EXP NO:1	8051 Assembly Language program using Keil simulator
DATE	

AIM:

To write 8051 Assembly Language Program for an 8-bit addition using Keil simulator and execute it.

SOFTWARE REQUIRED:

S.No	Software Requirements	Quantity
1	Keil µvision5 IDE	1

INTRODUCTION TO 8051 SIMULATORS:

A simulator is software that will execute the program and show the results exactly to the program running on the hardware, if the programmer finds any errors in the program while simulating the program in the simulator, he can change the program and re-simulate the code and get the expected result, before going to the hardware testing. The programmer can confidently dump the program in the hardware when he simulates his program in the simulator and gets the expected results.

8051 controller is a most popular 8-bit controller which is used in a large number of embedded applications and many programmers write programs according to their application. So testing their programs in the software simulators is a way. Simulators will help the programmer to understand the errors easily and the time taken for the testing is also decreased.

These simulators are very useful for students because they do need not to build the complete hardware for testing their program and validate their program very easily in an interactive way.

List of 8051 Simulators:

The list of simulators is given below with their features:

1. MCU 8051: MCU 8051 is an 8051 simulator that is very simple to use and has an interactive IDE (Integrated Development Environment). It is developed by Martin Osmera and most important of all is that it is completely free. There are many features for this IDE they are

- ✓ It supports both C and assembly language for compilation and simulation

- ✓ It has an in-built source code editor, graphical notepad, ASCII charts, Assembly symbol viewer, etc. It also supports several 8051 ICs like at89c51, A89S52, 8051, 8052, etc.
- ✓ It will support certain electronic simulations like LED, 7segment display, LCD etc. which will help in giving the output when you interface these things to the hardware directly.
- ✓ It has tools like hex decimal editors, base converters, special calculator, file converters, inbuilt hardware programmers, etc.
- ✓ It has syntax validation, pop base auto-completion etc.

You can download this tool from https://sourceforge.net/projects/mcu8051ide/files/.

2. EDSIM 51: This is a virtual 8051 interfaced with virtual peripherals like 7 segment display, motor, keypad, UART etc. This simulator is exclusively for students developed by James Rogers,. The features of this simulator are

- ✓ Have virtual peripherals like ADC, DAC with scope to display, comparator etc.
- ✓ Supports only assembly language
- ✓ IDE is completely written in JAVA and supports all the OS.
- ✓ Completely free and with user guide, examples, etc.

You can download this simulator from the https://www.edsim51.com/index.html.

3. 8051 IDE: This simulation software is exclusively for the Windows operating system (98/xp). The features of this simulator are

- ✓ Text editor, assembler, and software simulate in one single program.
- ✓ Has facilities like Breakpoint setter, execute to break point, predefined simulator watch window, etc.
- ✓ It is available in both free version and paid version.

You can download this tool from https://www.acebus.com/win8051.htm

4. KEIL µVision: KEIL is the most popular software simulator. It has many features like interactive IDE and supports both C and assembly languages for compilation and simulation.
You can download and get more information from https://www.keil.com/c51/.

INSTALLATION OF KEIL SOFTWARE

Set up Keil IDE for Programming

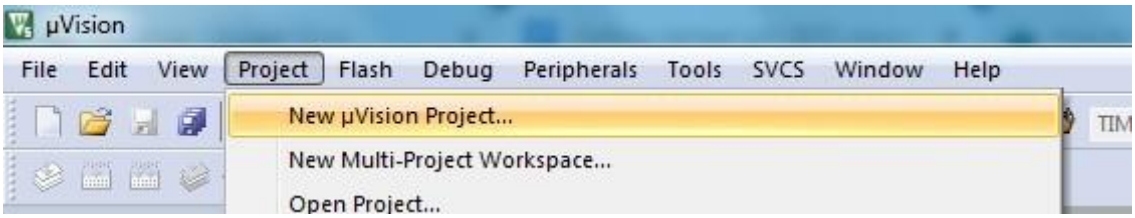
Keil µVision IDE is a popular way to program MCUs containing the 8051 architectures. It supports over 50 microcontrollers and has good debugging tools including logic analyzers and watch windows.

In this article, we will use the AT89C51ED2 microcontroller, which has:

- 64 KB FLASH ROM
- On-chip EEPROM
- 256 Bytes RAM
- In-System programming for uploading the program
- 3 Timer/counters
- SPI, UART, PWM

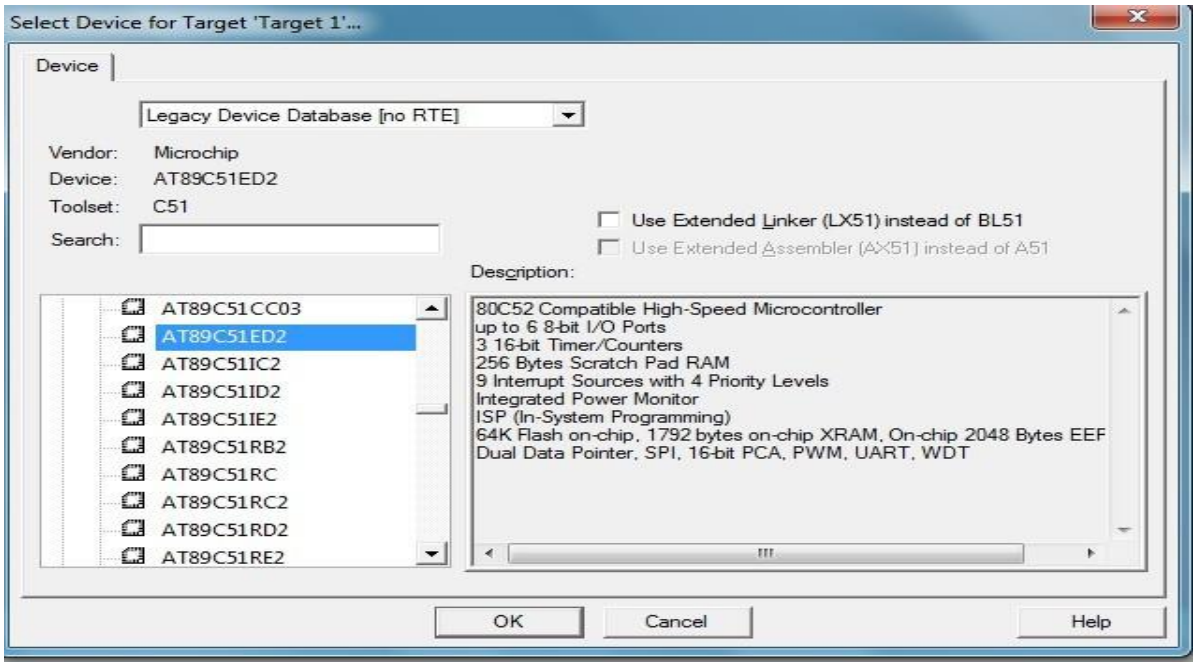
The Keil µVision icon.

To start writing a new program, you need to create a new project. Navigate to **project —> New µVision project**. Then save the new project in a folder.

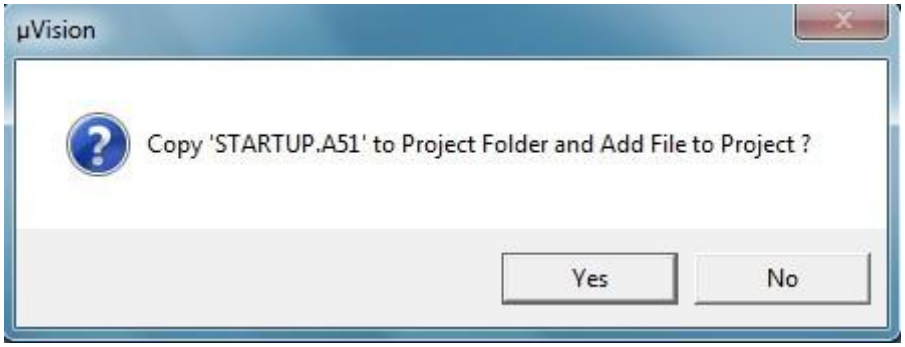


After saving the file, a new window will pop up asking you to select your microcontroller.

As discussed, we are using AT89C51/AT89C51ED2/AT89C52, so select this controller under the Microchip section (as Atmel is now a part of Microchip).

