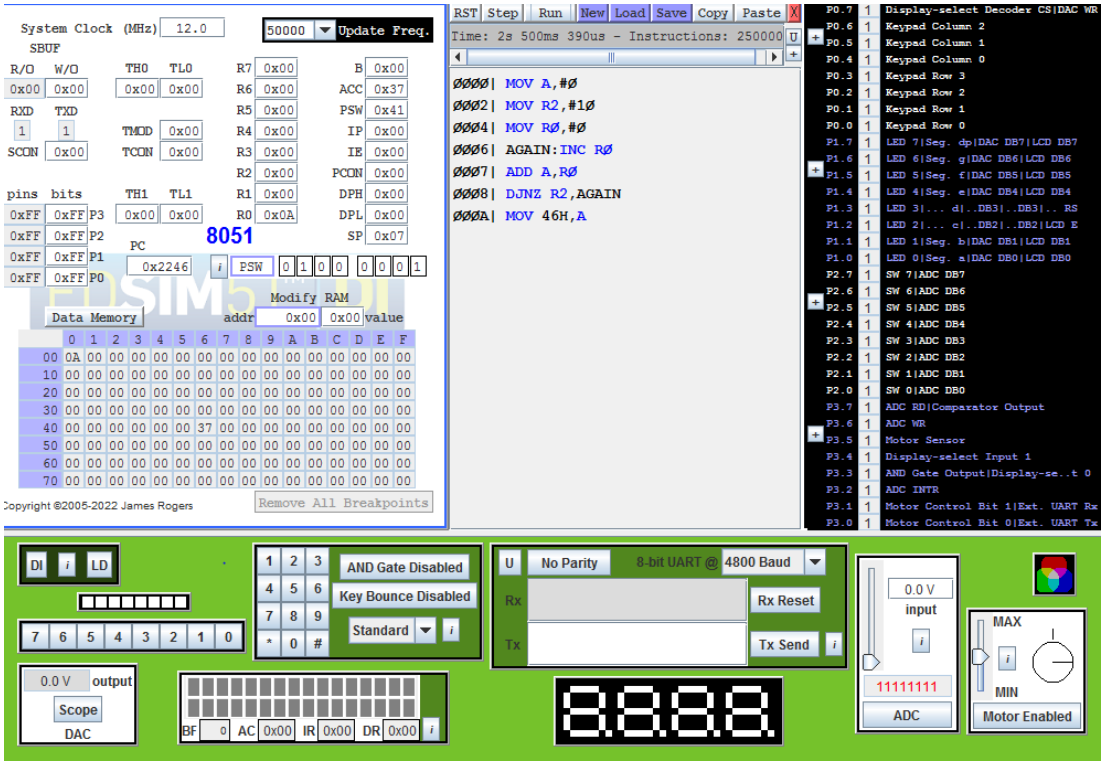
1.Sample of Assembly language program:



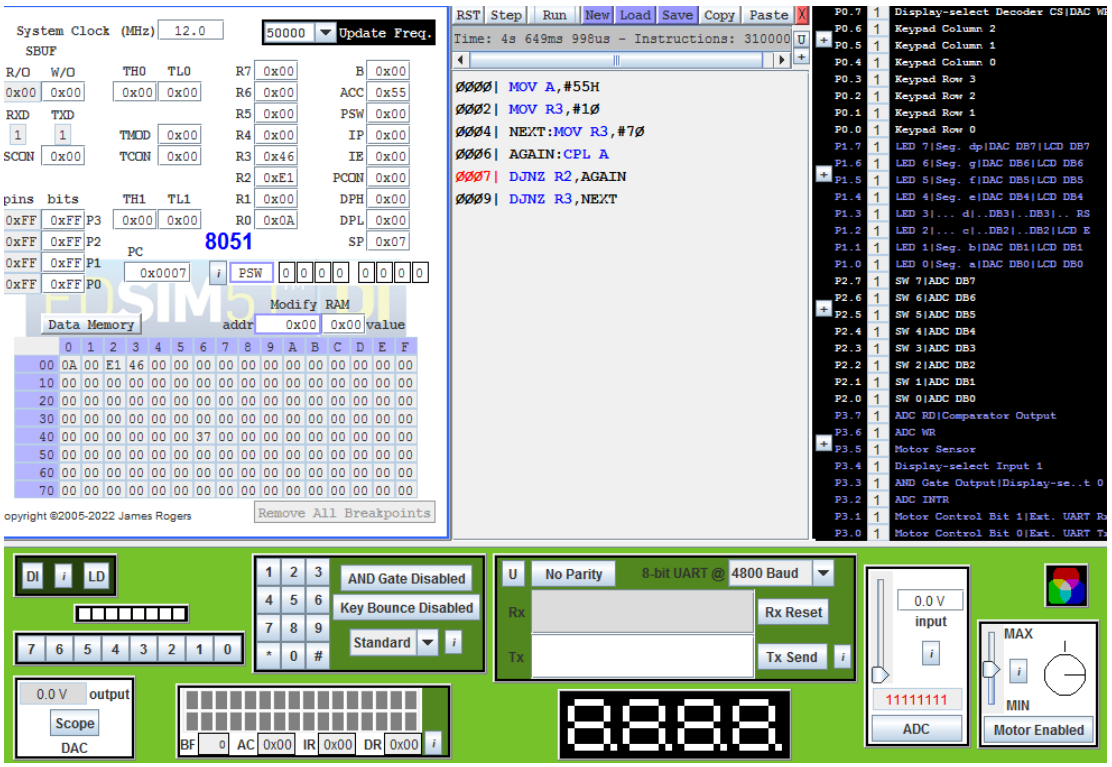
2.Multipy 25 by 10 using the technique of repeated addition:



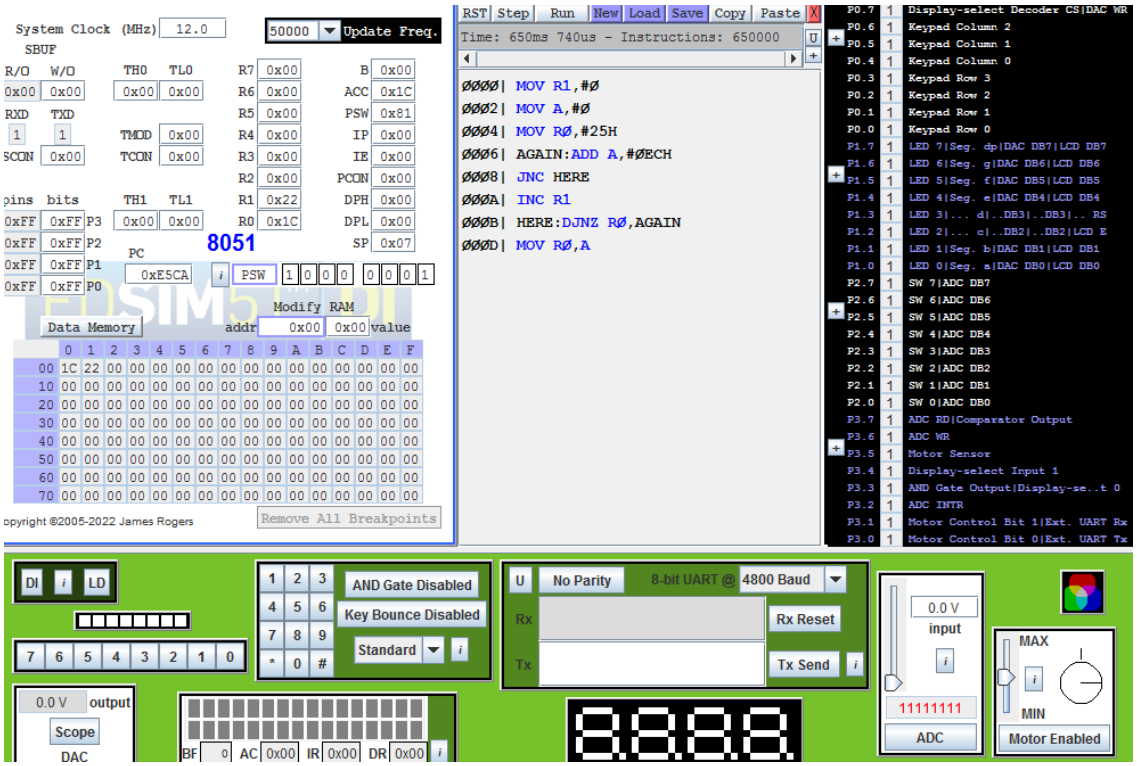
3.Program to add first 10 natural number:



4.Program to load the accumulator with the value of 55H and complement the ACC 700 times:



5.Multiply ECH by 25H using the technique of repeated addition:



Result:

Thus the 8051 assembly level program was written and implemented successfully

Ex No:2	To Text Data Transfer Between Register and Memory
Date:	

Aim:

The purpose of this experiment is to text data transfer between register and memory.

Software Required:

Edsim51 simulator.

Program Logic:

In 8051, the action of comparing and jumping are combined into single instruction called CJNE. The CJNE instruction compares two operands and jump if they are not equal. In addition, it changes the CY flagto indicate if the destination operand is larger or smaller. It is important to notice that the operand themselves remains unchanged.

Procedure:

Step-1: Enter the opcode in Edsim51 simulator.

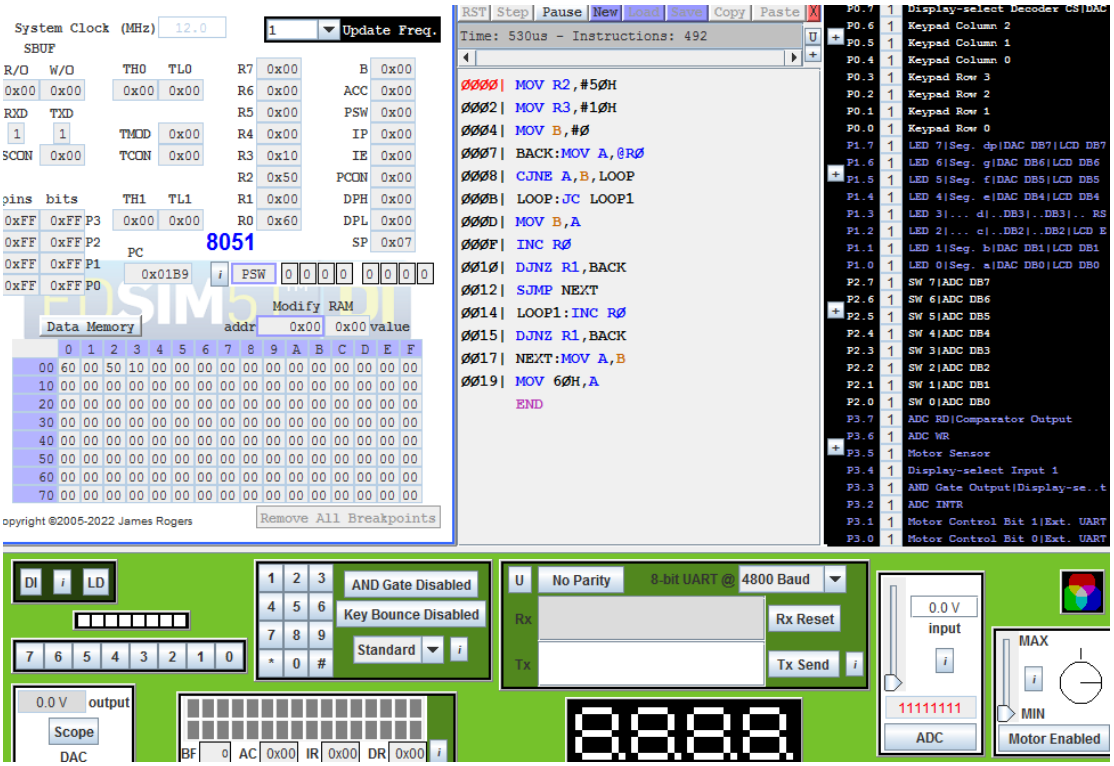
Step-2: Execute the program.

Step-3: Check the result in register A.

Program:

```
        MOV R0,#50H     ;R0 is the pointer to the data
        MOV R1,#10H     ;R1 is the counter
        MOV B,#0        ;B=0
BACK:   MOV A,@R0       ;move a number to A
        CJNE A,B,LOOP   ;compare with B
LOOP:   JC LOOP1        ;if A<B, jump to loop1
        MOV B,A         ;if A>B, move it to B ie., the biggest number should be in B
        INC R0          ;increment the pointer
        DJNZ R1,BACK    ;repeat until the counter=0
        SJMP NEXT       ;jump to EXIT, the biggest number is in B
LOOP1:  INC R0          ;this is another loop, taken when the biggest number was already in buffer
                        ;as comparison
        DJNZ R1,BACK    ;repeat until the counter=0
NEXT:   MOV A,B         ;transfer the biggest number to A register
        MOV 60H,A       ;transfer the result to ROM location 6011
        END             ;end of asm source file
```

Output:



Result:

Thus the program to text data transfer between Register and Memory was written and executed successfully.

Ex No:3	Perform ALU operation using 8051 microcontroller
Date:	

Aim:

The purpose of this experiment is to add, subtract, multiply and divide the given two 8 bit numbersand store them in a memory location.

Software Requirement:

Edsim 51 Simulator.

Program Logic:

To perform addition in 8051 one of the data should be in accumulator, another data can be in any of the general purpose register or in memory or immediate data. After addition the sum will be in accumulator.The sum of two 8-bit data can be either 8-bits(sum only) or 9-bits(sum and carry). The accumulator can accumulate only the sum and there is a carry the 8051 will indicate by setting carry flag. Hence one of the register is used to account for carry.

The 8051 has MUL instruction unlike many other 8-bit processors. MUL instruction multiplies the unsigned 8-bit integers in A and B. The lower order byte of the product is left in A and the higher order bytein B.

The 8051 has DIV instruction unlike many other 8-bit processors. DIV instruction divides the unsigned 8-bit integers in A and B. The accumulator receives the integer part of the quotient and the registerB receives t he remainder.

Procedure:

Step-1: Enter the opcodes from memory location 4200.

Step-2: Execute the program.

Step-3: Check for the result at 4100 and 4101.Using the accumulator, subtraction is performed and the result is stored. Immediate addressing is employed. The SUBB instruction drives the result in the accumulator.

Program:

1. ADDITION:

```
CLR C           ;make CY=0
MOV A,#45H      ;load the low byte into A
ADD A,#0ECH     ;add the low byte ,now A=31,CY=1
MOV R0,A        ;Save the low byte of sum in R0
MOV A,#02H      ;load the high byte into A
ADD A,#0FCH     ;add the high bytes with carry
MOV R1,A        ;save the high byte of result n R1
```

2. SUBTRACTION:

i) Subtraction with CY=0:

```
CLR C           ;make CY=0
MOV A,#3FH      ;load 3FH into A(A=3FH)
MOV R3,#23H     ;load 23H into R3(R3=23H)
SUBB A,R3       ;subtract A-R3,place the result in A
```

ii) Subtraction with CY=1:

```
MOV A,62H       ;A=62H
SUBB A,#96H     ;62H-96H=CCH with CY=1
MOV R7,A        ;save the result
MOV A,#27H      ;A=27H
SUBB A,#12H     ;27H-12H-1=14H
MOV R6,A        ;save the result
```

3) MULTIPLICATION:

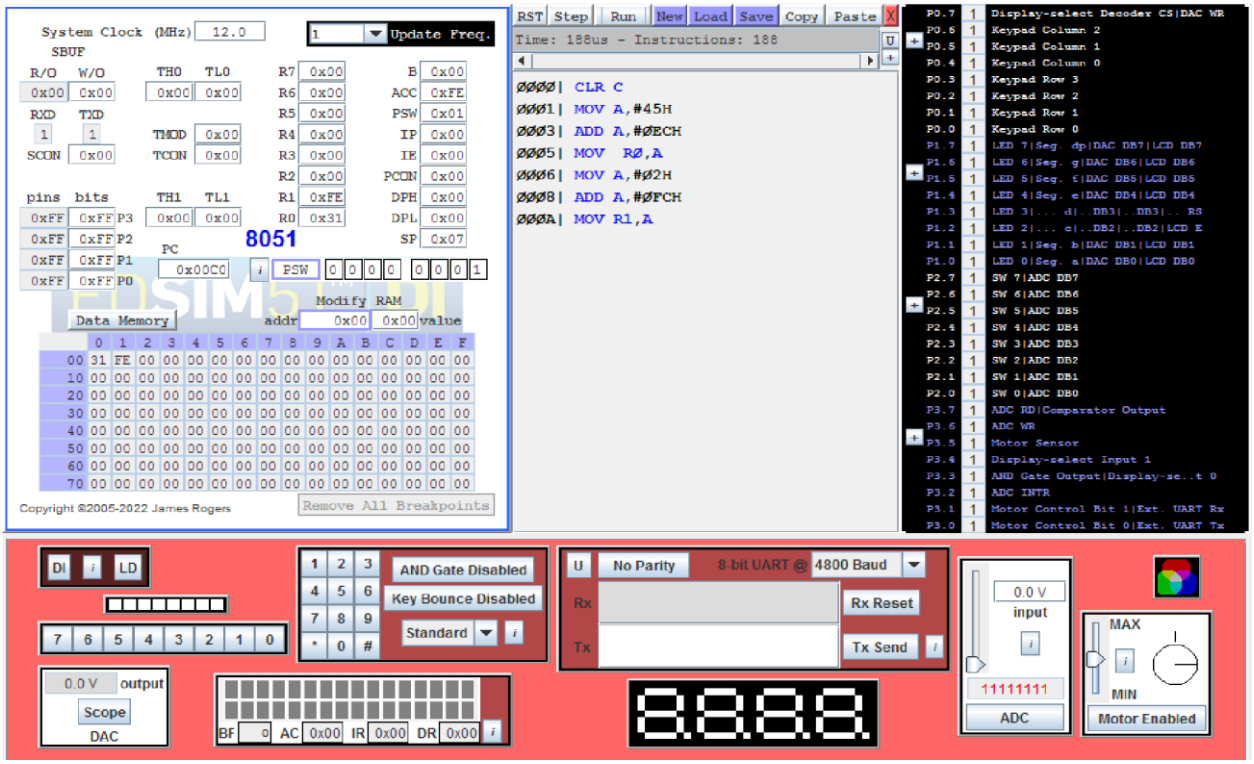
```
MOV A,#25H      ;load 25H to reg A
MOV B,65H       ;load 65H to reg B
MUL AB          ;25H*65H=E99 where B=0EH and A=99H
```

4) DIVISION:

```
MOV A,#95H      ;load 95 into A
MOV B,#0AH      ;move the divisor into B
DIV AB          ;divide the hex number by 10
MOV R0,B        ;the remainder is in B, move it to R0
MOV B,#0AH      ;reload the divisor into B
DIV AB          ;divide the quotient by 10
MOV R1,B        ;move the remainder to R1
MOV R2,A        ;move the last quotient to R2
```

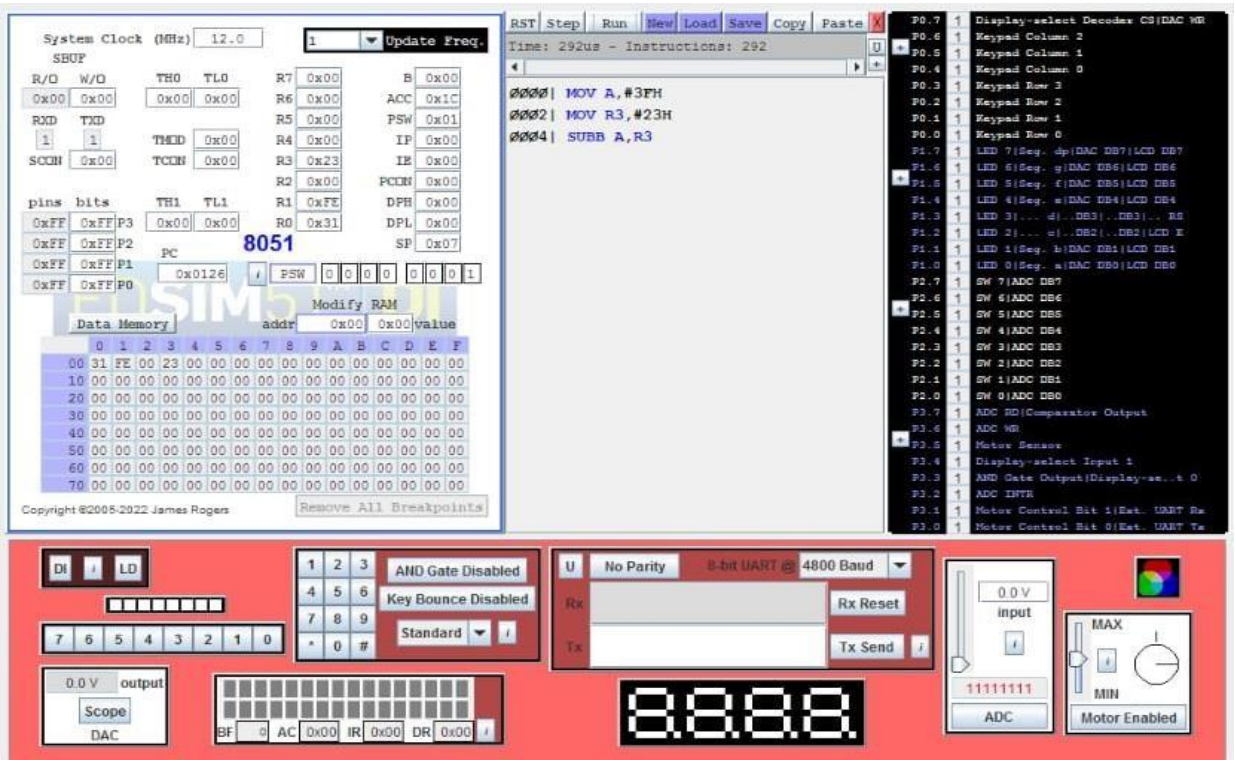
Output:

1.ADDITION:

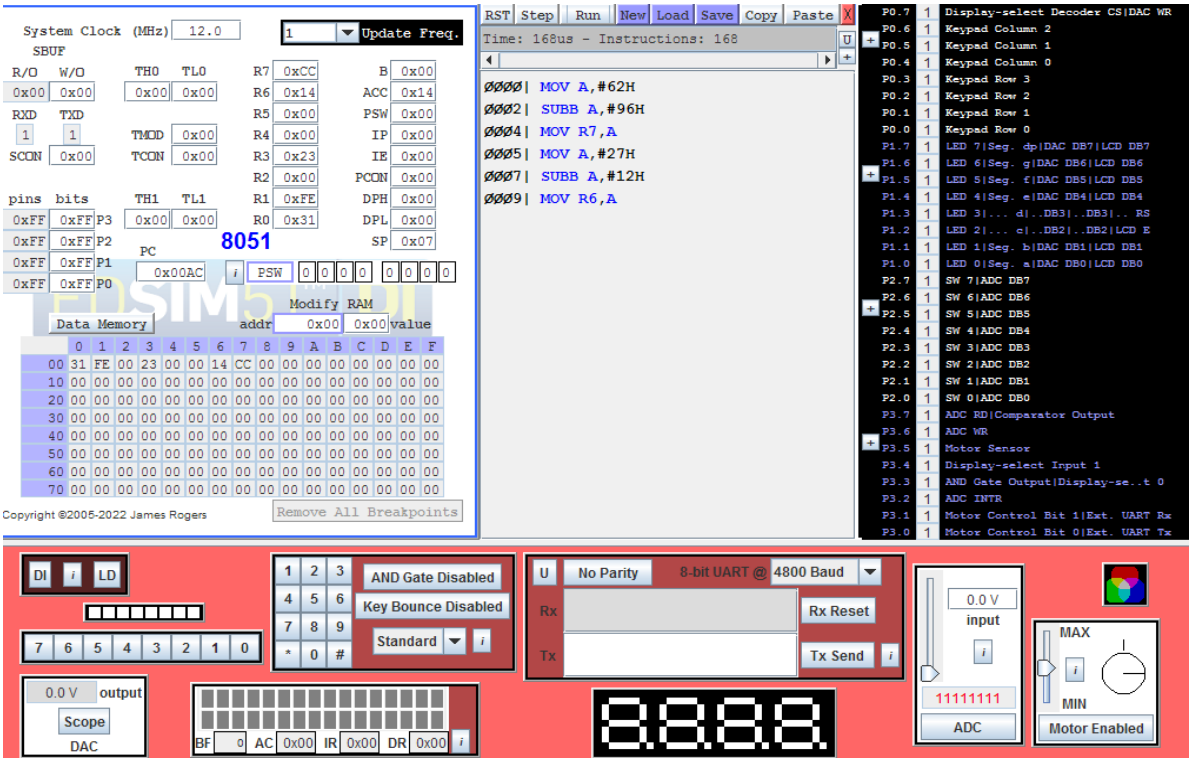


2.SUBTRACTION:

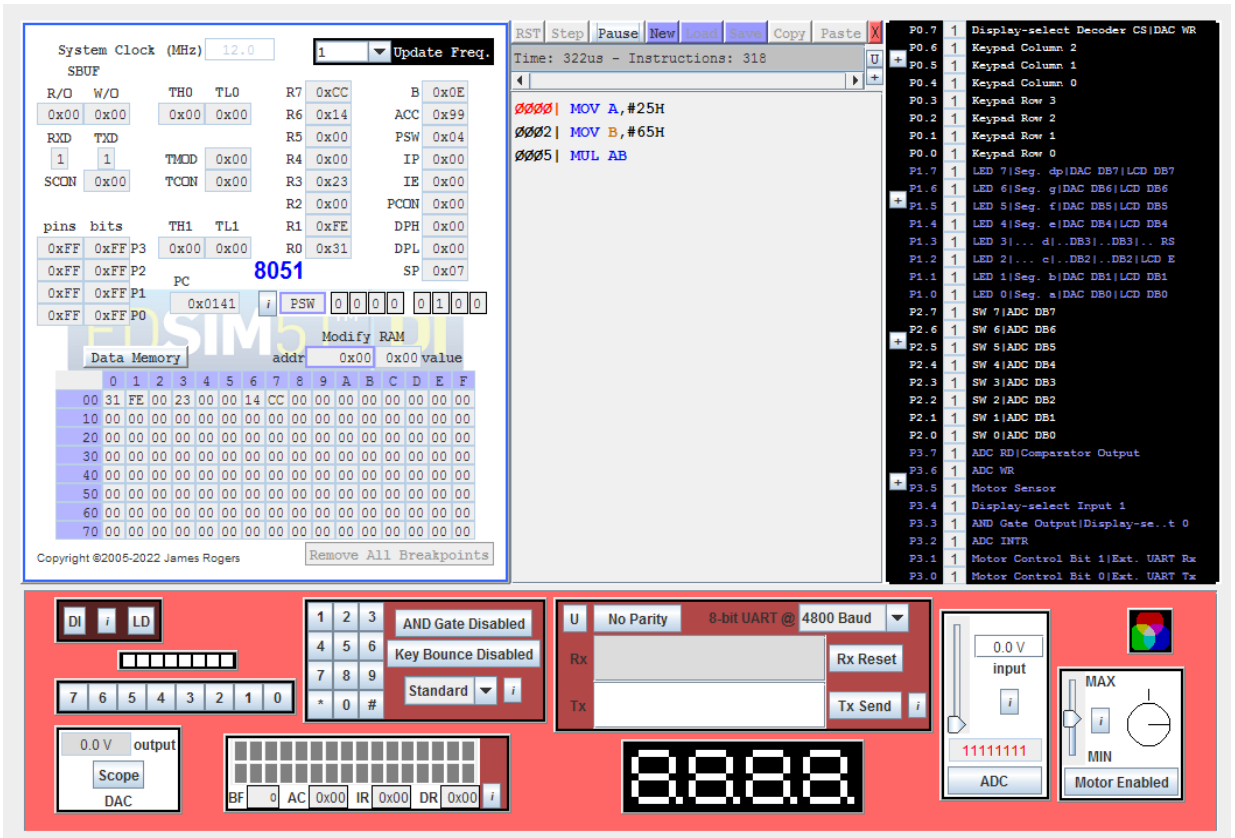
i) Subtraction with CY=0:



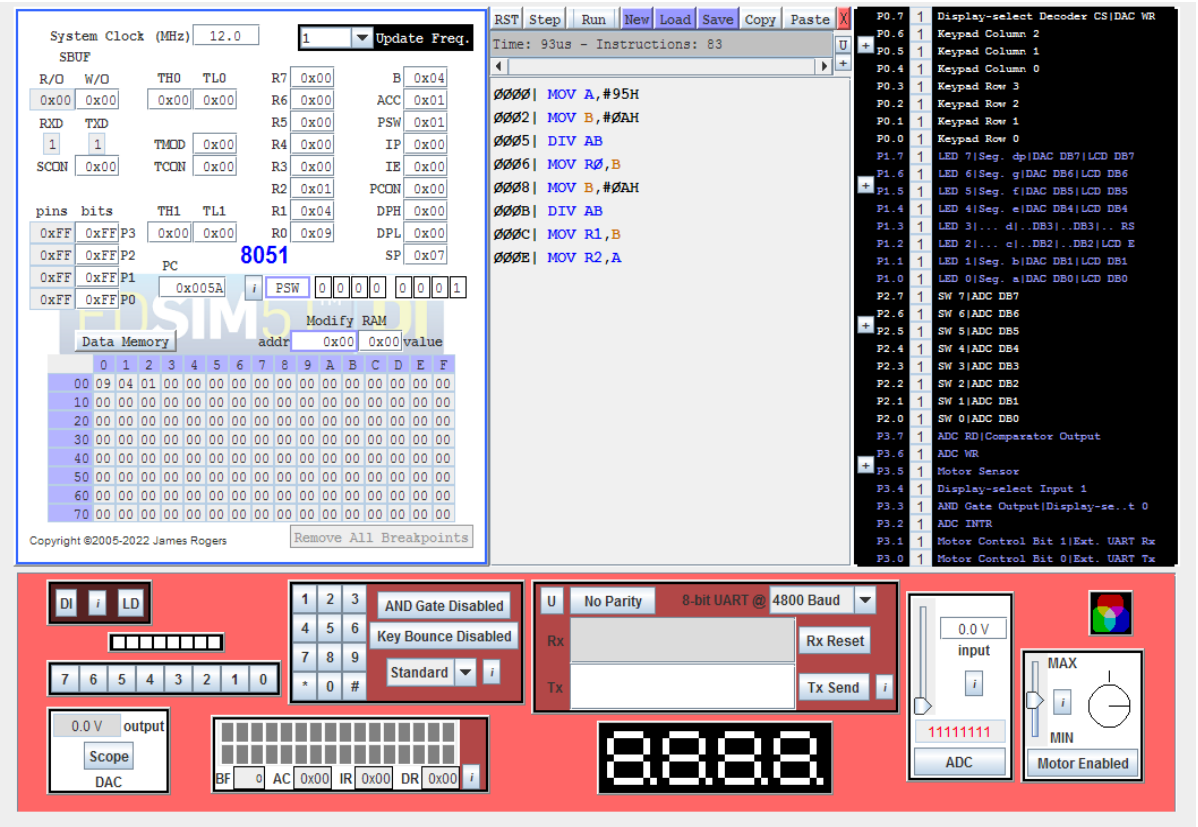
ii) Subtraction with CY=1:



3.MULTIPLICATION:



4.DIVISION



Result:

Thus, the ALU operation using 8051 microcontrollers was performed successfully.

Ex No:5	Introduction to Arduino Platform and Programming
Date:	

Aim:

To Perform Interfacing of LED with ARDUINO and evaluate the response of variations.

APPARATUS REQUIRED:

1. Arduino Kit
2. USB Cable
3. Arduino SDK Software tool
4. Patch cards

THEORY:

LEDs are the most efficient way to turn an electric current into illumination. When a current flows through a diode in the forward direction, it consists of surplus electrons moving in one direction in the lattice and "holes" (voids in the lattice) moving in the other. Occasionally, electrons can recombine with holes. When they do, the process releases energy in the form of photons.

This is true of all semiconductor junctions, but LEDs use materials that maximize the effect. The color of the light emitted (corresponding to the energy of the photon) is determined by the semiconductor materials that form the diode junction.

The latest high-brightness (HB) white LEDs are made possible by the discovery of semiconductor materials that produce blue or ultraviolet photons. In addition to the diode, an HB package contains "yellow" phosphors on the inside of its lens. Some "blue" photons escape, but others excite the phosphors, which then give off "yellow" photons. The result can be tuned in manufacturing to produce "white" light.

Figure. LED symbol

A great deal of LED engineering relates to controlling the quality of this light. From a circuit standpoint, there are a number of ways to interconnect multiple LEDs to increase and manage light output. The general approach is to drive series strings with a constant current.

PROCEDURE:

Step-1: There are 4 LEDs in the kit namely D4, D5, D6 and D7. They are to be connected to the D4, D5, D6 and D7 pins of Arduino board. The power supply +5V and Gnd pins of Arduino board also are to be connected in this section.

Step-2: Connect the USB connector to the USB of Arduino board and the computer system.

Step-3: Using this program, the first LED (left most LED) is switched on for 0.5 sec and then it is

switched off.