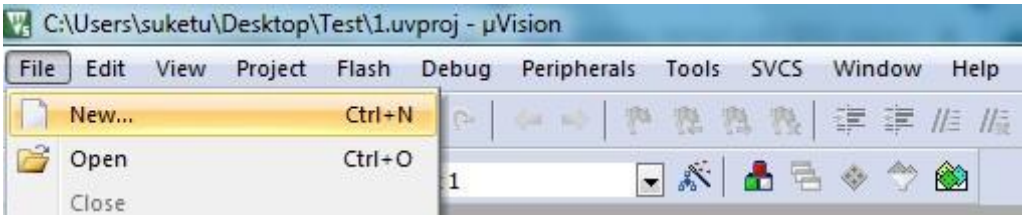


Select '**Yes**' in the next pop-up, as we do not need this file in our project.

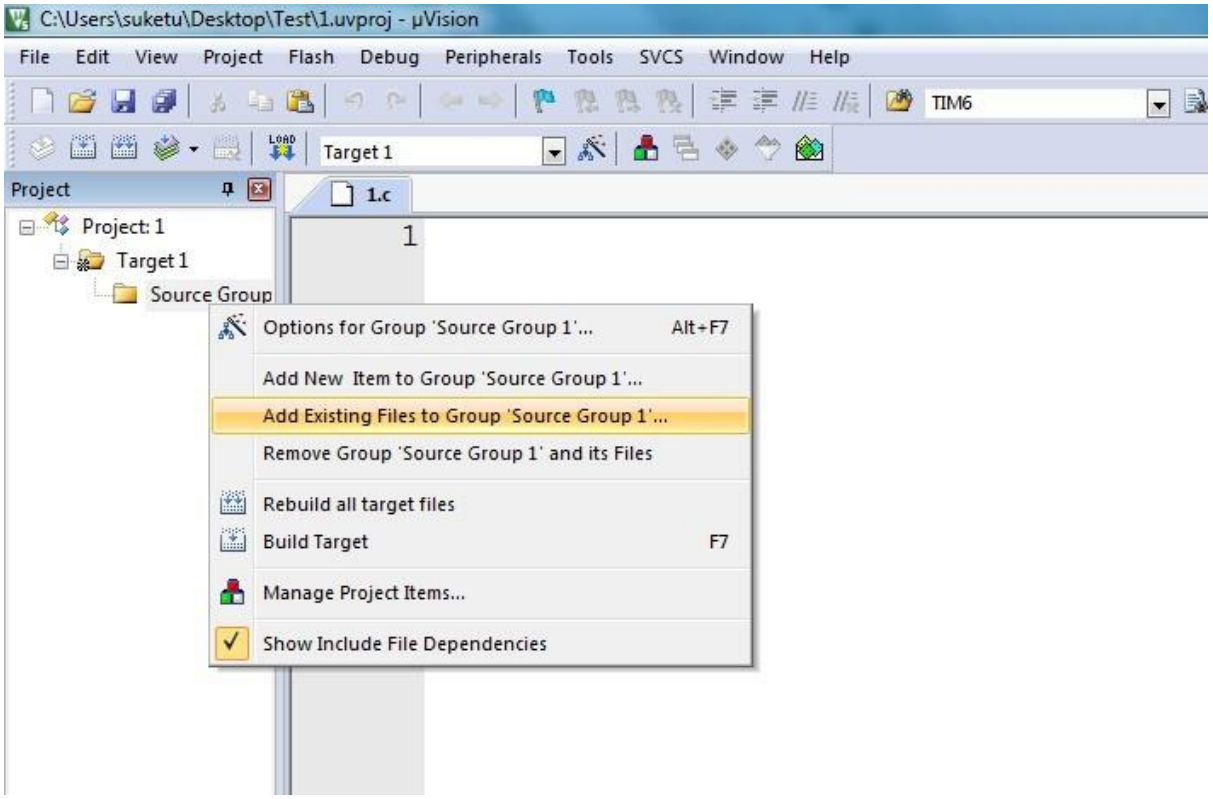


Our project workspace is now ready!

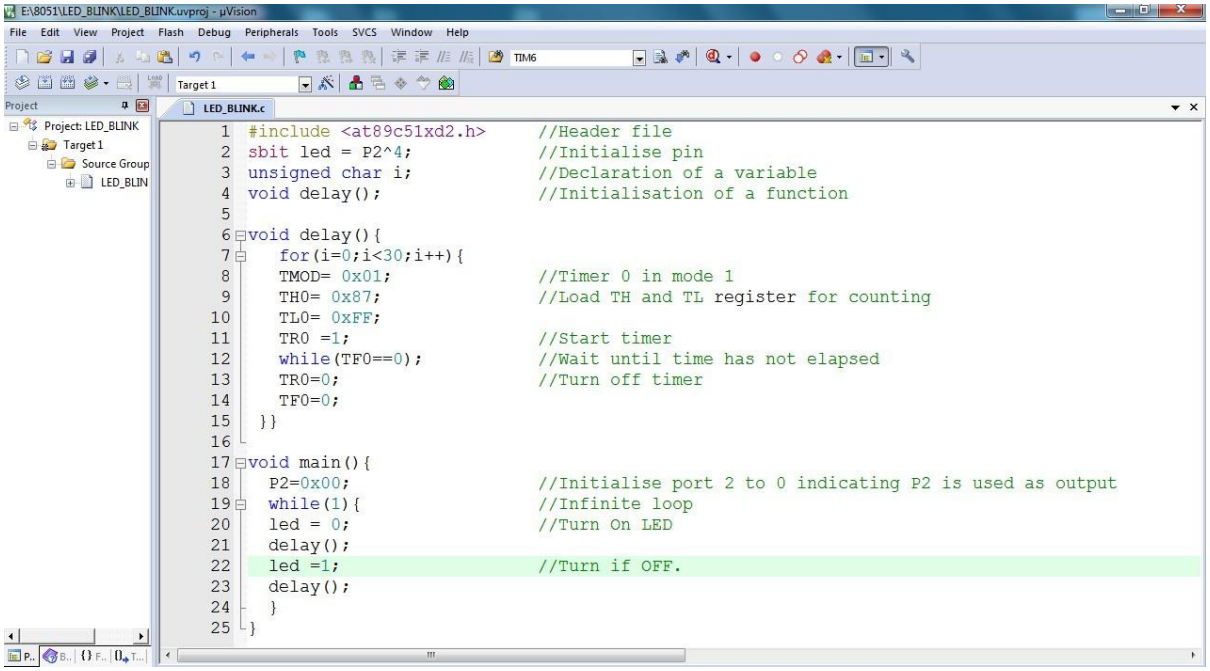
From here, we need to create a file where we can write our C code. Navigate to **File —> New**. Once the file is created, save it with .c extension in the same project folder.



Next, we have to add that .c or .asm file to our project workspace. Select **Add Existing** Files and then select the created .c or .asm file to get it added.



The workspace and project file are ready.



PROCEDURE

1. Create a new project, go to “Project” and close the current project “Close Project”.
2. Next Go to the Project New µVision Project and Create New Project Select Device for Target.
3. Select the device AT89C51ED2 or AT89C51 or AT89C52
4. Add Startup file Next go to “File” and click “New”.
5. Write a program on the editor window and save it with .asm extension.
6. Add this source file to Group and click on “Build Target” or F7.
7. Go to debugging mode to see the result of simulation by clicking Run or step run.8.

PROGRAM:

```
ORG 00H
SJMP START
ORG 30H
START:
MOV A, #03H
MOV R0, #03H
ADD A,R0
MOV R1,A
END
```

OUTPUT:

The sum of **03H + 03H** is stored in **R1 (06H).**

RESULT:

EXP NO:2	Test data transfer between registers and memory
DATE	

AIM:

To write and execute an Assembly language program to transfer data between registers and memory.

SOFTWARE REQUIRED:

S.No	Software Requirements	Quantity
1	Keil µvision5 IDE	1

PROCEDURE

1. Create a new project, go to “Project” and close the current project “Close Project”.
2. Next Go to the Project New µVision Project and Create a New Project Select Device for the Target.
3. Select the device AT89C51ED2 or AT89C51 or AT89C52
4. Add Startup file Next go to “File” and click “New”.
5. Write a program on the editor window and save it with .asm extension.
6. Add this source file to Group and click on “Build Target” or F7.
7. Go to debugging mode to see the result of the simulation by clicking Run or Step run.