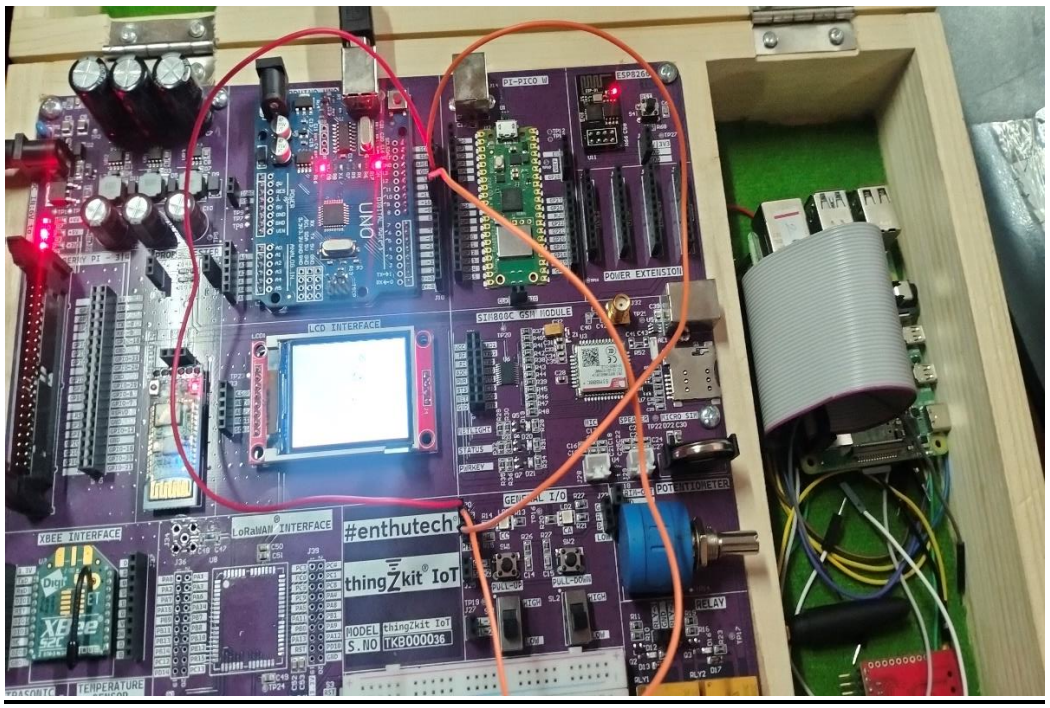
Step-4: After 0.5 sec., the second LED is switched on for 0.5 sec. and then it is switched off. In this way, all the four LEDs are switched on and off. Then the cycle repeats continuously.

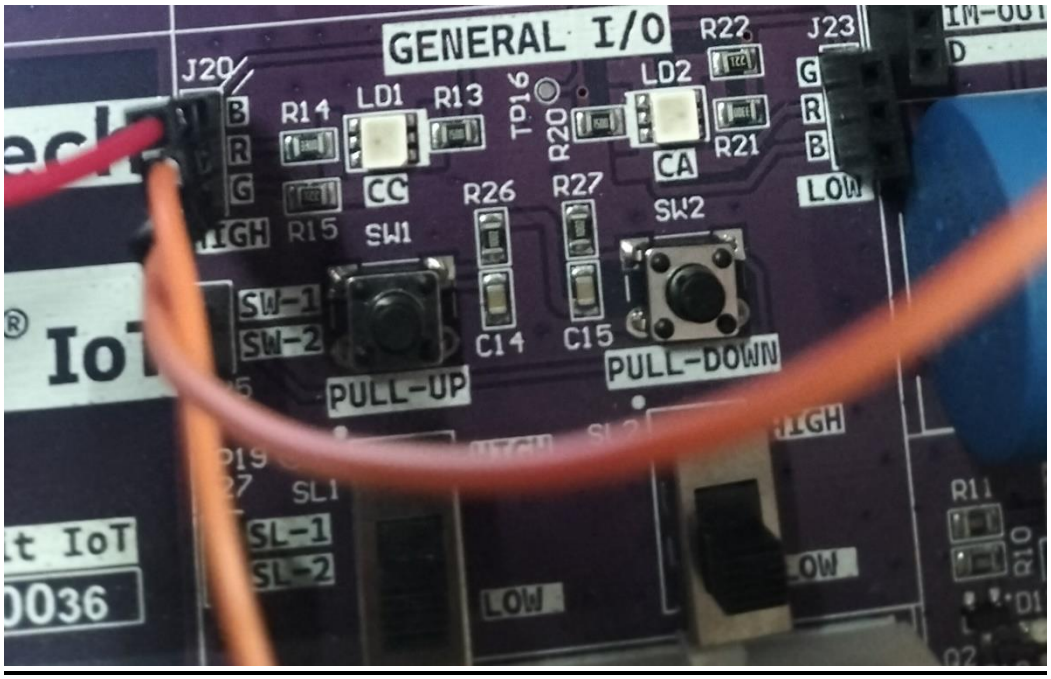
Step-5: In computer, open the sketch software and write the program **LED blinking** and execute the program in sketch and check for the proper result.

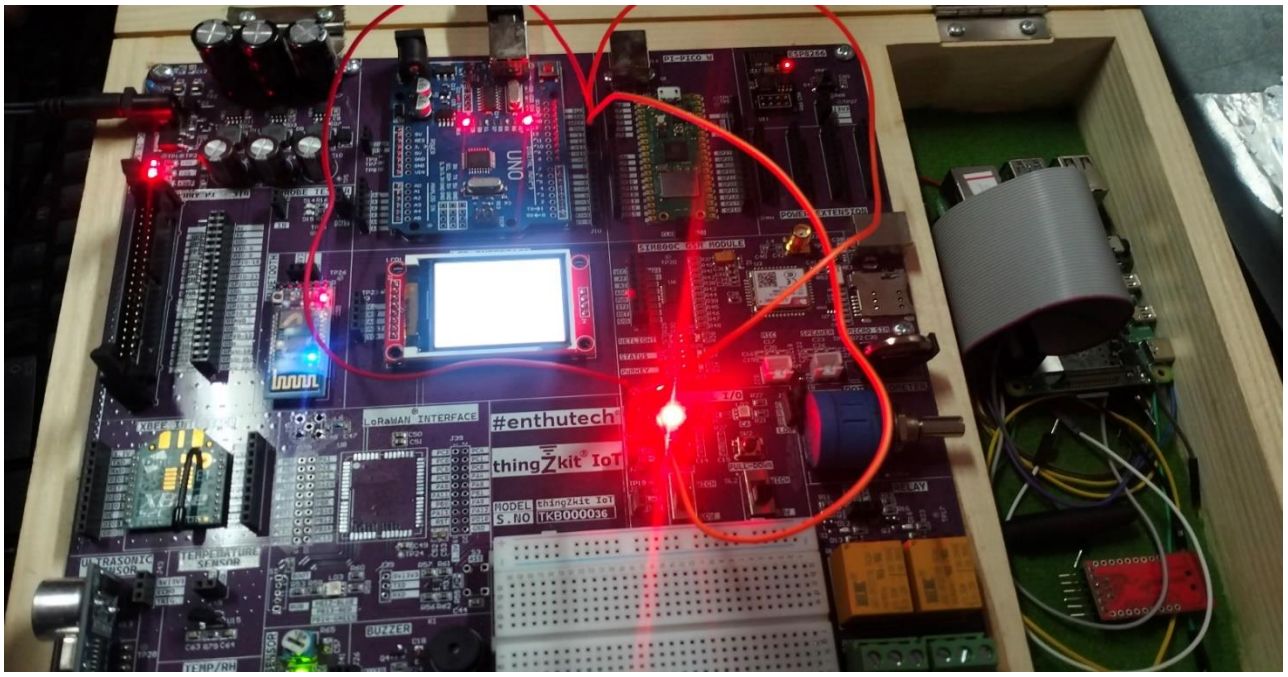
Program :

```
#include <Arduino.h>
/*RGB LED pins (Common Cathode)*/
#define Red_Pin 9
#define Green_Pin 10
#define Blue_Pin 11
/*Delay  (in milliseconds)*/
#define COLOR_CHANGE_DELAY 1000

void setup() {
  /*Initialize the RGB LED pins as outputs*/
  pinMode(Red_Pin, OUTPUT);
  pinMode(Green_Pin, OUTPUT);
  pinMode(Blue_Pin, OUTPUT);
}

void loop() {
  /*Turn on Red, turn off Green and Blue*/
  digitalWrite(Red_Pin, HIGH);
  digitalWrite(Green_Pin, LOW);
  digitalWrite(Blue_Pin, LOW);
  delay(COLOR_CHANGE_DELAY);

  /*Turn off Red, turn on Green, turn off Blue*/
  digitalWrite(Red_Pin, LOW);
  digitalWrite(Green_Pin, HIGH);
  digitalWrite(Blue_Pin, LOW);
  delay(COLOR_CHANGE_DELAY);

  /*Turn off Red, turn off Green, turn on Blue*/
  digitalWrite(Red_Pin, LOW);
  digitalWrite(Green_Pin, LOW);
  digitalWrite(Blue_Pin, HIGH);
  delay(COLOR_CHANGE_DELAY);

  /*Turn ON Red,Green,Blue*/
  digitalWrite(Red_Pin, HIGH);
  digitalWrite(Green_Pin, HIGH);
  digitalWrite(Blue_Pin, HIGH);
  delay(COLOR_CHANGE_DELAY);
}
```

Output:


UNO
PI-PICO W
POWER EXTENSION
LCD INTERFACE
GENERAL I/O
XBEE INTERFACE
LoRaWAN INTERFACE
#enthutech
thingZkit IoT
MODEL thingZkit IoT
S.NO TKB000036
POTENTIOMETER
RELAY
PULL-UP
PULL-DOWN
TEMPERATURE


GENERAL I/O
J20
J23
R14
LD1
R13
TP16
R20
LD2
R22
R21
CC
CA
R26
R27
SW1
SW2
R15
SW-1
SW-2
C14
C15
PULL-UP
PULL-DOWN
LOW
SL-1
SL-2
R11
R10



Result:

Thus the Interfacing of LED with ARDUINO was performed successfully.

Ex No: 6	Explore different communication methods with IoT devices (Zigbee, GSM, Bluetooth)
Date:	

AIM:

To write a python program to explore different communication methods with IoT devices (Zigbee, GSM, Bluetooth)

APPARATUS REQUIRED:

Arduino UNO kit, Arduino SDK Software

PROCEDURE:

1. Include SoftwareSerial library and define LED_pin constant.
2. Initialize SoftwareSerial object 'bluetooth', float variable 'data', and character array 'charArray'.
3. Setup:

   - Begin serial communication at 9600 baud rate.

   - Begin Bluetooth communication at 9600 baud rate (connect Bluetooth module's RX pin to Arduino pin 2, TX pin to Arduino pin 3).

   - Set LED_pin as an output (connect LED's anode to Arduino pin 13, cathode to ground).

4. Loop:

   If Bluetooth data available:

   - Read received character.

   - Print received character.

   - If character is '1', turn on LED (connect LED_pin to digital pin HIGH).

   - If character is '0', turn off LED (connect LED_pin to digital pin LOW).

   - If character is '2':

   - Convert 'data' to charArray.

   - Send charArray over Bluetooth.

   - Delay for transmission.

PROGRAM:

```
#include <SoftwareSerial.h>
 #define LED_pin 13
SoftwareSerial bluetooth(2, 3); // RX, TX
float data=25.98;
char charArray[10];
void setup() {
  Serial.begin(9600); // Initialize the serial monitor
  bluetooth.begin(9600); // Initialize the Bluetooth communication
  pinMode(LED_pin,OUTPUT);
}
void loop() {
  // Check if data is available from Bluetooth module
  if (bluetooth.available()) {
    char receivedChar = bluetooth.read(); // Read the character received via Bluetooth
    Serial.print("Received: ");
    Serial.println(receivedChar);
    // Example: Send a response back to the Bluetooth module
    if (receivedChar == '1') {
      digitalWrite(LED_pin, HIGH);
      Serial.print("LED_ON");
    }
    else if(receivedChar == '0')
    {
     digitalWrite(LED_pin, LOW);
      Serial.print("LED_OFF");
    }
    else if(receivedChar == '2'){
      dtostrf(data, 6, 2, charArray); // float to char convertion
      bluetooth.write(charArray);
      delay(100);
```

```
    }
    }
  }
```

Output:

RESULT:

Thus the python program to explore different communication methods with IoT devices (Zigbee, GSM, Bluetooth) was written and executed successfully.

Ex No:7	Introduction to Raspberry PI Platform and Python Programming
Date:	

Aim:

The purpose of this experiment is to study about Raspberry PI platform and Python programming.

Introduction to Raspberry PI:

The Raspberry Pi is a small, affordable, single-board computer developed by the Raspberry Pi Foundation, with the primary goal of promoting computer science education and fostering DIY projects. It was first introduced in 2012 and has since gained immense popularity as a versatile and accessible computing platform.

Key Features of Raspberry Pi:

1. Affordability:

Raspberry Pi boards are cost-effective, making them accessible to a wide range of users.

2. Compact Size:

Raspberry Pi is credit card-sized, portable, and suitable for embedded systems.

3. Hardware Specifications:

Different models offer varying hardware specs, typically including a CPU, RAM, USB ports, HDMI output, audio output, Ethernet (on some models), and GPIO pins.

4. Operating System:

Raspberry Pi runs on a Linux-based OS, often Raspbian (now known as Raspberry Pi OS), but supports other OS options.

5. GPIO Pins:

General Purpose Input/Output (GPIO) pins allow interaction with the physical world, making it ideal for IoT and electronics projects.

6. Community and Support:

Raspberry Pi has a large and active user community, providing online resources, tutorials, forums, and project ideas.

Common Raspberry Pi Models:

1. Raspberry Pi 4:

Features a quad-core ARM Cortex-A72 CPU, up to 8GB of RAM, dual HDMI outputs, USB 3.0 ports, and Gigabit Ethernet (as of my last update in January 2022).

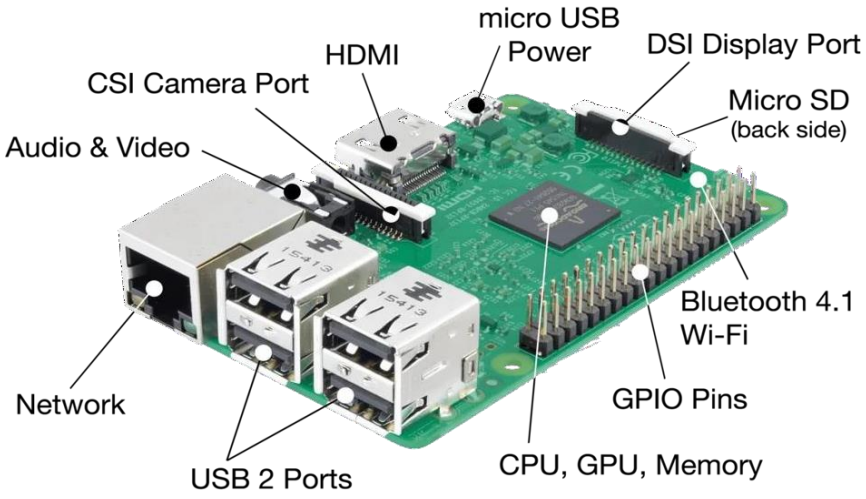
2.Raspberry Pi 3:

Equipped with a quad-core ARM Cortex-A53 CPU, 1GB of RAM, Wi-Fi, and Bluetooth.

3.Raspberry Pi Zero:

An ultra-compact, low-cost model with a single-core CPU, 512MB of RAM, and a mini HDMI port.

Parts in Raspberry Pi:



Raspberry Pi Hardware Specifications:

The raspberry pi board comprises a program memory (RAM), processor and graphics chip, CPU, GPU, Ethernet port, GPIO pins, Xbee socket, UART, power source connector. And various interfaces for other external devices. It also requires mass storage, for that we use an SD flash memory card. So that raspberry pi board will boot from this SD card similarly as a PC boots up into windows from its hard disk.

Essential hardware specifications of raspberry pi board mainly include SD card containing Linux OS, US keyboard, monitor, power supply and video cable. Optional hardware specifications include USB mouse, powered USB hub, case, internet connection, the Model A or B: USB WiFi adaptor is used and internet connection to Model B is LAN cable.

Memory

The raspberry pi model Aboard is designed with 256MB of SDRAM and model B is designed with 51MB.Raspberry pi is a small size PC compare with other PCs. The normal PCs RAM memory is available in gigabytes. But in raspberry pi board, the RAM memory is available more than 256MB or 512MB

CPU (Central Processing Unit):

The Central processing unit is the brain of the raspberry pi board and that is responsible for carrying out the instructions of the computer through logical and mathematical operations. The raspberry pi uses ARM11 series processor, which has joined the ranks of the Samsung galaxy phone.

GPU (Graphics Processing Unit):

The GPU is a specialized chip in the raspberry pi board and that is designed to speed up the operation of image calculations. This board designed with a Broadcom video core IV and it supports OpenGL

Ethernet Port:

The Ethernet port of the raspberry pi is the main gateway for communicating with additional devices. The raspberry pi Ethernet port is used to plug your home router to access the internet.

GPIO Pins:

The general purpose input & output pins are used in the raspberry pi to associate with the other electronic boards. These pins can accept input & output commands based on programming raspberry pi. The raspberry pi affords digital GPIO pins. These pins are used to connect other electronic components. For example, you can connect it to the temperature sensor to transmit digital data.

XBee Socket:

The XBee socket is used in raspberry pi board for the wireless communication purpose.

Power Source Connector:

The power source cable is a small switch, which is placed on side of the shield. The main purpose of the power source connector is to enable an external power source.