PROGRAM:

```
ORG 00H
MOV R0,#30H
MOV R1,#40H
MOV R3,#05
MOV A,@R0
UP: MOV @R1,A
INC R0
INC R1
DJNZ R1,UP
END
```

OUTPUT

The contents of memory from **30H to 34H** are copied to **40H to 44H.**

Result

EXP NO:3	ALU operations
DATE	

AIM:

To write and execute the ALU program using the Keil simulator.

SOFTWARE TOOLS REQUIRED:

S.No	Software Requirements	Quantity
1	Keil µvision5 IDE	1

PROCEDURE

1. Create a new project, go to “Project” and close the current project “Close Project”.
2. Next Go to the Project New µVision Project and Create New Project Select Device for Target.
3. Select the device AT89C51ED2 or AT89C51 or AT89C52
4. Add Startup file Next go to “File” and click “New”.
5. Write a program on the editor window and save it with .asm extension.
6. Add this source file to Group and click on “Build Target” or F7.
7. Go to debugging mode to see the result of simulation by clicking Run or step run.

PROGRAM:

```
// ARTHAMETIC AND LOGICAL OPERATION

// ADDITION
MOV A,#25H
MOV B,#12H
ADD A,B
MOV 40H,A

// SUBTRACTION
MOV A,#25H
SUBB A,B
MOV 41H,A

// MULTIPLICATION
MOV A,#25H
MUL AB
MOV 42H,A
MOV 43H,B

// DIVISION
MOV A,#25H
MOV B,#12H
DIV AB
MOV 44H,A
MOV 45H,B

//AND OPERATION
MOV A,#25H
MOV B,#12H
ANL A,B
MOV 46,A

// OR OPERATION
MOV A,#25H
MOV B,#12H
ORL A,B
MOV 47,A
LCALL 0003H
END
```

OUTPUT:

- **Addition:** 25H + 12H = 37H → Stored in 40H
- **Subtraction:** 25H - 12H = 13H → Stored in 41H
- **Multiplication:** 25H × 12H = 0288H → Stored in 42H, 43H
- **Division:** 25H ÷ 12H = 02H (quotient) & 01H (remainder) → Stored in 44H, 45H
- **AND Operation:** 25H & 12H = 00H → Stored in 46H
- **OR Operation:** 25H | 12H = 37H → Stored in 47H

RESULT:

EXP NO:4	WRITE BASIC PROGRAMS USING EMBEDDED C
DATE	

AIM:

To write a basic embedded C program to control a port 0 pin 0 connected to an 8051 microcontroller using a Keil simulator.

SOFTWARE REQUIRED:

S.No	Software Requirements	Quantity
1	Keil µvision5 IDE	1

PROCEDURE

1. Create a new project, go to "Project" and close the current project "Close Project".
2. Next Go to the Project New µVision Project and Create a New Project Select Device for the Target.
3. Select the device AT89C51ED2 or AT89C51 or AT89C52
4. Add Startup file Next go to "File" and click "New".
5. Write a program on the editor window and save it with .asm extension.
6. Add this source file to Group and click on "Build Target" or F7.
7. Go to debugging mode to see the result of the simulation by clicking Run or Step run.

PROGRAM:

1. **Embedded C Addition**

```
#include <reg51.h>
void main(void)
{
unsigned char X,Y,Z;
X =0x12;
Y =0x34;
P0 =0x00;
Z =X+Y;
P0 = Z;
}
```

Expected Output: 12H + 34H = 46H

2.**Embedded C Subtraction**

```
#include <reg51.h>
void main(void)
{
unsigned char X,Y,Z;
X =0x12;
```

```
Y =0x34;
P0 =0x00;
Z =X-Y;
P0 = Z;
}
```

Output: 34H - 12H = 22H

3.**Embedded C Multiplication**

```
#include <reg51.h>
void main(void)
{
unsigned char X,Y,Z;
X =0x12;
Y =0x34;
P0 =0x00;
Z =XxY;
P0 = Z;
}
```

Output: 12H × 02H = 24H

4.**Embedded C Division**

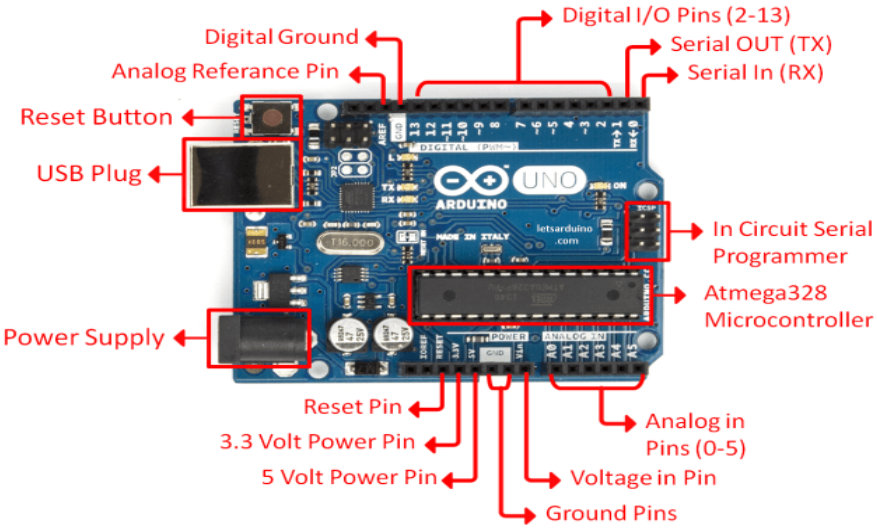
```
#include <reg51.h>
void main(void)
{
unsigned char X,Y,Z;
X =0x12;
Y =0x34;
P0 =0x00;
Z =X/Y;
P0 = Z;
while (1);
}
```

Output: 24H ÷ 12H = 02H

Result

EXP.NO : 5 Introduction to Arduino Platform and Programming

HARDWARE DETAILS



Advantages :

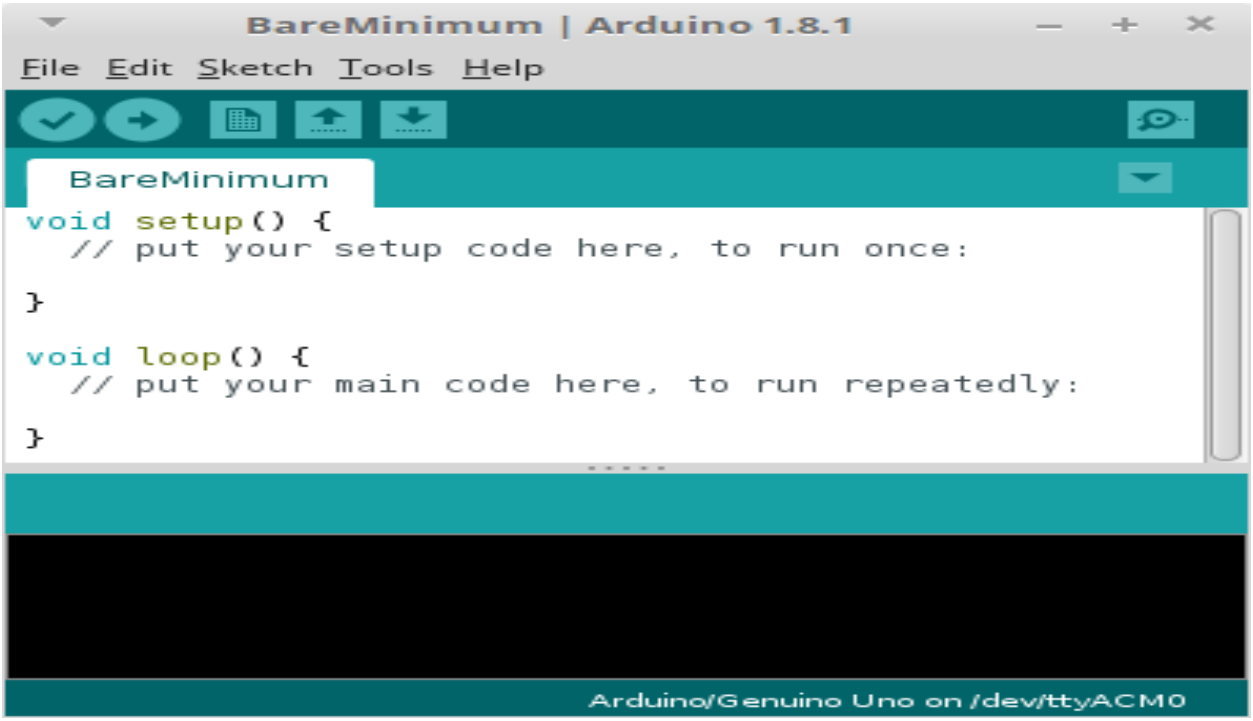
- It is cheap
- It comes with an open supply hardware feature that permits users to develop their own kit
- The software of the Arduino is well-suited with all kinds of in operation systems like Linux, Windows, and Macintosh, etc.
- It also comes with open supply software system feature that permits tough software system developers to use the Arduino code to merge with the prevailing programing language libraries and may be extended and changed.
- For beginners also it is very simple to use.