UART:

The Universal Asynchronous Receiver/ Transmitter is a serial input & output port. That can be used to transfer the serial data in the form of text and it is useful for converting the debugging code.

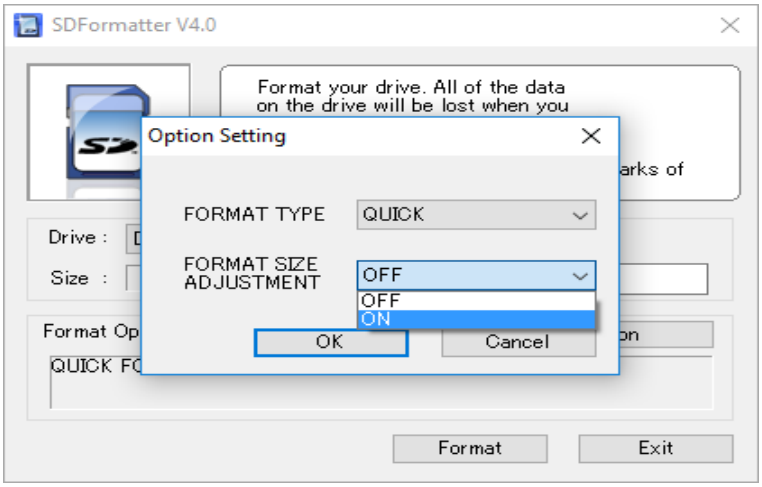
Display

The connection options of the raspberry pi board are two types such as HDMI and Composite.Many LCD and HD TV monitors can be attached using an HDMI male cable and with a low-cost adaptor. The versions of HDMI are 1.3 and 1.4 are supported and 1.4 version cable is recommended. The O/Ps of the Raspberry Pi audio and video through HMDI, but does not support HDMI I/p. Older TVs can be connected using composite video. When using a composite video connection, audio is available from the 3.5mm jack socket and can be sent to your TV. To send audio to your TV, you need a cable which adjusts from 3.5mm to double RCA connectors.

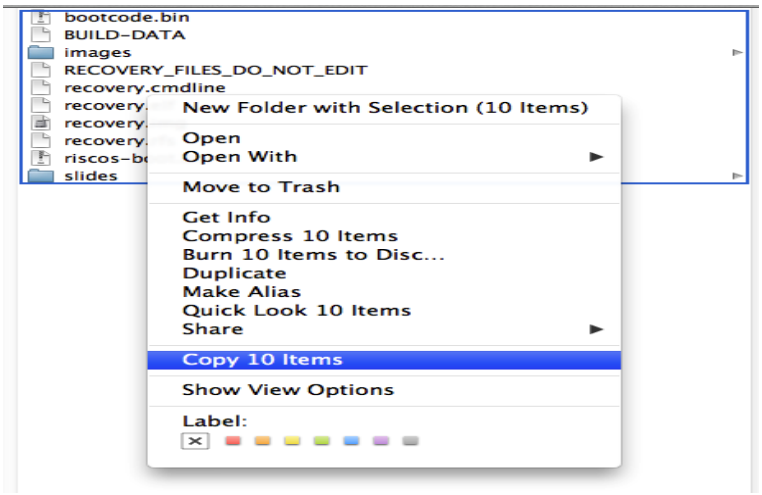
Steps to install:

Step 1: Download NOOBS and extract it.

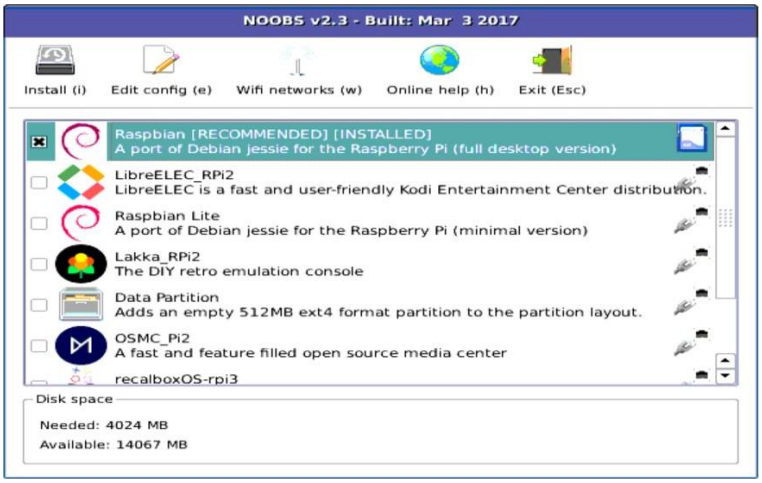
Step 2: Format an SD card.



Step 3: Put the NOOBS files on the SD card.

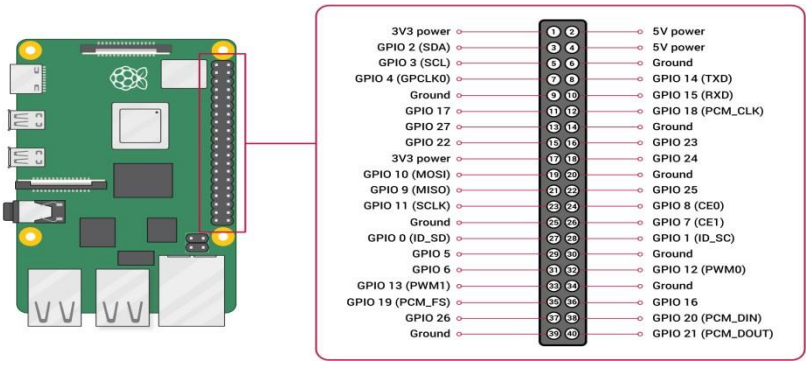


Step 4: Put your SD card into your Raspberry Pi and boot it up.



Boot up Rainbow screen.

Raspberry pi GPIO PINOUT



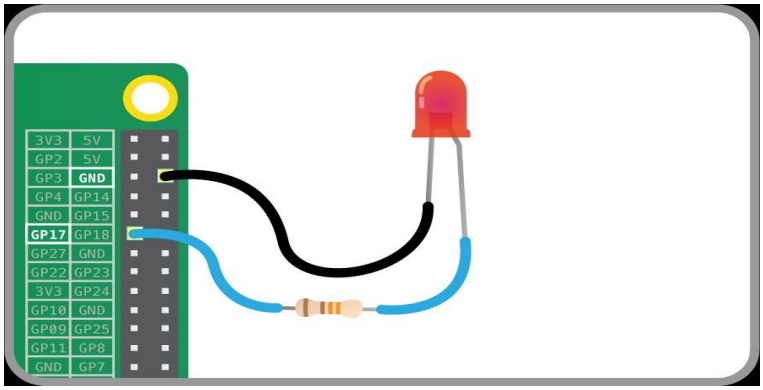
Python Programing in Raspberry Pi:

Python programming on the Raspberry Pi is a powerful combination that has revolutionized the world of embedded computing and DIY electronics. The Raspberry Pi, a credit-card-sized single-board computer, is particularly well-suited for Python due to its simplicity, versatility, and a vast array of libraries and frameworks.

Python's clear syntax and readability make it an ideal language for beginners, allowing enthusiasts to quickly grasp the basics of programming while experimenting with hardware interfaces. The Raspberry Pi's GPIO (General Purpose Input/Output) pins enable users to interact with the physical world by connecting sensors, actuators, and other devices, all of which can be easily controlled through Python scripts. This seamless integration has sparked a surge in innovative projects ranging from home automation and robotics to IoT applications, fostering a vibrant community of developers and makers exploring the limitless possibilities of Python programming on the Raspberry Pi platform.

LED Blinking Using RaspberryPI:

Blinking is done by connecting an LED to one of GPIO pins of PI and turning it ON and OFF.



PROGRAM:

```
import machine as Gpio
import utime as TM

# Define the RGB LED pins (common cathode)
Red_Pin = Gpio.Pin(15, Gpio.Pin.OUT)
Green_Pin = Gpio.Pin(14, Gpio.Pin.OUT)
Blue_Pin = Gpio.Pin(13, Gpio.Pin.OUT)

# Helper function to set RGB color
def set_rgb_color(red, green, blue):
Red_Pin.value(red)
Green_Pin.value(green)
Blue_Pin.value(blue)

# Main loop
while True:
# Red
set_rgb_color(1, 0, 0)
TM.sleep(1)

# Green
set_rgb_color(0, 1, 0)
TM.sleep(1)

# Blue
set_rgb_color(0, 0, 1)
TM.sleep(1)
```

Result:

Thus, the Raspberry Pi platform and python programing was studied successfully.

Ex No: 8	Interfacing sensors with Raspberry PI
Date:	

AIM:

To write a python program to interface sensors with Raspberry PI.

APPARATUS REQUIRED:

Raspberry PI kit, IR sensors, connecting wires, python IDE.

PROCEDURE:

1. Setup and connect the Raspberry PI with the monitor and I/O devices.
2. Now connect the Ultrasonic sensor with Raspberry PI.
3. Type the python program to measure the distance with ultrasonic sensor.
4. Now save and run the progr

PROGRAM:

```
import requests
import json
import time
import RPi.GPIO as GPIO


GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)


GPIO_TRIGGER = 24 #RPi 18
GPIO_ECHO    = 25 #RPi 22
print ("Ultrasonic Measurement")


# Set pins as output and input
GPIO.setup(GPIO_TRIGGER,GPIO.OUT)  # Trigger
GPIO.setup(GPIO_ECHO,GPIO.IN)      # Echo

 # Set trigger to False (Low)
GPIO.output(GPIO_TRIGGER, False)
 time.sleep(0.5)
```

```
while True:
try:
    # Send 10us pulse to trigger
    GPIO.output(GPIO_TRIGGER, True)
    time.sleep(0.00001)
    GPIO.output(GPIO_TRIGGER, False)
    while GPIO.input(GPIO_ECHO)= =0:
       start = time.time()
    while GPIO.input(GPIO_ECHO)= =1:
       stop = time.time()
    # Calculate pulse length

    elapsed = stop-start
    # Distance pulse travelled in that time is time
    # multiplied by the speed of sound (cm/s)
    distance = elapsed * 34300
    # That was the distance there and back so halve the value
    distance = distance / 2  #in centimeter
    inch=distance/2.5  #in Inch
    print ("Distance (cm): %.1f" % distance)
    print ("Distance (inch): %.1f" % inch)
    time.sleep(20)
```

RESULT:

Thus the python program to interface sensors with Raspberry PI was written and executed successfully

Ex No: 09	Communicate between Arduino and Raspberry PI using any wireless medium.
Date:	

AIM:

To Perform and communicate between Arduino and Raspberry Pi using any Wireless medium and evaluate the response.

APPARATUS REQUIRED:

- Arduino IOT Kit
- USB Cable
- Arduino SDK Software tool
- VGA to mini HDMI Converter cable
- Power Adapter
- Short USB power cable
- Patch cards

PROCEDURE:

Arduino:

- Write the arduino program and upload it
- As per connection bluetooth to arduino TX-RX , RX-TX

Raspberry:

edit the DOS Prompt

sudo bluetoothctl ## Bluetooth path

agent on ## agent registered

scan on ## our bluetooth will display copy the mac address

sudo rfcomm --help ## rfcomm command will display

sudo rfcomm bind 7 --mac address-------- ## enter command paste your mac ad

Connect the bluetooth on raspberry pi ,right corner click bluetooth icon and pair

- Run the program both side arduino & raspberry
- Enter data 1 led will go on state
- enter data 0 led will go off state
- finally disconnect the raspberry bluetooth -sudo rfcomm release 7 ----mac address---

Arduino Program:

```
int led=13;
void setup()
{

  pinMode(led, OUTPUT);
  Serial.begin(9600); //default baud rate for bt 38400

}
void loop()
{
  if(Serial.available())
           {
             int a=Serial.parseInt();
             Serial.println(a);
          if (a==1)
          {
            digitalWrite(led, HIGH);

           }
           if (a == 0)
           {
               digitalWrite(led, LOW);

             }
  }
}
```

Raspberry Pi Program:

```
import serial
import time
bluetooth=serial.Serial("/dev/rfcomm7",9600)
```

```
while True:
    a=input("enter:-")
    string='X{0}'.format(a)
    bluetooth.write(string.encode("utf-8"))
```

RESULT:

Thus, the communication between Arduino and Raspberry Pi using any Wireless medium was done successfully.

Ex No: 10	Setup a cloud platform to log the data.
Date:	

AIM:

To setup a cloud platform to log the data.

CLOUD PLATFORM:

ThingSpeak

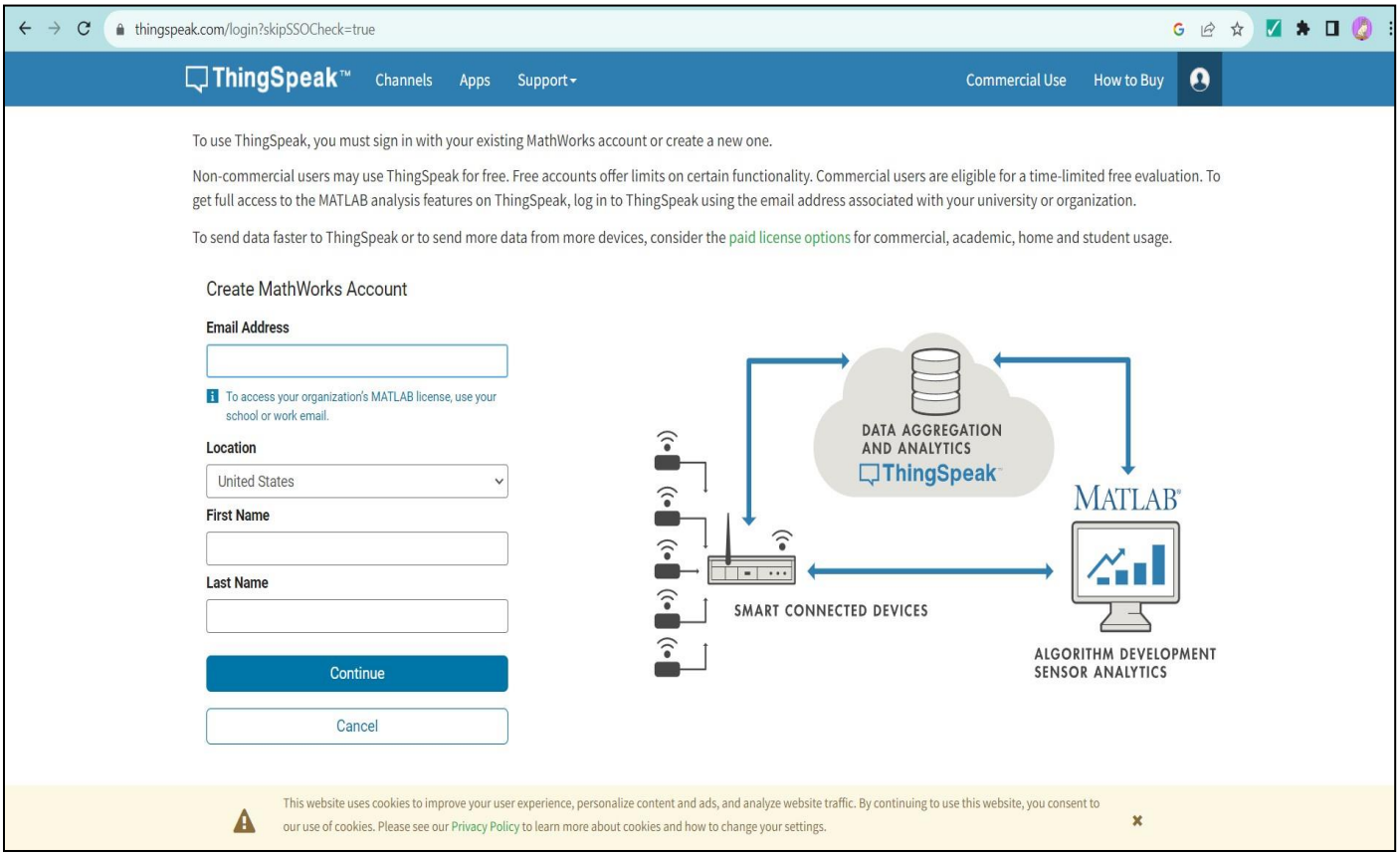
ThingSpeak:

ThingSpeak is an IoT analytics platform service that allows you to aggregate, visualize, and analyze live data streams in the cloud. You can send data to ThingSpeak from your devices, create instant visualization of live data, and send alerts.