SOFTWARE DETAILS

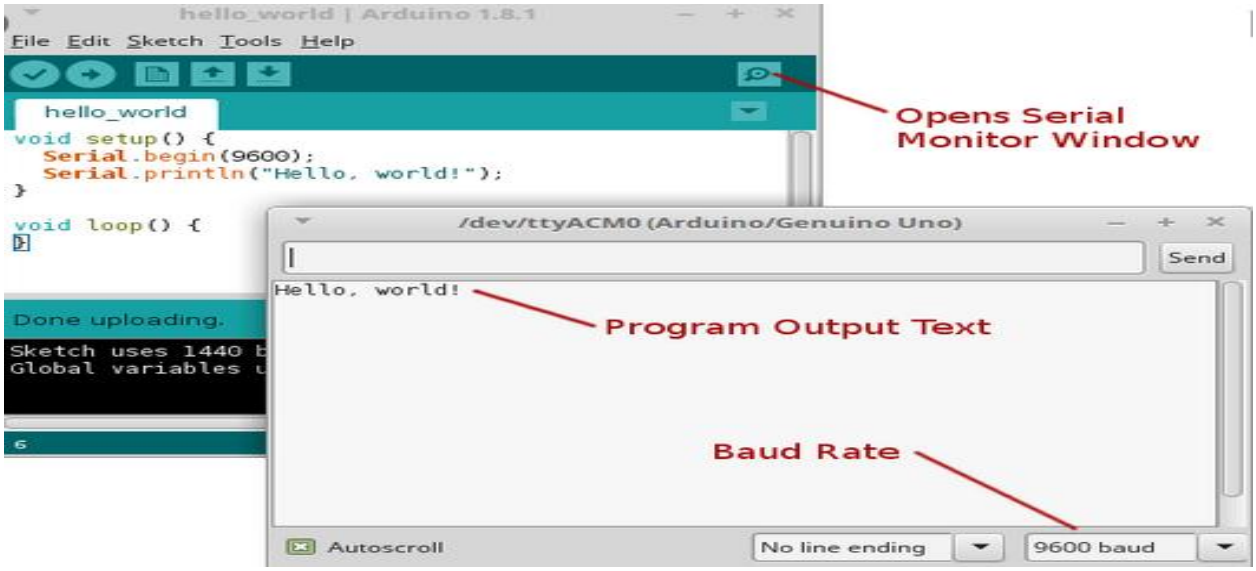
ARDUINO IDE

Platform

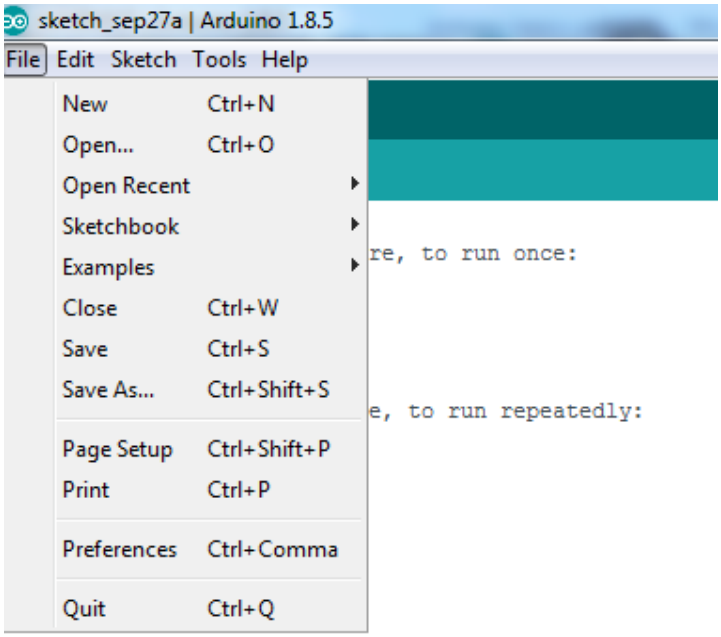
Programming Structure:



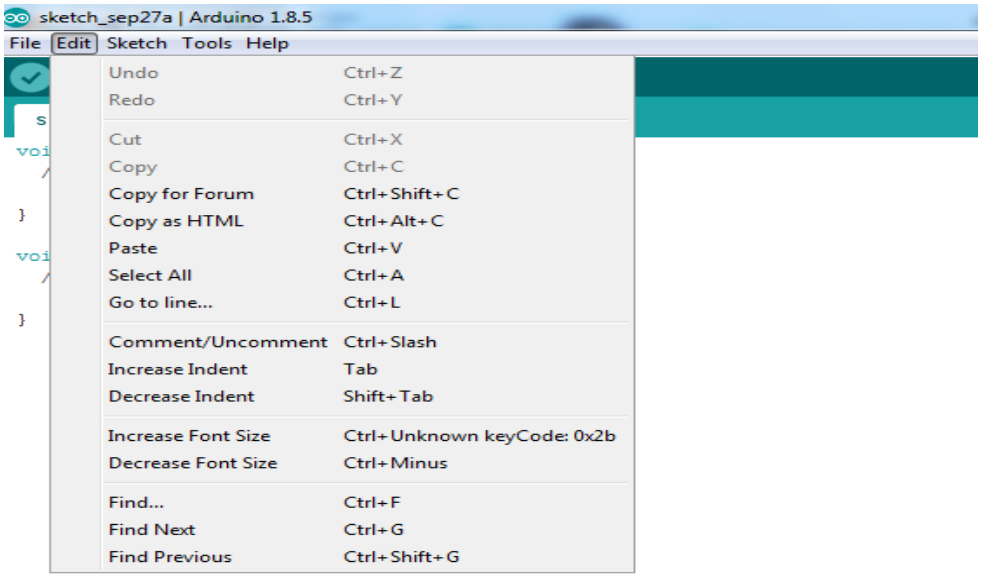
Programming :


hello_world | Arduino 1.8.1
File Edit Sketch Tools Help
hello_world
void setup() {
Serial.begin(9600);
Serial.println("Hello, world!");
}
void loop() {
Done uploading.
Opens Serial
Monitor Window
/dev/ttyACM0 (Arduino/Genuino Uno)
Send
Hello, world!
Program Output Text
Baud Rate
Autoscroll
No line ending
9600 baud

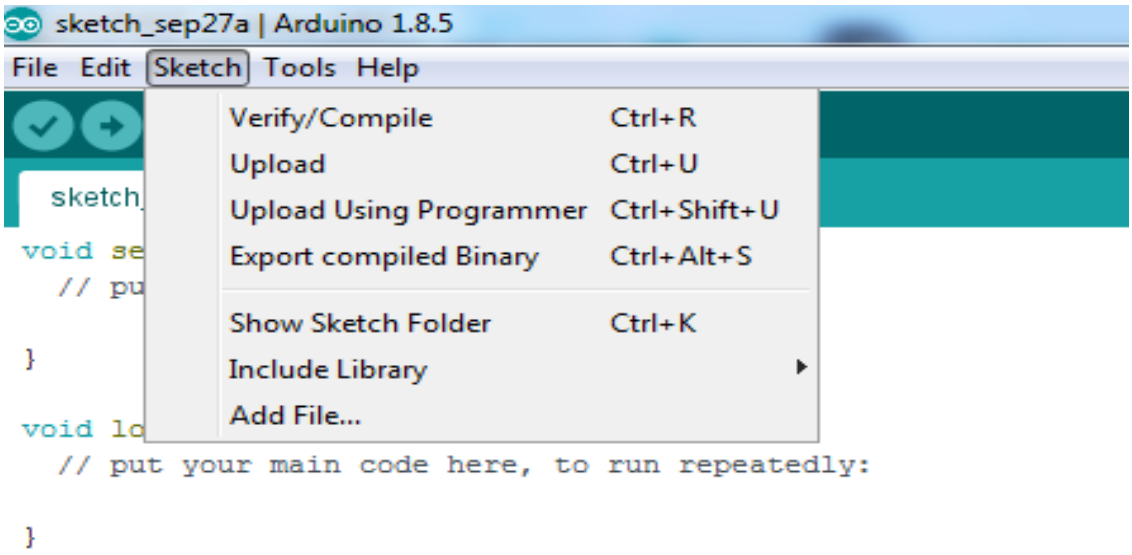
File :


sketch_sep27a | Arduino 1.8.5
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
re, to run once:
e, to run repeatedly:

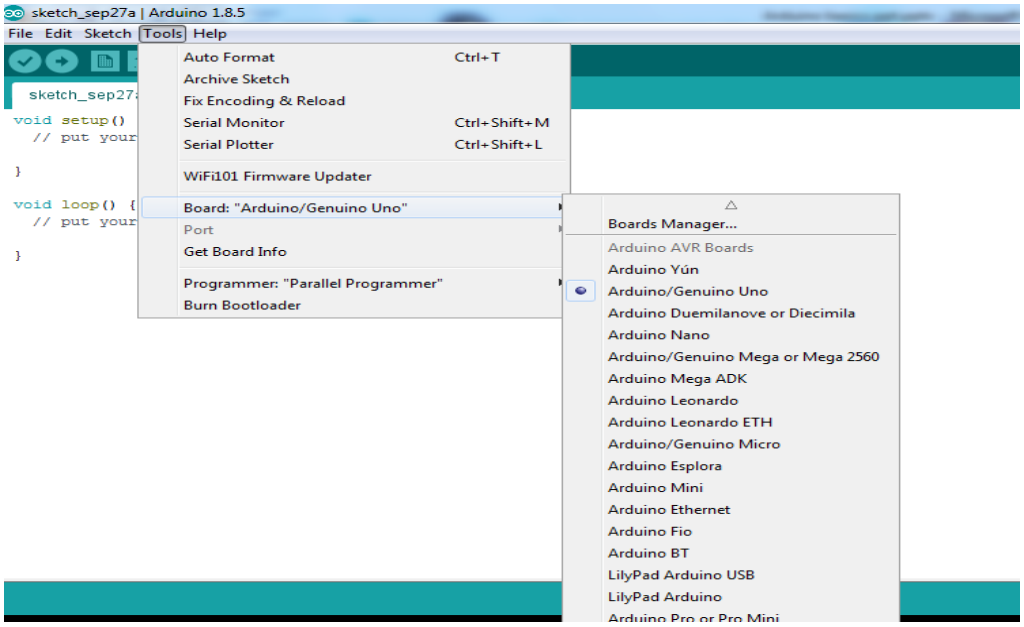
Edit :



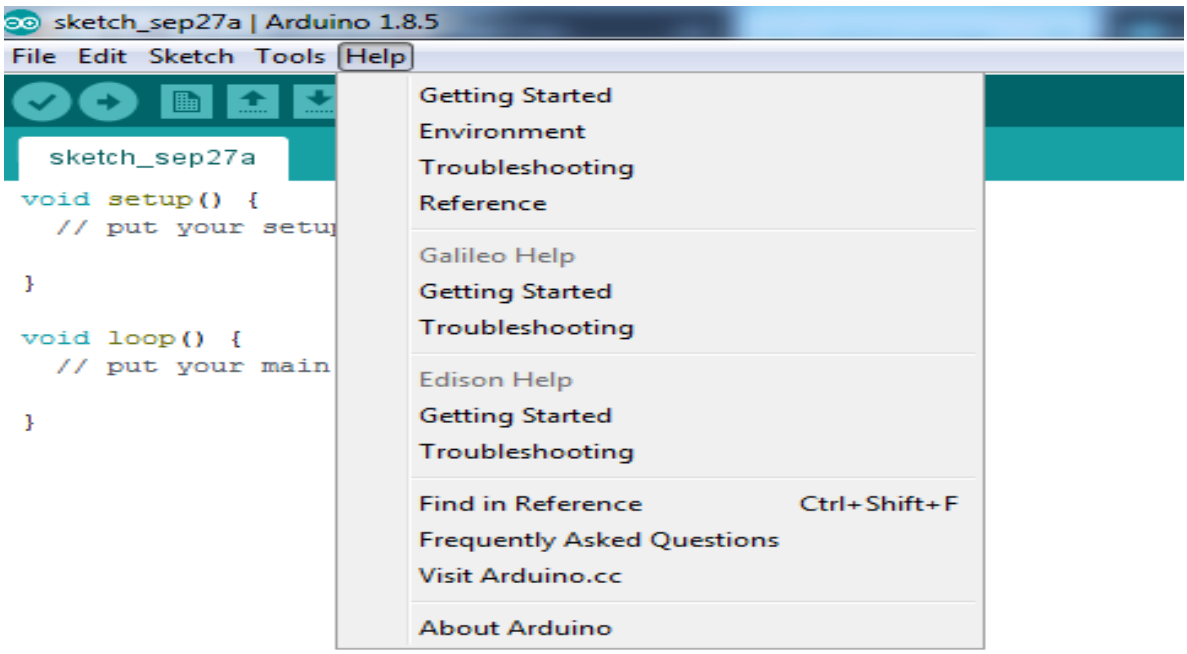
Sketch :



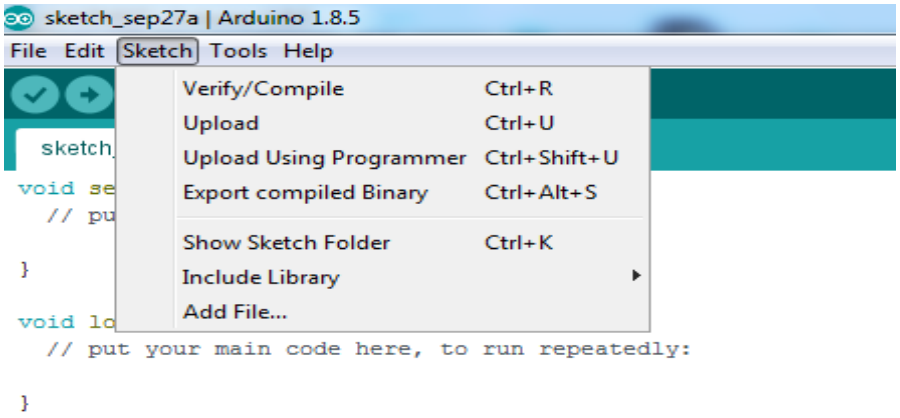
Tools :



Help :



Compile & Upload programs :



Commands

```
#include <Wire.h>


void setup()
{
 pinMode(3, OUTPUT);
 pinMode(2, INPUT);

 Serial.begin(9600);
}
```

```
if digitalRead(x==1)
digitalWrite(2, LOW);
digitalWrite(2, HIGH);
delay(40);


 Serial.println(distance);
 Serial.println("distance");


val = analogRead(potPin);
val = map(val, 0, 1023, 0, 255);
analogWrite (ledPin, val);
```

EXP.NO : 5.a **LED BLINKING**

Aim :

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.

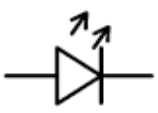


Figure. LED symbol

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 :

- 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.
- Connect the USB connector to the USB of Arduino board and the computer system.
- Using this program, the first LED (left most LED) is switched on for 0.5 sec and then it is switched off.
- 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.
- 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 :

```
int LED_PIN = 8;

void setup()
{
   pinMode(LED_PIN, OUTPUT);
}

void loop()
{
 digitalWrite(LED_PIN, HIGH);
 delay(1000);
 digitalWrite(LED_PIN, LOW);
 delay(1000);
}
```

Output

1. **LED connected to pin 8 will turn ON for 1 second**
2. **LED will then turn OFF for 1 second**
3. **This process will repeat indefinitely**

Result:

EXP.NO : 5b **LED pattern**

Aim :

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

PROCEDURE :

- 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.
- Connect the USB connector to the USB of Arduino board and the computer system.
- The 4 LEDs are used as 4 bits. To represent 0-15 decimal number in binary, 4 bits are needed. The program is written such that in one for loop, 0-15 is counted and after every count, the bits of the number is read one by one and displayed in the LEDs. In this way, binary pattern of 0-15 decimal number is displayed in 4 LEDs