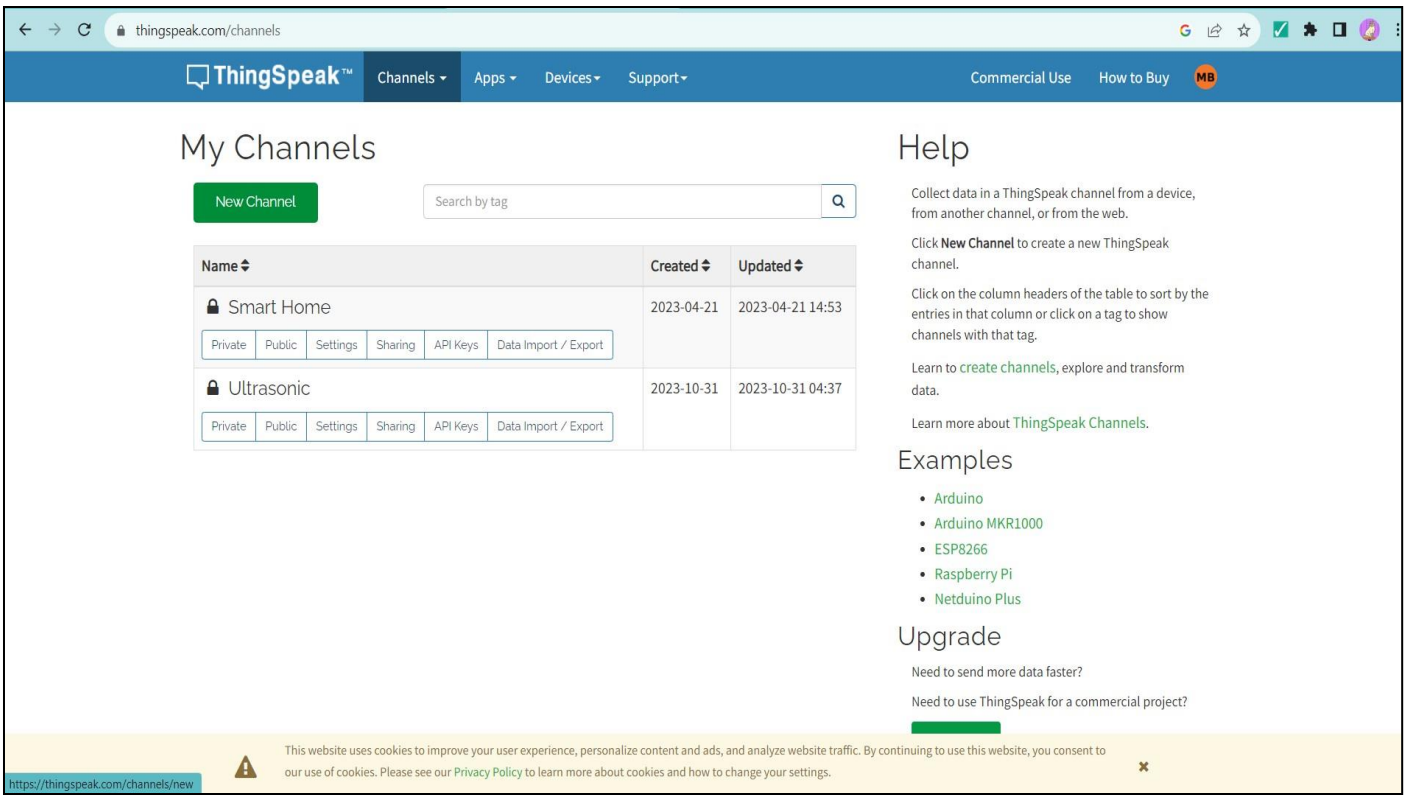
Here we are going to setup ThingSpeak and log the data of Ultrasonic Sensor.

PROCEDURE:

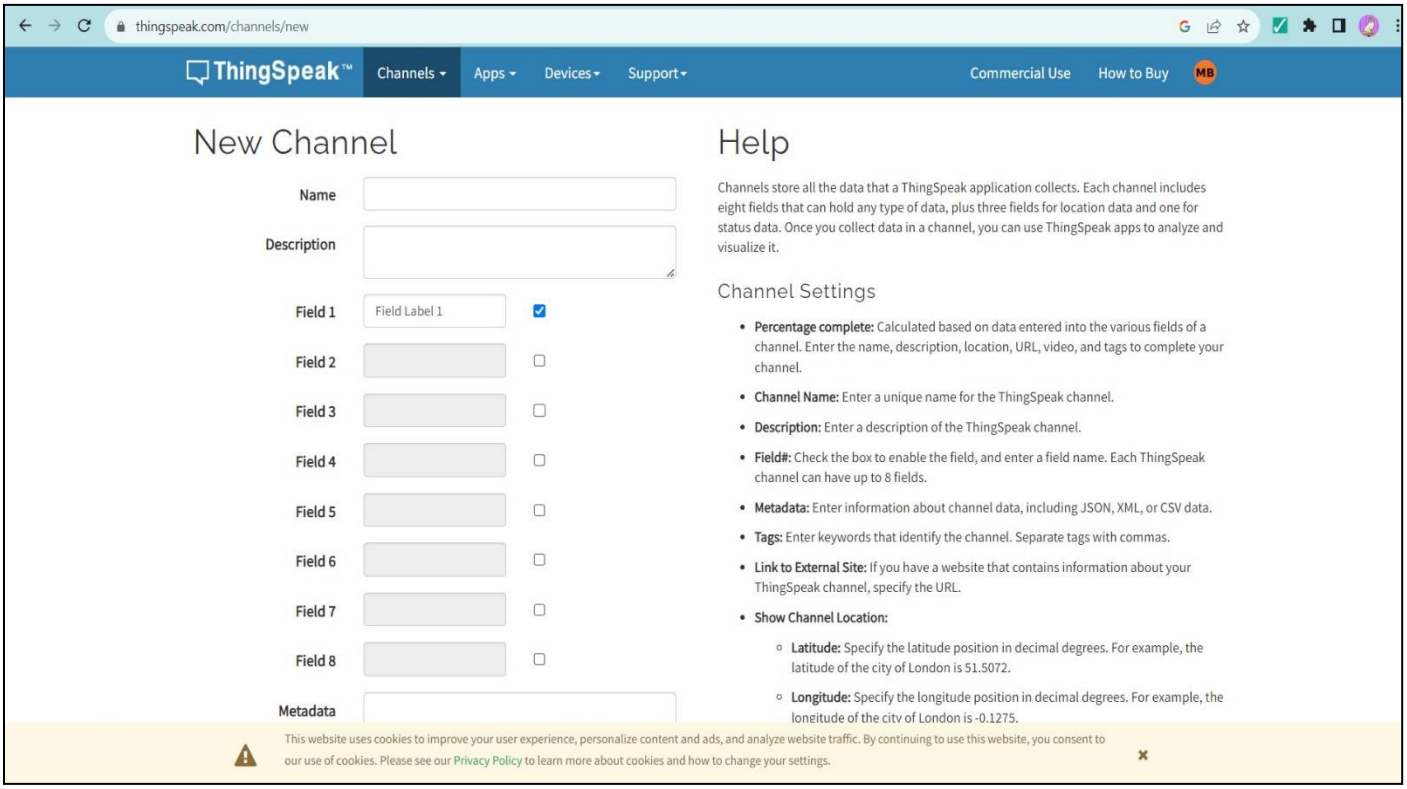
1. **Create a ThingSpeak Account:**
   If you don't have an account, sign up on the ThingSpeak website.



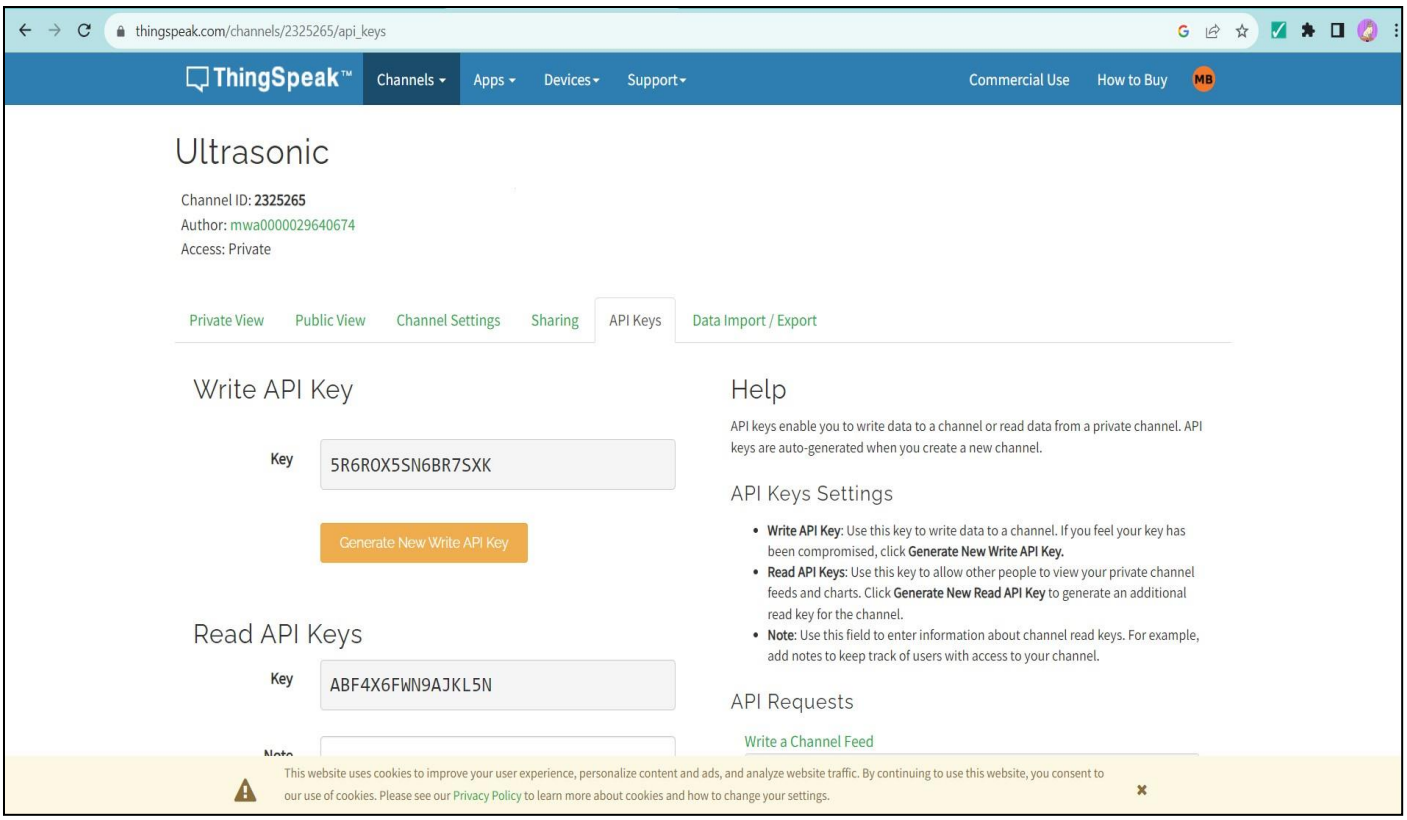
2. **Log in to ThingSpeak:**
   Once you're logged in, go to your ThingSpeak channel.



3. **Create a Channel:**
   a. Click on "Channels" and then "New Channel."
   b. Fill in the required information, like Name, Field labels (e.g., Distance), and any other relevant details.



4. **Get Your Write API Key:**
    a. Go to the "API Keys" tab to find your Write API Key.
    b. This key is essential for updating data on your channel.



5. **Configure Your Ultrasonic Sensor:**
    a. Set up your ultrasonic sensor to measure distance.
    b. Connect your sensor to a microcontroller or single-board computer (like Arduino or Raspberry Pi).

6. **Program Your Device:**
    a. Use the ThingSpeak API to send data to your channel.
    b. Include your Write API Key in the HTTP POST request.

7. **Test Your Setup:**
    a. Upload your code to your device.
    b. Check your ThingSpeak channel to ensure data is being logged correctly.

RESULT:

Thus a cloud platform to log the data was setup successfully.

Ex No: 11	Log Data using Raspberry PI and upload to the cloud platform.
Date:	

AIM:

To write a python program to log data using Raspberry PI and upload to the cloud platform.

APPARATUS REQUIRED:

Raspberry PI kit, Ultrasonic sensors, connecting wires, python IDE, ThingSpeak.

PROCEDURE:

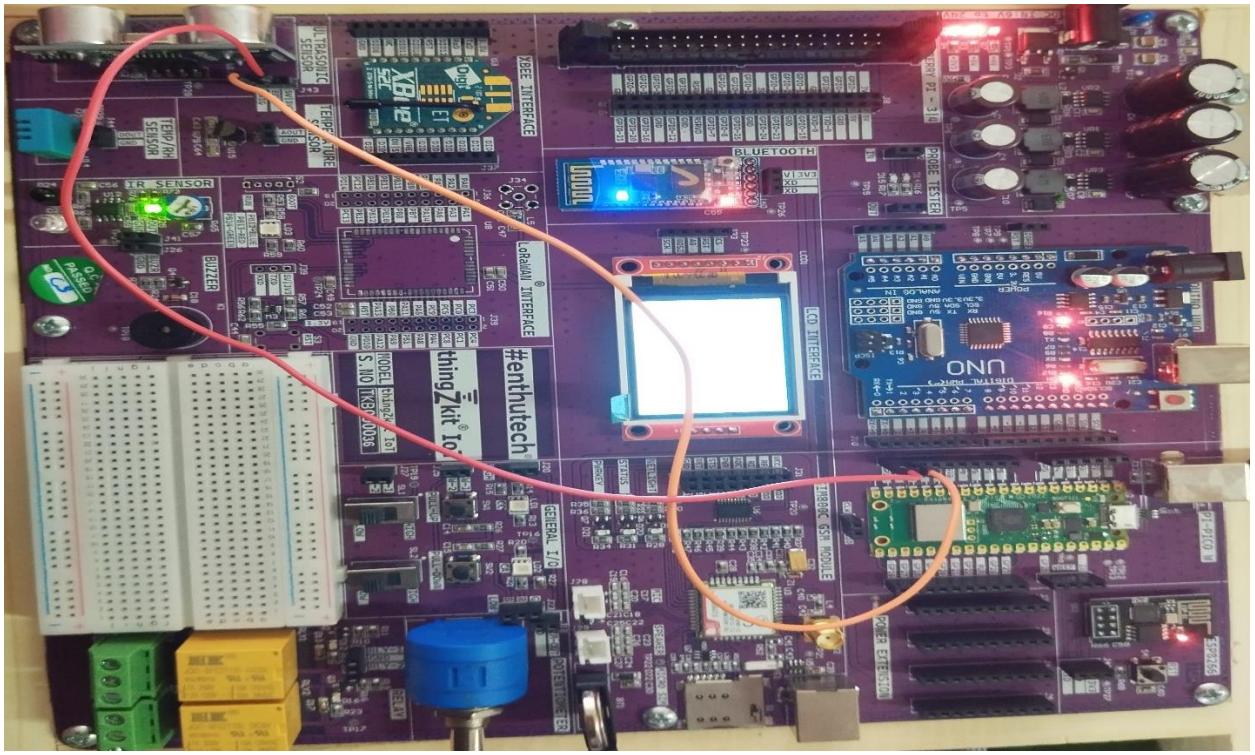
1. Setup and connect the Raspberry PI with the monitor and I/O devices.
2. Connect the Ultrasonic Sensor with Raspberry PI.
3. Setup the cloud platform (ThingSpeak) to upload the data.
4. Type the python program to upload the data from Ultrasonic Sensor to ThingSpeak.
5. Copy and paste your ThingSpeak channel write API key in the program.
6. Now save and run the program.

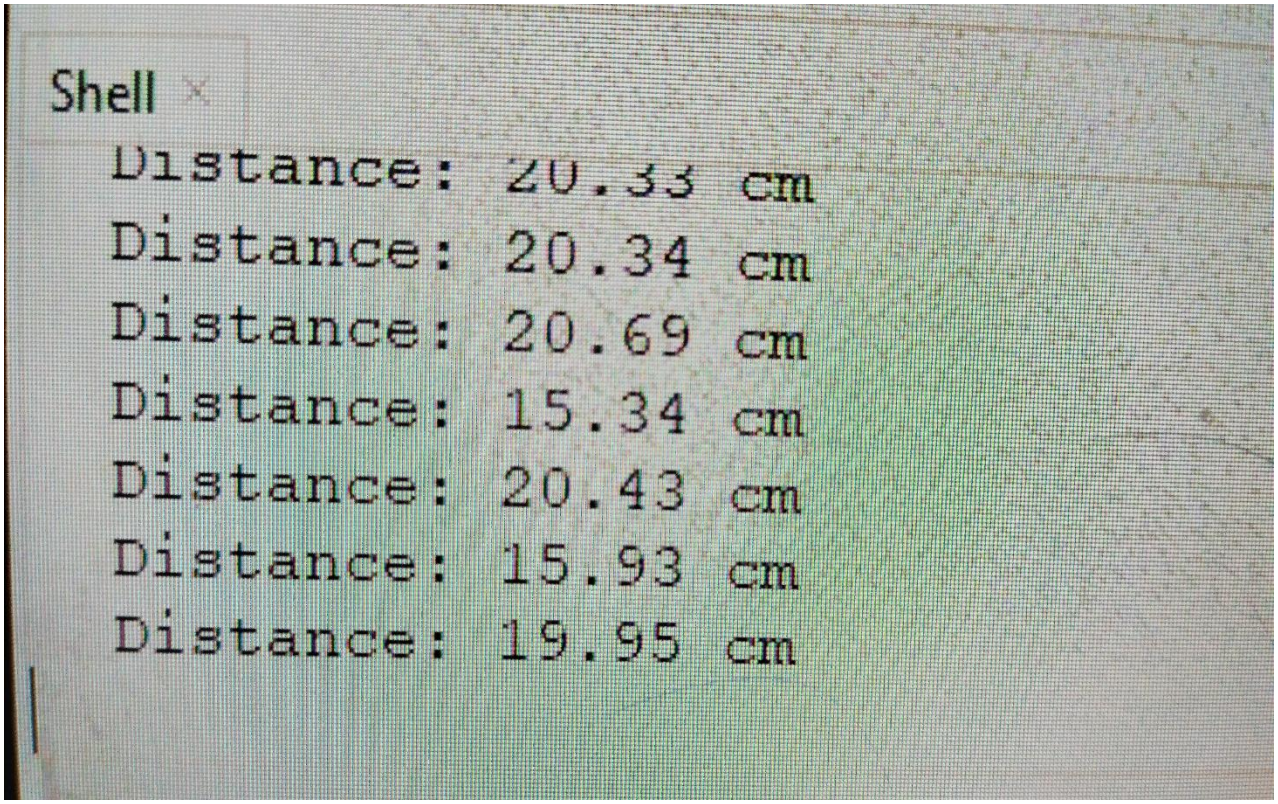
PROGRAM:

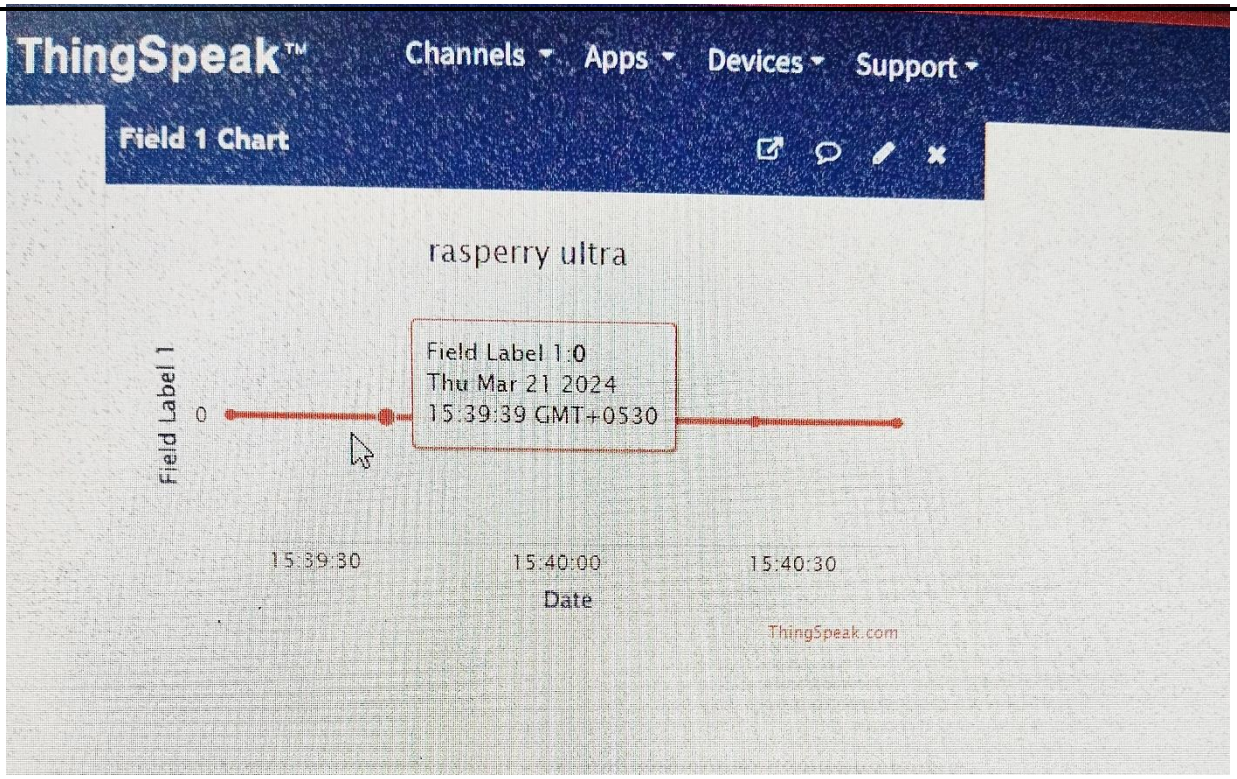
```
import machine
import utime
import urequests
import network

IR_SENSOR_PIN = 2  # GPIO pin for the IR sensor
API_KEY = "BYRGZSN129A5E7P9"
THINGSPEAK_URL = "https://api.thingspeak.com/update?api_key={}".format(API_KEY)
WIFI_SSID = "SHRUTI L G"
WIFI_PASSWORD = "12345678"

ir_sensor = machine.Pin(IR_SENSOR_PIN, machine.Pin.IN)

wlan = network.WLAN(network.STA_IF)

def connect_to_wifi():
    wlan.active(True)
    if not wlan.isconnected():
        print("Connecting to WiFi...")
        wlan.connect(WIFI_SSID, WIFI_PASSWORD)
        while not wlan.isconnected():
            pass
        print("Connected to WiFi")
```

```
def read_ir_sensor():
    return ir_sensor.value()

def send_to_thingspeak(data):
    # Send data to ThingSpeak
    url = "{}&field1={}".format(THINGSPEAK_URL, data)
    response = urequests.get(url)
    print("ThingSpeak response:", response.text)

# Connect to WiFi
connect_to_wifi()

while True:
    ir_data = read_ir_sensor()
    print("IR Sensor Data:", ir_data)

    # Send data to ThingSpeak via HTTP
    send_to_thingspeak(ir_data)

    utime.sleep(15)  # Wait for 15 seconds before the next reading
```

OUTPUT:


XBEE INTERFACE
BLUETOOTH
IR SENSOR
BUZZER
LCD INTERFACE
#enthutech
UNO


Shell
Distance: 20.33 cm
Distance: 20.34 cm
Distance: 20.69 cm
Distance: 15.34 cm
Distance: 20.43 cm
Distance: 15.93 cm
Distance: 19.95 cm



RESULT:

Thus the python program to log data using Raspberry PI and upload to the cloud platform waswritten and executed successfully.

Ex No: 12	Design an IOT based system.
Date:	

AIM:

To design an IOT based System with Node MCU ESP8266 and BLYNK App and evaluate the response of variations.

APPARATUS REQUIRED:

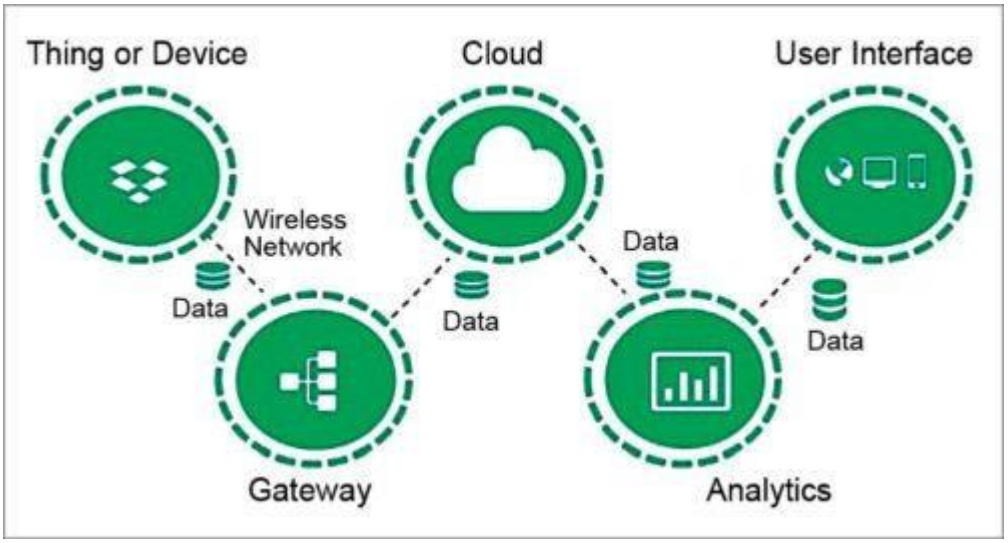
1. Embedded IOT Kit
2. USB Cable
3. Software tool
4. Patch cards
5. Power Adapter

THEORY:

A quick guide to designing a perfect Internet of Things (IoT) system taking into account performance, connectivity, power consumption and security issues

The Internet of Things (IoT) is no longer a technology of the future. Smart cities, connected industries and smart households have indeed ushered in an era where machines can communicate. The beauty of this technology lies in the fact that the complex backend structure of systems is represented to the end-user in the simplest possible form. This requires profound design know-how.

The IoT can be designed at different scales for different uses. It can start from our homes with simple lighting or appliance control, and expand into the realm of factories and industries with automated machines, smart security systems and central management systems—called connected factories. It has scaled up to entire cities with smart parking, smart metering, waste management, fire control, traffic management and any similar functions involved. However, irrespective of the scale of application, the main IoT backbone remains similar.



The IoT architecture is multi-layered with delicate components intricately connected to each other. It starts with sensors, which are the source of data being collected. Sensors pass data onto an adjacent edge device, which converts data into readable digital values and stores these temporarily. When the edge senses a suitable wireless network or the Internet, it pushes the locally stored data to a cloud server involved in the application. The data is processed, analysed, stored and forwarded to the end-user device, represented by an application software. All the design fundamentals and challenges revolve around these layers.

Designing the connectivity module:

The ESP8266 is a system on a chip (SOC) Wi-Fi microchip for Internet of Things (IoT) applications produced by Espressif Systems.

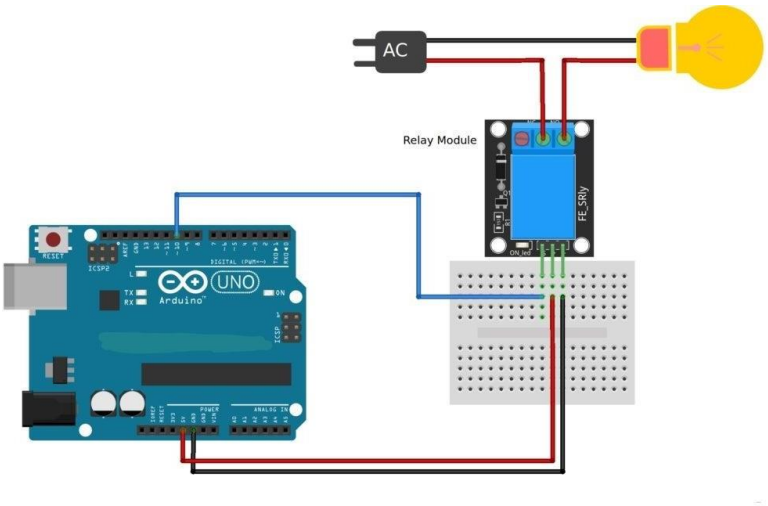
Given its low cost, small size and adaptability with embedded devices, the ESP8266 is now used extensively across IoT devices. Although it's now been succeeded by the newer generation ESP32 microcontroller chip, the ESP8266 is still a popular choice for IoT developers and manufacturers.

In this article, we'll explain the main features of ESP8266 modules and development boards and their application within the world of IoT.

The ESP8266 module enables microcontrollers to connect to 2.4 GHz Wi-Fi, using IEEE 802.11 bgn. It can be used with ESP-AT firmware to provide Wi-Fi connectivity to external host MCUs, or it can be used as a self-sufficient MCU by running an RTOS-based SDK. The module has a full TCP/IP stack and provides the ability for data processing, reads and controls of GPIOs.

Espressif NodeMCU module V1.0:

This board has the ESP-12E module and comes with 4 Mbits of flash and features a row of pins on each side of the breadboard. The board comes with four communication interfaces: SPI, I2C, UART, and I2S, with 16 GPIO and one ADC. The RAM is 160KB, divided into 64KB for instruction and 96KB for data.

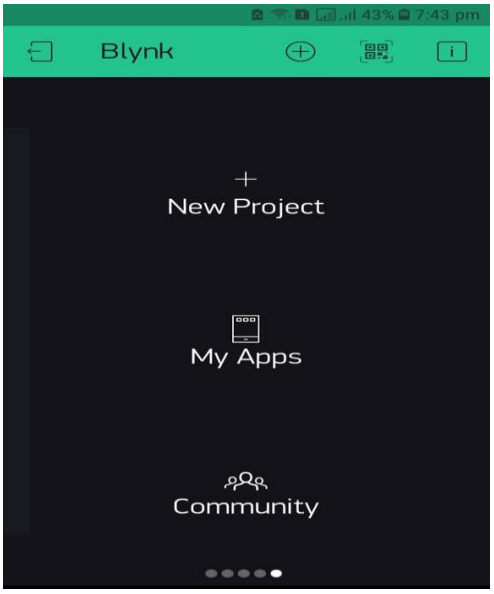


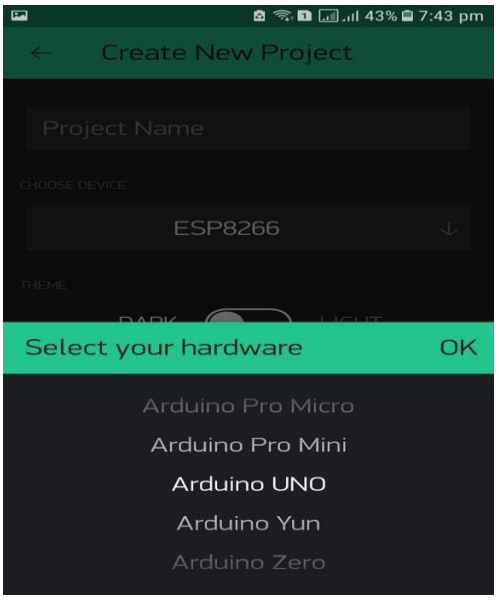
PROGRAM:

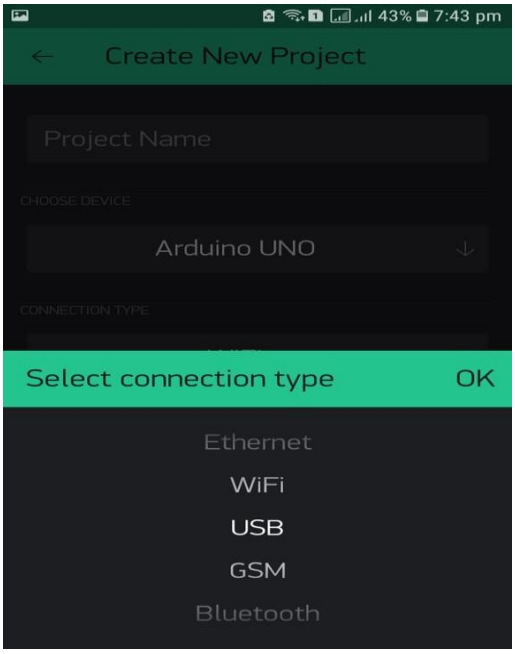
```
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>

// Your Wi-Fi credentials
char auth[ ] = "          ";
char ssid[ ] = "          ";
char pass[ ] = "          ";

void setup( )
{
  // Initialize Serial Monitor
  Serial.begin(9600);

  // Connect to Wi-Fi
  Blynk.begin(auth, ssid, pass);

  // Set GPIO pins for relays as OUTPUT

  pinMode(D1, OUTPUT); // Relay 1

}
void loop( )
{
  Blynk.run( );
}
```

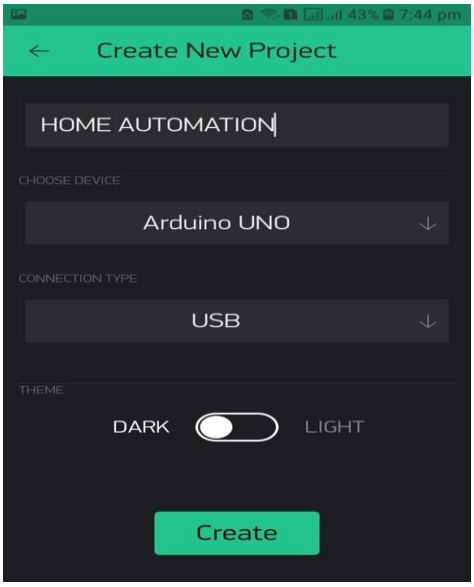
```
// Blynk virtual pins for the 8 relays (V1)
BLYNK_WRITE(V1)
{
  int value = param.asInt( ); // Get the value from the app
  digitalWrite(D1, value);
}
```