Program

```
// 8 LEDs used to display binary pattern of decimal numbers 0-15
// For loop is written to generate a decimal number 0-15. Byte variable is used.
// For every count, the first 4 bits are read from the byte representation of the number
// 8 bits are assigned to LEDs. Thereby, binary pattern of decimal number 0-15 is
displayed

int t = 40;
int rnd =5;
int pat1t =75;

void setup()
{
  for(int i=3; i<=12; i++) // pin d3 to d11 used
  pinMode(i,OUTPUT);
```

```
}

void loop(){

  for(int i=0; i<=rnd; i++) {
  pat1();}
  for(int i=0; i<=rnd; i++){
  pat2();}
  for(int i=0; i<=rnd; i++) {
  pat3();}
  for(int i=0; i<=rnd; i++) {
  pat4();}
  for(int i=0; i<=rnd; i++){
  pat5();}
  for(int i=0; i<=rnd; i++) {
  pat6();}
  for(int i=0; i<=rnd; i++) {
  pat7();}
  for(int i=0; i<=rnd; i++) {
  pat8();}
  for(int i=0; i<=rnd; i++) {
  pat9();}


}

void pat1(){
   for(int i=3; i<=12; i++) {
     digitalWrite(i,HIGH);
     delay(pat1t);
     digitalWrite(i,LOW);

   }

   for(int i=11; i>=4; i--) {
     digitalWrite(i,HIGH);
     delay(pat1t);
     digitalWrite(i,LOW);
   }
}
void pat2(){
    for(int i=3; i<=12; i++) {
     digitalWrite(i,HIGH);
     digitalWrite(i-1,HIGH);
     digitalWrite(i+1,HIGH);
     delay(100);
     digitalWrite(i,LOW);
     digitalWrite(i-1,LOW);
     digitalWrite(i+1,LOW);
   }
```

```
   for(int i=11; i>=4; i--) {
     digitalWrite(i,HIGH);
     digitalWrite(i-1,HIGH);
     digitalWrite(i+1,HIGH);
     delay(100);
     digitalWrite(i,LOW);
     digitalWrite(i-1,LOW);
     digitalWrite(i+1,LOW);
   }
}
void pat3(){
  for(int i=3; i<=12; i=i+2) {
     digitalWrite(i,HIGH);
     delay(100);
     digitalWrite(i,LOW);

   }

   for(int i=12; i>=3; i=i-2) {
     digitalWrite(i,HIGH);
     delay(100);
     digitalWrite(i,LOW);
   }
}
void pat4(){
     for(int i=3; i<=12; i++) {
     digitalWrite(i,HIGH);
     delay(100);
   }

   for(int i=12; i>=2; i--) {
     digitalWrite(i,HIGH);
     delay(100);
     digitalWrite(i,LOW);
   }
}
void pat5(){
    for(int i=3; i<=12; i++) {
     digitalWrite(i,HIGH);
     }
   delay(100);
   for(int i=3; i<=12; i++) {
     digitalWrite(i,LOW);
     }
   delay(100);
}
void pat6(){
      for(int i=3; i<=8; i++) {
     digitalWrite(i,HIGH);
```

```
    }
   for(int i=8; i<=12; i++) {
    digitalWrite(i,LOW);
    }
   delay(200);
   for(int i=3; i<=8; i++) {
    digitalWrite(i,LOW);
    }
   for(int i=8; i<=12; i++) {
    digitalWrite(i,HIGH);
    }
   delay(200);
}
void pat7(){
     for(int i=3; i<=12; i=i+2) {
    digitalWrite(i,HIGH);
    }
   for(int i=4; i<=12; i=i+2) {
    digitalWrite(i,LOW);
    }
   delay(200);
    for(int i=3; i<=12; i=i+2) {
    digitalWrite(i,LOW);
    }
   for(int i=4; i<=12; i=i+2) {
    digitalWrite(i,HIGH);
    }
   delay(200);
}
void pat8(){
   digitalWrite(7,HIGH);
   digitalWrite(8,HIGH);
   delay(t);
   digitalWrite(7,LOW);
   digitalWrite(8,LOW);
   delay(t);
   digitalWrite(6,HIGH);
   digitalWrite(9,HIGH);
   delay(t);
   digitalWrite(6,LOW);
   digitalWrite(9,LOW);
   delay(t);
   digitalWrite(5,HIGH);
   digitalWrite(10,HIGH);
   delay(t);
   digitalWrite(5,LOW);
   digitalWrite(10,LOW);
   delay(t);
   digitalWrite(4,HIGH);
   digitalWrite(11,HIGH);
```

```
   delay(t);
   digitalWrite(4,LOW);
   digitalWrite(11,LOW);
   delay(t);
   digitalWrite(3,HIGH);
   digitalWrite(12,HIGH);
   delay(t);
   digitalWrite(3,LOW);
   digitalWrite(12,LOW);
   delay(t);
    digitalWrite(4,HIGH);
   digitalWrite(11,HIGH);
   delay(t);
   digitalWrite(4,LOW);
   digitalWrite(11,LOW);
   delay(t);
   digitalWrite(5,HIGH);
   digitalWrite(10,HIGH);
   delay(t);
   digitalWrite(5,LOW);
   digitalWrite(10,LOW);
   delay(t);
   digitalWrite(6,HIGH);
   digitalWrite(9,HIGH);
   delay(t);
   digitalWrite(6,LOW);
   digitalWrite(9,LOW);
   delay(t);
}

void pat9(){
   for(int i=3; i<=12; i++) {
     digitalWrite(i,HIGH);
   }
    for(int i=3; i<=12; i++) {
     digitalWrite(i,LOW);
     delay(100);
     digitalWrite(i,HIGH);
   }
   for(int i=11; i>=4; i--) {
     digitalWrite(i,LOW);
     delay(100);
     digitalWrite(i,HIGH);
   }

}
```

Output

8 LEDs are connected to **pins 3 to 12**, they will:

1. Sequentially turn ON/OFF.
2. Blink in groups.
3. Move in patterns across the LED strip.
4. Flash in interesting sequences.

Result:

EXP.NO 5.c. LED Pattern with Push Button control Arduino

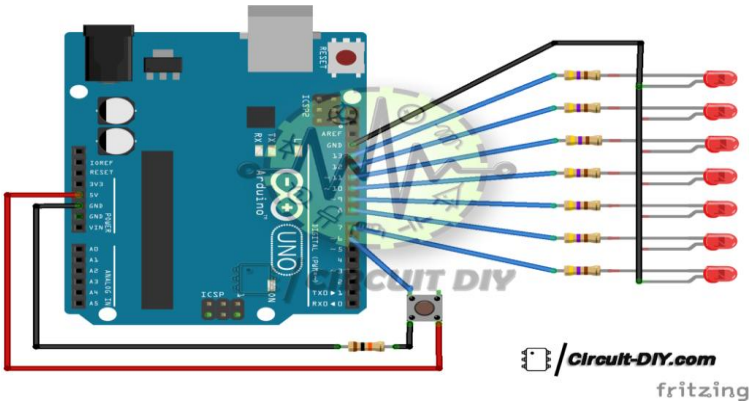
Aim :

Perform Interfacing of LED Pattern with Push Button Control 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:



Usually each push button switch has two pairs of contacts. Each pair of contacts consists of a NO contact and a NC contact. When the button is pressed, the two pairs of contacts act simultaneously, the NC contact is disconnected, and the NO contact is closed.

A 'push to make' switch allows electricity to flow between its two contacts when held in. When the button is released, the circuit is broken. This type of switch is also known as a Normally Open (NO) Switch. (Examples: doorbell, computer case power switch, calculator buttons, individual keys on a keyboard)

A 'push to break' switch does the opposite, i.e. when the button is not pressed, electricity can flow, but when it is pressed the circuit is broken. This type of switch is also known as a Normally Closed (NC) Switch. (Examples: Fridge Light Switch, Alarm Switches in Fail-Safe circuits)

Many Push switches are designed to function as both 'push to make' and 'push to break' switches. For these switches, the wiring of the switch determines whether the switch functions as a 'push to make' or as a 'push to break' switch.

PROCEDURE :

- There are 4 switches and 4 LEDs. The program is written such that when the particular switch is pressed, the LED just above the switch will glow. Though the program is lengthy, the program is simple.
- Connect the LEDs D4, D5, D6 and D7 to digital pins 4, 5, 6, 7 of Arduino. Similarly, connect Sw8, Sw9, Sw10 and Sw11 to digital pins 8,9,10,11 of Arduino. Connect +5V and Gnd from Arduino board.