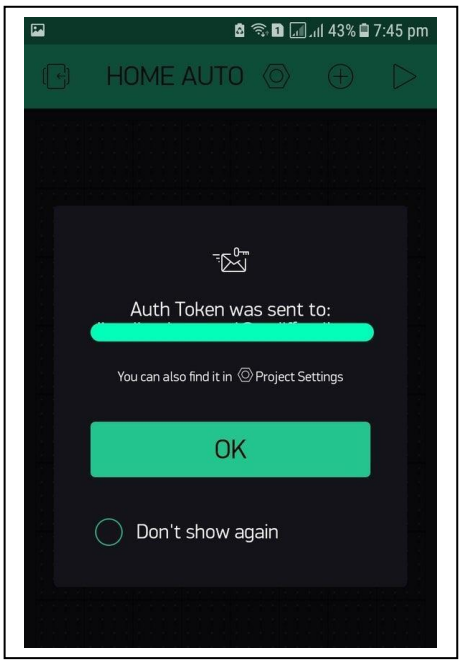
WORKING PRINCIPLE :

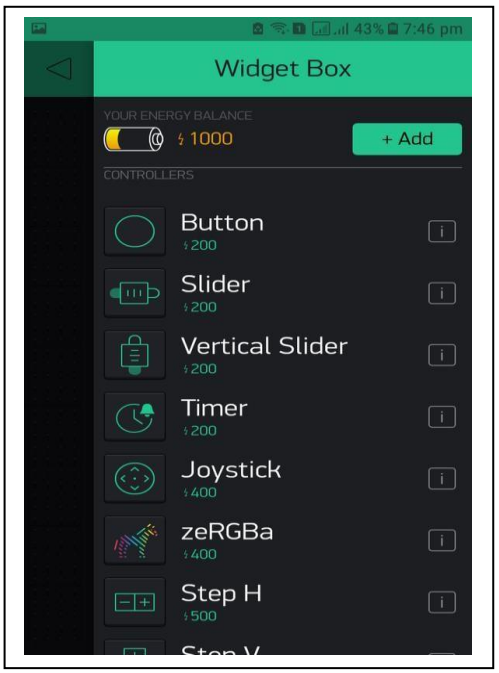


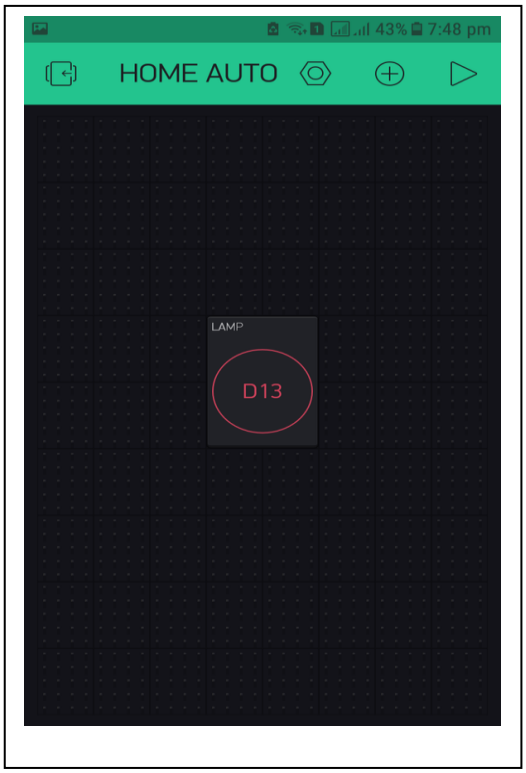


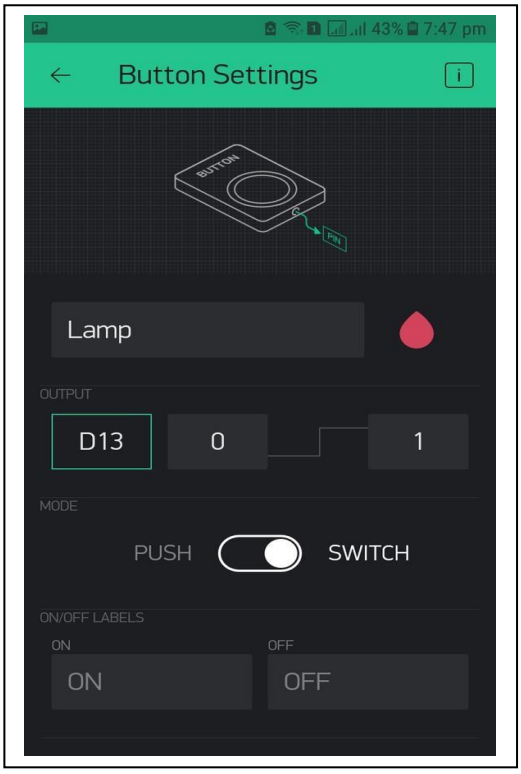


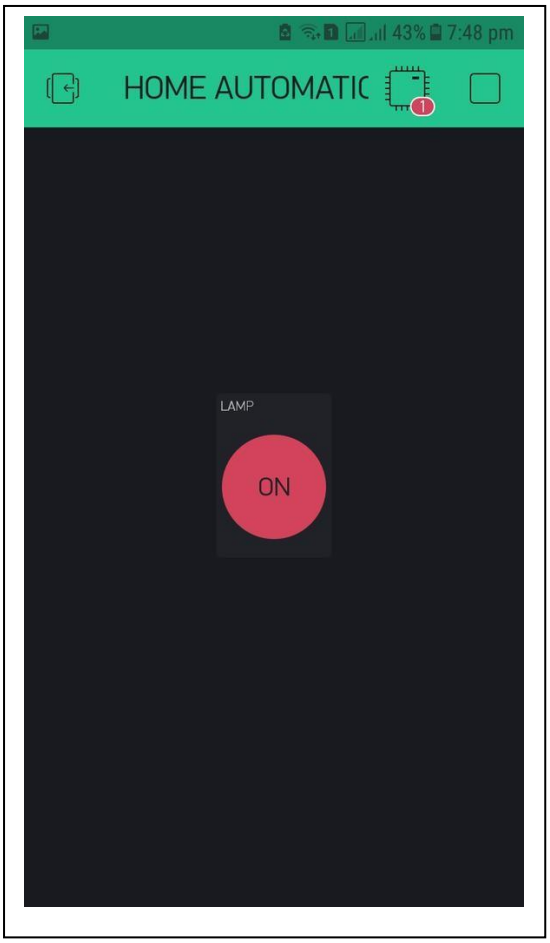


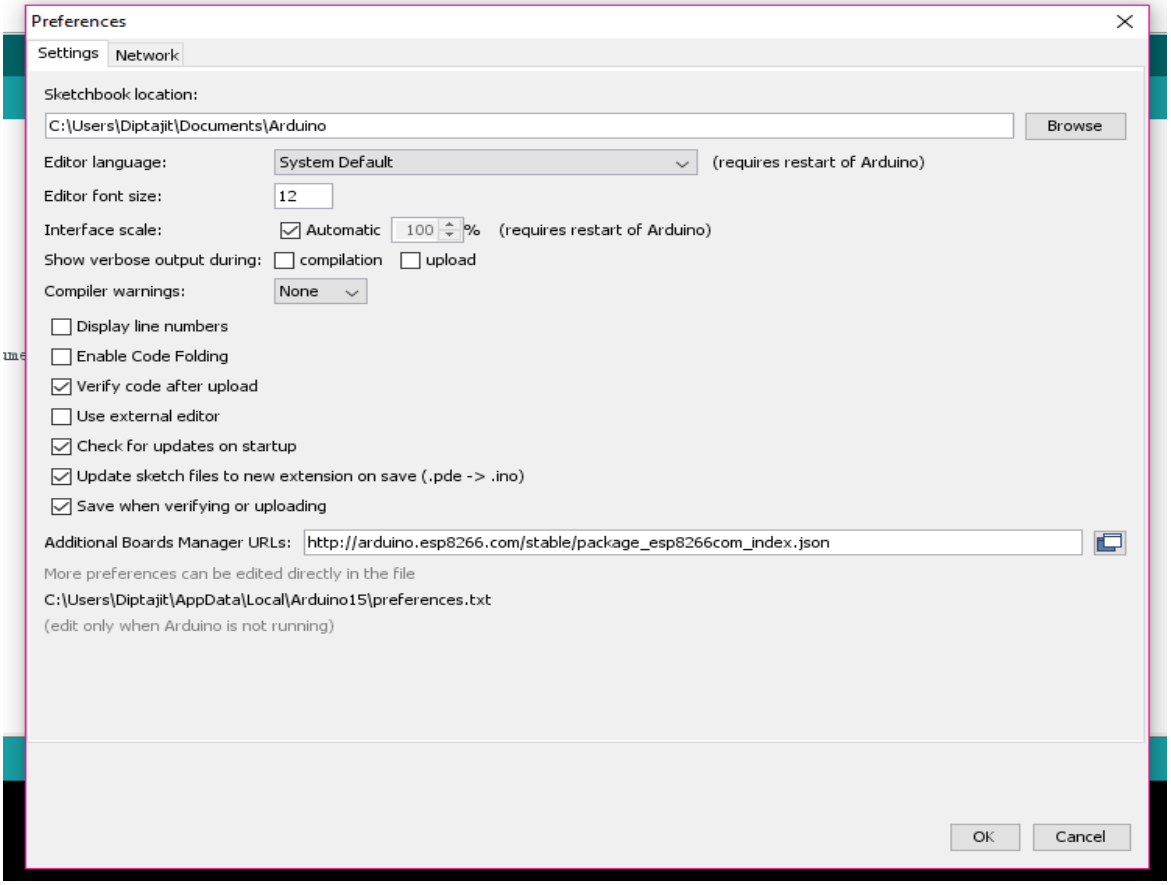

43% 7:45 pm
HOME AUTO
Auth Token was sent to:
You can also find it in Project Settings
OK
Don't show again

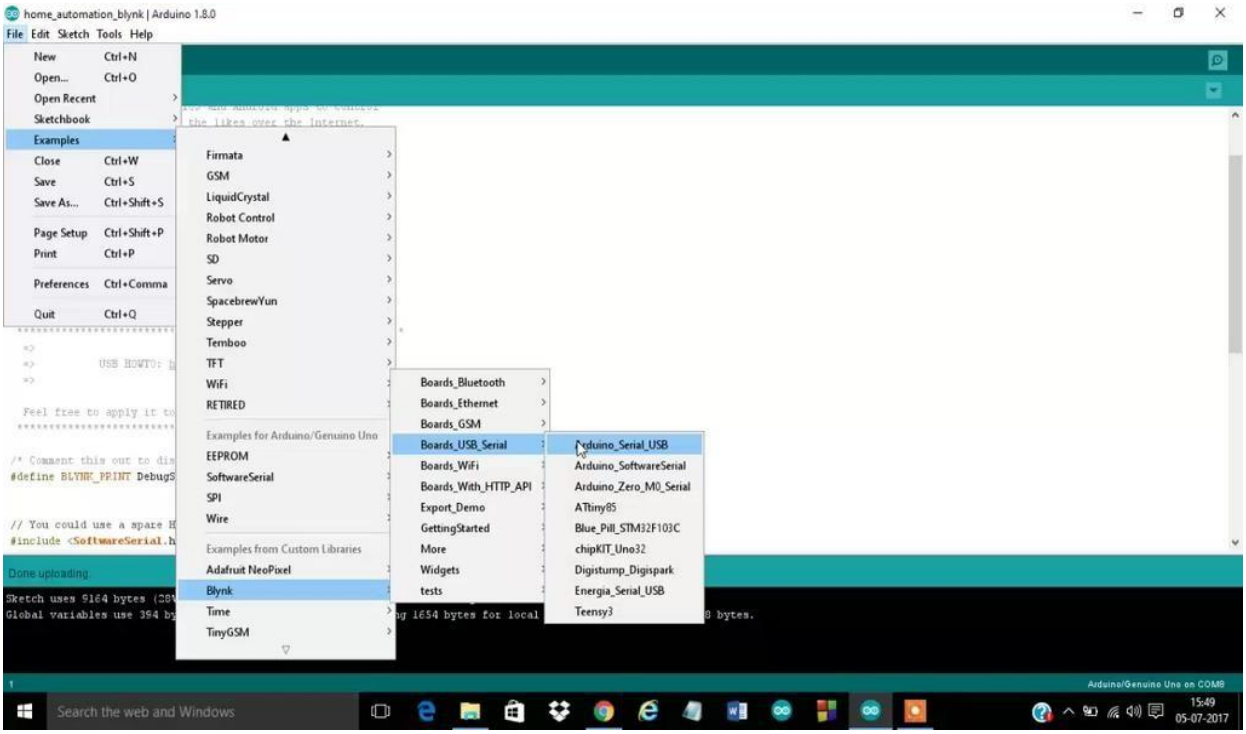

43% 7:46 pm
Widget Box
YOUR ENERGY BALANCE
1000
+ Add
CONTROLLERS
Button
200
Slider
200
Vertical Slider
200
Timer
200
Joystick
400
zeRGBa
400
Step H
500

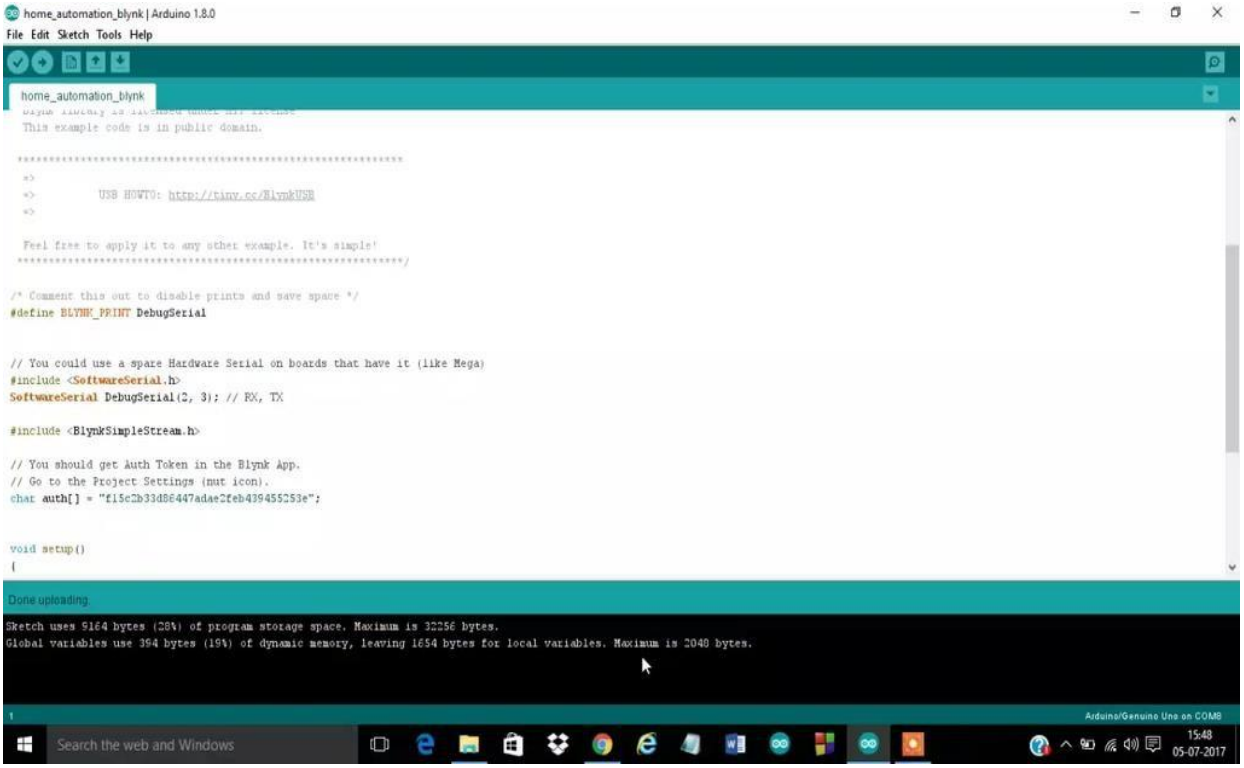

43% 7:48 pm
HOME AUTO
LAMP
D13

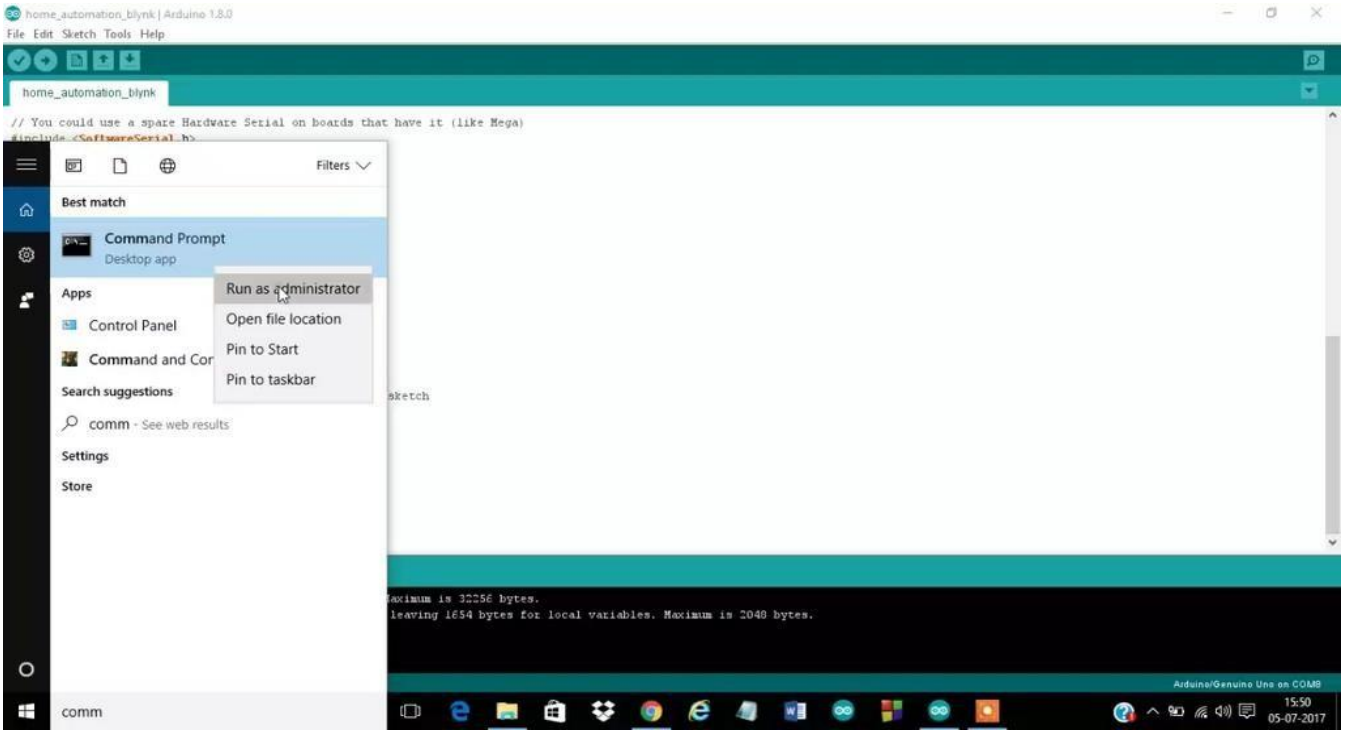

43% 7:47 pm
Button Settings
BUTTON
PIN
Lamp
OUTPUT
D13
0
1
MODE
PUSH
SWITCH
ON/OFF LABELS
ON
OFF
ON
OFF