Program

```
// Program to switch on the LED when the corresponding switch is pressed
// variable declaration and initialization
//Switches and LEDs are assigned to specific digital i/o pins
const int buttonPin = 8;     // the number of the pushbutton pin
const int ledPin =  4;       // the number of the LED pin
```

```
// variables will change:
int buttonState = 0;         // variable for reading the pushbutton status

void setup() {
  // initialize the LED pin as an output:
  pinMode(ledPin, OUTPUT);
  // initialize the pushbutton pin as an input:
  pinMode(buttonPin, INPUT);
}

void loop() {
  // read the state of the pushbutton value:
  buttonState = digitalRead(buttonPin);

  // check if the pushbutton is pressed. If it is, the buttonState is HIGH:
  if (buttonState == LOW) {
    // turn LED on:
    digitalWrite(ledPin, HIGH);
  } else {
    // turn LED off:
    digitalWrite(ledPin, LOW);
  }
}
```

Output

Button State	**LED State**
Pressed (LOW)	☼ ON
Not Pressed (HIGH)	■ OFF

EXP.NO : 5d. Switch and LED pattern

The switch S9 is used to increment a counter from 0-15. For every count value, the binary pattern is displayed in the 4 LEDs. Connect the LEDs D4, D5, D6 and D7 to digital pins 4, 5, 6, 7 of Arduino. Similarly, connect Sw8, Sw9, Sw10 and Sw11 to

digital pins 8,9,10,11 of Arduino. Connect +5V and Gnd from Arduino board.

Program

```
//D9 switch is used to increment a counter 0-15
// The 4 LEDs shows the binary pattern of the number for every count

int ledPin[] = {4,5,6,7};                    //  ledPins are assigned to digital pins as array
variable
int SW = 9;                                  // SW 9 is assigned to digital pin 9
int i = 0;                                   // variable used for a counter
void setup()
{
 for (int i =0;i<4;i++)
 {    pinMode(ledPin[i], OUTPUT);       // ledPins are configured as OUTPUT
 }
  pinMode(SW,INPUT);              // Switch is configured as INPUT
}
void loop()
{
 if(digitalRead(SW))    {                    //  switch status is read
                                             // if it is 1, then counter variable is incremented
   i++;
 displayBinary(i);                           // displayBinary function is called
 delay(500);
 }
 if(i>15)   {                          // the counter is counted from 0 to 15 again

  i=0;
 }
 }
void displayBinary(byte number)          // displayBinary user defined function
{
 for (int i =0;i<4;i++)  {
    if (bitRead(number,i)==1)  {
       digitalWrite(ledPin[i], HIGH);
  }
  else  {
       digitalWrite(ledPin[i], LOW);
  }
 }
}
```

Output (LED Behavior)

Decimal (i)	Binary (4 LEDs)	LED Pattern (4,5,6,7)
0	0000	☐☐☐☐ (All OFF)
1	0001	☐☐☐☼
2	0010	☐☐☼☐
3	0011	☐☐☼☼

Decimal (i)	Binary (4 LEDs)	LED Pattern (4,5,6,7)
4	0100	☐ 🔆 ☐ ☐
...	...	...
15	1111	🔆 🔆 🔆 🔆 (All ON)

Result:

EXP.NO : 5. E. **Displaying Hello world message in LCD**

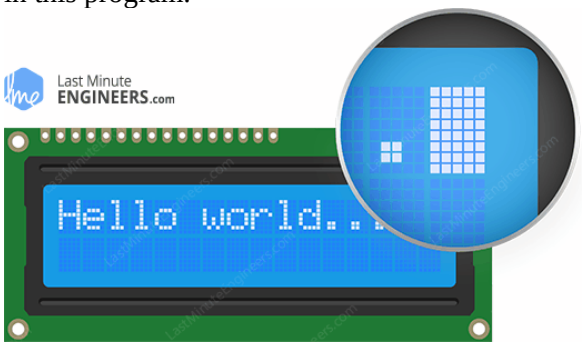
Aim :

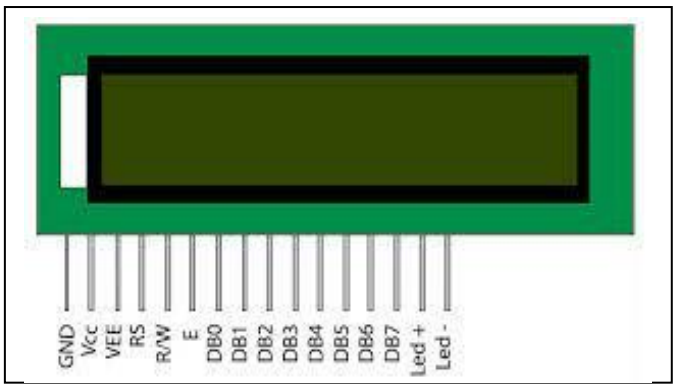
APPARATUS REQUIRED :

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

THEORY :

In this experiment, 16 characters (columns) and 2 lines (rows) LCD is used.
In the internal library of Sketch software, there is a library called LiquidCrystal. It is defined as LiquidCrystal.h
This header file has to be used in the beginning of the program. This library has got many functions that can be used in the program. We are going to see only a few functions of the LiquidCrystal.h in this program.





Program

```
/*
  LiquidCrystal Library - Hello World

 Demonstrates the use a 16x2 LCD display.  The LiquidCrystal
 library works with all LCD displays that are compatible with the
 Hitachi HD44780 driver. There are many of them out there, and you
 can usually tell them by the 16-pin interface.

 This sketch prints "Hello World!" to the LCD
 and shows the time.

  The circuit:
 * LCD RS pin to digital pin 12
 * LCD Enable pin to digital pin 11
 * LCD D4 pin to digital pin 4
 * LCD D5 pin to digital pin 5
 * LCD D6 pin to digital pin 6
 * LCD D7 pin to digital pin 7
 * LCD R/W pin to ground
```

```
 * LCD VSS pin to ground
 * LCD VCC pin to 5V
 * 10K resistor:
 * ends to +5V and ground
 * wiper to LCD VO pin (pin 3)
*/

#include <LiquidCrystal.h>

// initialize the library by associating any needed LCD interface pin
// with the arduino pin number it is connected to
const int rs = 12, en = 11, d4 = 4, d5 = 5, d6 = 6, d7 = 7;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

void setup() {
  // set up the LCD's number of columns and rows:
  lcd.begin(16, 2);
  // Print a message to the LCD.
  lcd.print("hello, world!");

}

void loop() {
  // set the cursor to column 0, line 1
  // (note: line 1 is the second row, since counting begins with 0):
 // lcd.setCursor(0, 1);
  // print the number of seconds since reset:
 // lcd.print(millis() / 1000);
 }
```

Output

hello, world!

Result

EXP.NO : 5.f **RC Servo position control**

Aim :

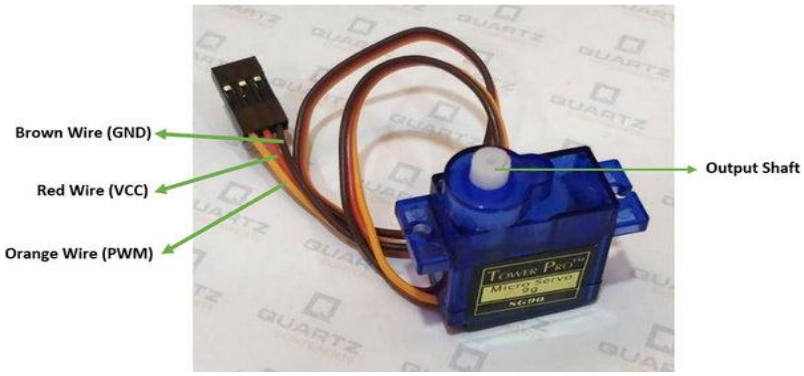
Perform Interfacing of RC Servo Motor with Arduino board and evaluate the response of variations.

APPARATUS REQUIRED :

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

THEORY :

RC servo motor can be used to set the position of the shaft of the servo motor to 0 to 180 degrees. 50 Hz frequency pulses (Period 20msec) with pulse width (0.5 to 2.5msec) positions the servo between 0 degrees and 180 degrees.
There is a built in servo library in sketch software. It is used by writing #include Servo.h as the first line in the Arduino program. In this experiment, potentiometer is used to output 0-5V analog voltage.