43% 7:48 pm
HOME AUTOMATIC
LAMP
ON

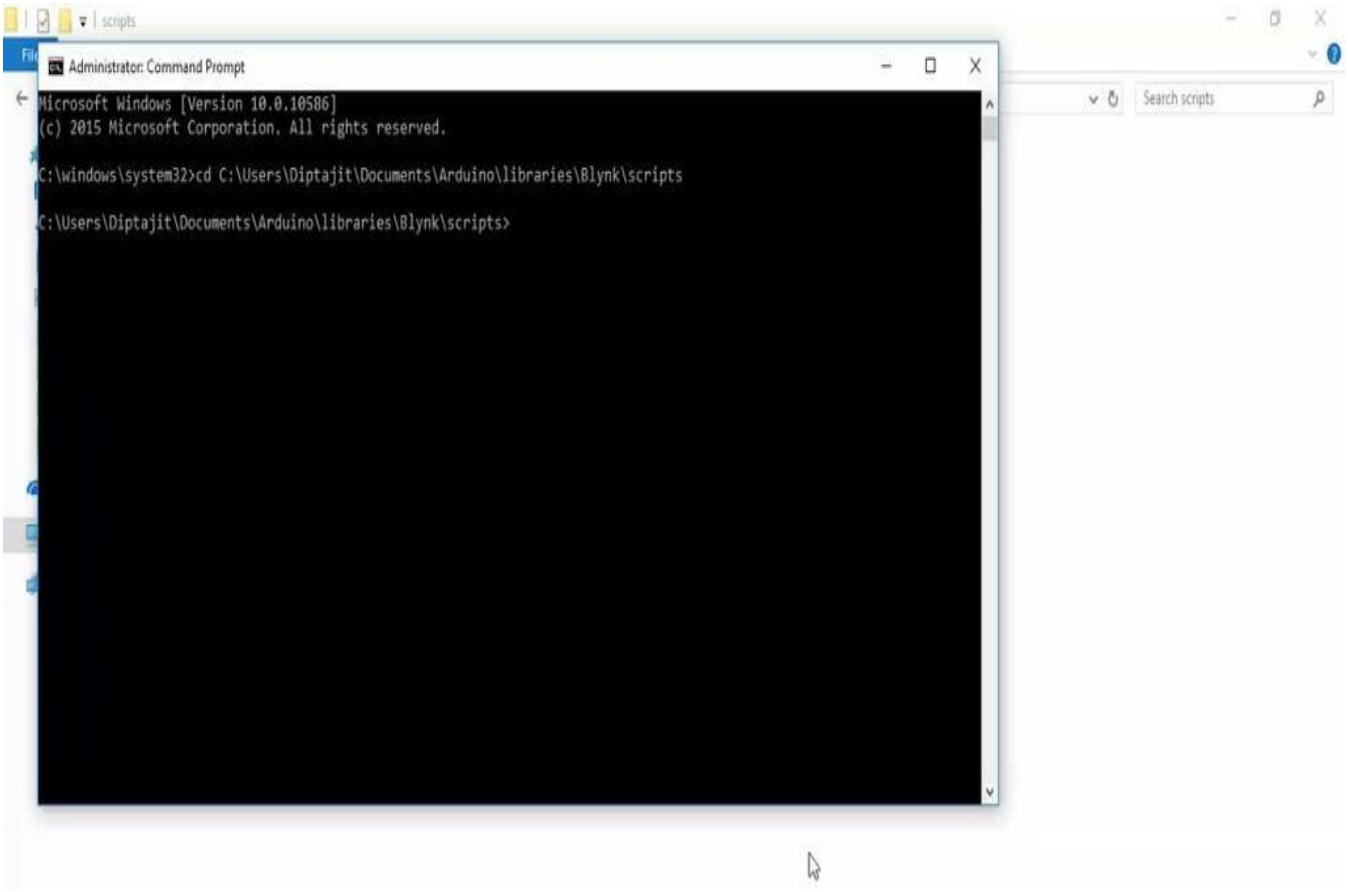

Preferences
Settings
Network
Sketchbook location:
C:\Users\Diptajit\Documents\Arduino
Browse
Editor language:
System Default
(requires restart of Arduino)
Editor font size:
12
Interface scale:
Automatic
100
%
(requires restart of Arduino)
Show verbose output during:
compilation
upload
Compiler warnings:
None
Display line numbers
Enable Code Folding
Verify code after upload
Use external editor
Check for updates on startup
Update sketch files to new extension on save (.pde -> .ino)
Save when verifying or uploading
Additional Boards Manager URLs:
http://arduino.esp8266.com/stable/package_esp8266com_index.json
More preferences can be edited directly in the file
C:\Users\Diptajit\AppData\Local\Arduino15\preferences.txt
(edit only when Arduino is not running)
OK
Cancel


home_automation_blynk | Arduino 1.8.0
File Edit Sketch Tools Help
New Ctrl+N
Open... Ctrl+O
Open Recent
Sketchbook
Examples
Close Ctrl+W
Save Ctrl+S
Save As... Ctrl+Shift+S
Page Setup Ctrl+Shift+P
Print Ctrl+P
Preferences Ctrl+Comma
Quit Ctrl+Q
Firmata
GSM
LiquidCrystal
Robot Control
Robot Motor
SD
Servo
SpacebrewYun
Stepper
Temboo
TFT
WiFi
RETIRED
Examples for Arduino/Genuino Uno
EEPROM
SoftwareSerial
SPI
Wire
Examples from Custom Libraries
Adafruit NeoPixel
Blynk
Time
TinyGSM
Boards_Bluetooth
Boards_Ethernet
Boards_GSM
Boards_USB_Serial
Boards_WiFi
Boards_With_HTTP_API
Export_Demo
GettingStarted
More
Widgets
tests
Arduino_Serial_USB
Arduino_SoftwareSerial
Arduino_Zero_M0_Serial
ATtiny85
Blue_Pill_STM32F103C
chipKIT_Uno32
Digistump_Digispark
Energia_Serial_USB
Teensy3
Done uploading.
Arduino/Genuino Uno on COM8
Search the web and Windows
15:49
05-07-2017


home_automation_blynk | Arduino 1.8.0
File Edit Sketch Tools Help
home_automation_blynk
This example code is in public domain.
=>
=> USB HOWTO: http://tinyc.cc/BlynkUSB
=>
Feel free to apply it to any other example. It's simple!
/* Comment this out to disable prints and save space */
#define BLYNK_PRINT DebugSerial
// You could use a spare Hardware Serial on boards that have it (like Mega)
#include <SoftwareSerial.h>
SoftwareSerial DebugSerial(2, 3); // RX, TX
#include <BlynkSimpleStream.h>
// You should get Auth Token in the Blynk App.
// Go to the Project Settings (nut icon).
char auth[] = "f15c2b33d86447adae2feb439455253e";
void setup()
{
Done uploading.
Sketch uses 5164 bytes (28%) of program storage space. Maximum is 32256 bytes.
Global variables use 394 bytes (19%) of dynamic memory, leaving 1654 bytes for local variables. Maximum is 2048 bytes.
Arduino/Genuino Uno on COM8
Search the web and Windows
15:48
05-07-2017


home_automation_blynk | Arduino 1.8.0
File Edit Sketch Tools Help
home_automation_blynk
// You could use a spare Hardware Serial on boards that have it (like Mega)
Filters
Best match
Command Prompt
Desktop app
Apps
Control Panel
Command and Cor
Search suggestions
comm - See web results
Settings
Store
Run as administrator
Open file location
Pin to Start
Pin to taskbar
comm
Arduino/Genuino Uno on COM8
15:50
05-07-2017


scripts
Share
View
C:\Users\Diptajit\Documents\Arduino\libraries\Blynk\scripts
Name
certs

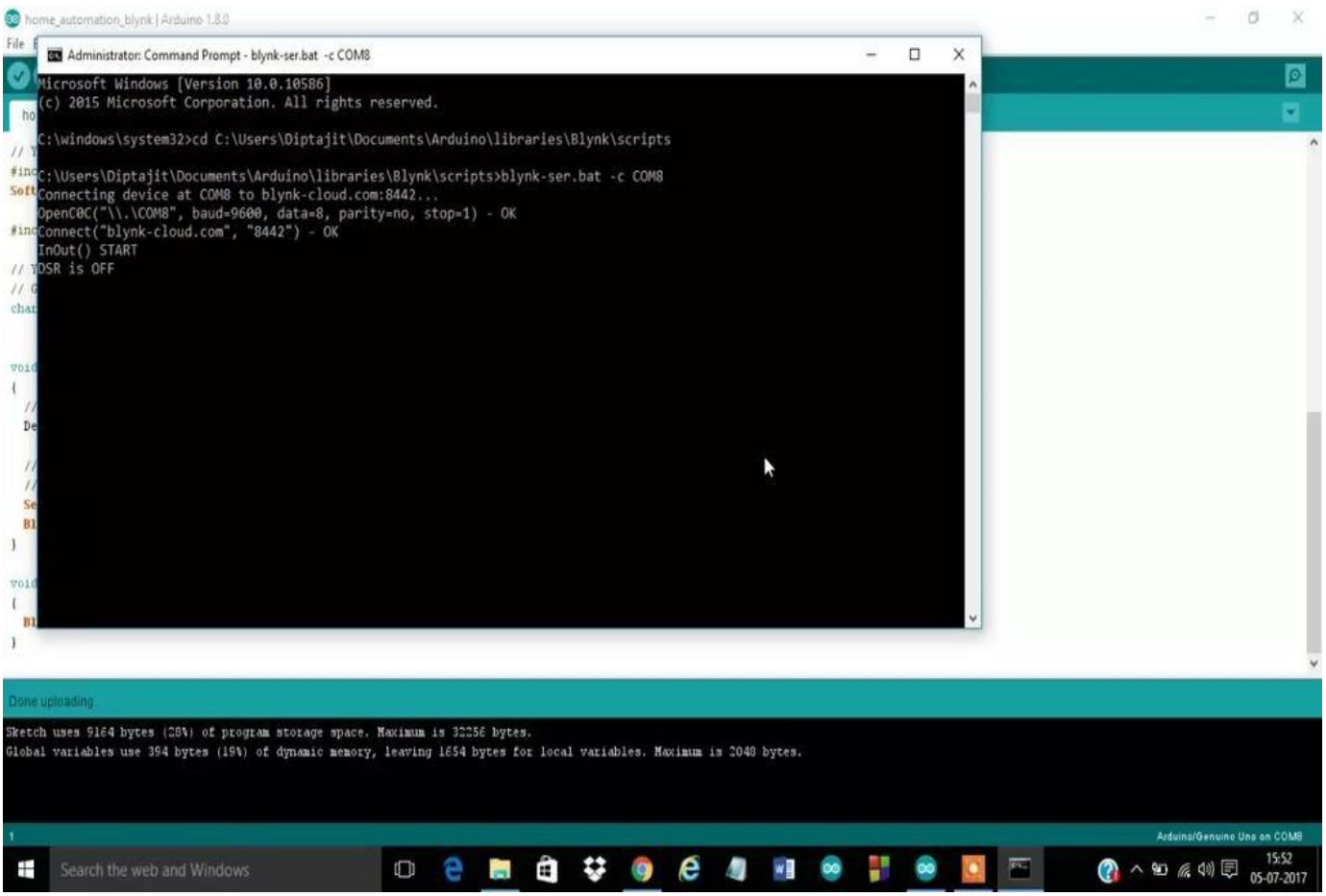


Write your path to blynk-ser.bat folder. For example:

```
cd C:\blynk-library-0.3.1\blynk-library-0.3.1\scripts
```

Run `blynk-ser.bat` file. For example : `blynk-ser.bat -c COM4` (where COM4 is port with your Arduino)

And press "Enter", press "Enter" and press "Enter"



Working of Relay module:

According to the diagram we can see that there is switch like thing inside the relay module whose one end is connected to COM i.e. Pin 4 and the other end is either connected between NO i.e. Pin 5 or NC i.e. Pin 6. When we are applying 0 V to the signal pin i.e. Pin 3 then the switch remains in NO position (normally open). When we apply +5 V to signal pin the switch drips from NO to NC (normally connected).

Creating the project in BLYNK App:

Download the BLYNK App from Google Playstore (link has been already given). Open it and you have to make an account there. After that click on "New Project". Now you have to click "CHOOSE DEVICE" and you will be asked to select required hardware, you will choose "Arduino UNO" and in "CONNECTION TYPE" you have to select "USB".You have to give a project name also. Then you click on "Create".Your project is now created and BLYNK will send an authorization token to your mail which you have to put in the arduino code.Then you will get a free space where you have to add buttons,graphs etc. You will get all these from the widget box. In this project as we are operating only one appliance so we will add only one button. After clicking on "Button" the icon will be added in the free space. You can plasce the button anywhere on the screen. Then you have to click on the button to customize it. You have to give a name there and you have to select whether you are using digital or analog ao virtual pin. You also have to mention the pin no. As in this project we are using D13 i.e. Digital pin 13 . Now select the mode whether "Push" or "Slide" , it depends upon you. After that return to the main screen ,you will see a play button on the right corner of the screen, you have to click on that to activate the project .If your system is ready and connected to internet then on mobile after clicking the play button it will show "Online" otherwise "Offline".

3. Code analysis and final connection :

First of all you have to add the following link in "additional boards manager URL" in preferences in the Arduino IDE. **Link :http://arduino.esp8266.com/stable/package_esp8266c...**

You have to go to the following link : https://github.com/blynkkk/blynk-library/releases/... and download the blynk library. After downloading the zip file you have to unzip it and copy the contents of the files(libraries and folders) to the sketchbook-folder of the Arduino IDE. To check whether the blynk library has been added or not restart the Arduino IDE and check in the library section , if you see **"Blynk"** it means that blynk library has been successfully added.

Just copy the code(already provided) or you can get the code from **Examples-->Blynk-->Boards_USB_Serials-->Arduino_Serial_USB**. In both cases the only change you have to make is that copy the authorization code sent to your mail to Arduino code. Don't upload the code now. Now open "Command Prompt" and run it as administration. A black screen will appear on the screen. Then you have to copy the path of **"scripts"** folder. In my case it is "My Documents\Arduino\libraries\Blynk\scripts" and paste it on the black screen and place enter. Then you have to copy and paste the .bat file in the black screen. The file is "blynk-ser.bat -c COM4" .You have to change the COM port number. In my case it was COM8 .Now upload the arduino code .Now come back to the command prompt part and press **"enter"** thrice. This will connect you to Blynk Server .

4. Control with Blynk App :

Now open blynk app from your mobile and open the project you have created. If your system is connected to Blynk server then you will see **'Online'** in your mobile otherwise you will see **'Offline'**. Now click on the button to On or Off the appliance. If it is not working then check whether the system is connected to the blynk server.

RESULT:

Thus an IOT based system was designed successfully.