The analog voltage is read in A0 pin using analogRead function. It converts the 0-5V analog voltage to 10 bit digital outputs (10 bit resolution) and digital output with decimal value 0-1023. This read value is sent out from the Arduino as pwm signal in pin D9. The 0-1023 value is

changed to the output of 0-255. If the analog converted value is 1023, then the output will have full on time pulse width 255. If the analog converted value is 500, then the output will have on time pulse width of (1023/255) x 500. Since the pulses are continuous, the average output will be dc voltage depending upon the analog voltage input.
However, a map function to convert 0-1023 to 0-180 is used in the program. Servo.write() function is used to give control input to the servo motor 0-180 degrees.

Program

```
//Small servo motor is tested for 0 to 180 degrees positioning
// Library Servo is included in this program
// Servo motor is operated by giving the angle information through the arduino pin 9
// as pwm signal which goes from 0 to 255, but it is mapped to 0-180
// Potentiometer voltage input is used to provide angle information

    #include <Servo.h>

    int servoPin = 9;
    Servo servo;
    int angle = 0;  // servo position in degrees

    void setup() {
      servo.attach(servoPin);
    }

    void loop() {

      // scan from 0 to 180 degrees
      for(angle = 0; angle < 180; angle++)
      {
        servo.write(angle);
        delay(15);
      }

      // now scan back from 180 to 0 degrees
      for(angle = 180; angle > 0; angle--)
      {
        servo.write(angle);
        delay(15);
      }
}
```

Output

- **The servo motor will continuously rotate from 0° to 180° and then back to 0°.**
- **It moves slowly (due to a 15ms delay) for a smooth transition.**

Result

EXP.NO : 5 .g. **Temperature measurement using LM35 Thermistor IC**

Aim :

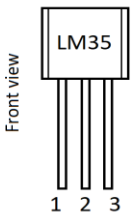
Perform Interfacing of temperature sensor experiment with Arduino and evaluate the response of variations.

APPARATUS REQUIRED :

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

THEORY:

Temperature is the most-measured process variable in industrial automation. Most commonly, a temperature sensor is used to convert temperature value to an electrical value. Temperature Sensors are the key to read temperatures correctly and to control temperature in industrials applications.



LM35 Pinout:

1. +Vs
2. Vout
3. GND

A large distinction can be made between temperature sensor types. Sensors differ a lot in properties such as contact-way, temperature range, calibrating method and sensing element. The temperature sensors contain a sensing element enclosed in housings of plastic or metal. With the help of conditioning circuits, the sensor will reflect the change of environmental temperature.

In the temperature functional module we developed, we use the LM34 series of temperature sensors. The LM34 series are precision integrated-circuit temperature sensors, whose output voltage is linearly proportional to the Fahrenheit temperature. The LM34 thus has an advantage over linear temperature sensors calibrated in degrees Kelvin, as the user is not required to subtract a large constant voltage from its output to obtain convenient Fahrenheit scaling. The LM34 does not require any external calibration or trimming to provide typical accuracies of ±1.2°F at room temperature and ±11.2°F over a full -50 to +300°F temperature range. The LM34 is rated to operate over a -50° to +300°F temperature range.

LM 35 IC is a 3 pin IC. It has Vcc and Gnd pins and another output pin. +5V and Gnd are connected to Vcc and Gnd pins. The output pin gives the output 10mV per degree centigrade. For example, if the ambient temperature is 30°C, then the output pin gives 300mV dc output. The output pin is connected to A0 analog input of Arduino. Arduino uses 10 bit analog to digital converter. The 10 bit resolution is 1024. 0-5V input is read as 0-1023 inside the Arduino in this analog input A0. There are A0, A1, A2, A3, A4 and A5 analog input pins available in Arduino. Connect the +5V and Gnd and A0 of Arduino to this LM 35 sensor section.
In this program, LCD is used to display the temperature.

Program

```
// Temperature is measured using sensor LM35.
// It gives 10mV output per degree centigrade
// since adc has 10 bit resolution, maximum voltage is 5V
// 10 bit resolution, number 1023 is output for 5V
// for one bit change, voltage required is 4.88mV
// 10mV/4.88mV = 2.05. Use this factor to display degree centigrade
// LCD is used to display temperature
#include<LiquidCrystal.h>
LiquidCrystal lcd(12, 11, 4, 5, 6, 7);
const int sensor=A1; // Assigning analog pin A1 to variable 'sensor'
float tempc;  //variable to store temperature in degree Celsius
float tempf;  //variable to store temperature in Fahreinheit
float vout;  //temporary variable to hold sensor reading
void setup()
{
pinMode(sensor,INPUT); // Configuring pin A1 as input
Serial.begin(9600);
lcd.begin(16,2);
  delay(500);
}
void loop()
{
vout=analogRead(sensor);
vout=(vout*500)/1023;
```

```
tempc=vout; // Storing value in Degree Celsius
tempf=(vout*1.8)+32; // Converting to Fahrenheit
lcd.setCursor(0,0);
lcd.print("in DegreeC= ");
lcd.print(tempc);
lcd.setCursor(0,1);
lcd.print("in Fahrenheit=");
lcd.print(tempf);
delay(1000); //Delay of 1 second for ease of viewing in serial monitor
}
```

Output

in DegreeC= 27.5
in Fahrenheit= 81.5

Result

EXP.NO : 5.h. Distance measurement using ultrasonic sensor

Aim :

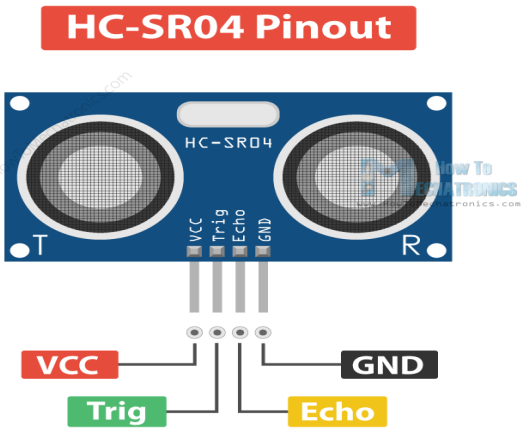
Perform Interfacing of temperature sensor experiment with Arduino and evaluate the response of variations.

APPARATUS REQUIRED :

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

THEORY :

An ultrasonic sensor is an electronic device that measures the distance of a target object by emitting ultrasonic sound waves, and converts the reflected sound into an electrical signal. Ultrasonic waves travel faster than the speed of audible sound (i.e. the sound that humans can hear).



Ultrasonic sensor HC-SR04 sensor is used. A burst of ultrasonic waves is transmitted. It hits the object and return back in the form of echo. Time difference of outgoing waves and returning echo is proportional to twice the distance travelled. HC – SR04 has 4 pins. 2 pins are +5V and Gnd. There are Triggering and Echo signal pins.
Connect trigger signal pin to A4 analog pin and echo signal pin to A5 analog pin of Arduino

PROGRAMM :

```
// Ultrasonic sensor has 40KHz transmitter and receiver
// It has 2 pins Trigger and Echo
// Initially, in trigger pin, 10microsec pulse is given
// Immediately, 8 nos. of 40KHz sound pulses are transmitted
// In receiver, the bounce back sound pulses are received.
// pulseIn() function gives time delay in microsecs between reception and transmission
// The distance is calculated and displayed in LCD

// LiquidCrystal library is used
#include <LiquidCrystal.h> // includes the LiquidCrystal Library
LiquidCrystal lcd(12, 11, 4, 5, 6, 7); // Creates an LCD object. Parameters: (rs, enable, d4, d5, d6, d7)
const int trigPin = 2;
const int echoPin = 3;
long duration;
int distanceCm, distanceInch;
void setup() {
  lcd.begin(16, 2); // Initializes the interface to the LCD screen, and specifies the dimensions (width and height) of the display
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
}
```

```
void loop() {
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH);
  distanceCm = duration * 0.034 / 2;
  distanceInch = duration * 0.0133 / 2;
  lcd.setCursor(0, 0); // Sets the location at which subsequent text written to the LCD
will be displayed
  lcd.print("Distance: "); // Prints string "Distance" on the LCD
  lcd.print(distanceCm); // Prints the distance value from the sensor
  lcd.print(" cm");
  delay(10);
  lcd.setCursor(0, 1);
  lcd.print("Distance: ");
  lcd.print(distanceInch);
  lcd.print(" inch");
  delay(10);
}
```

Output

The **10K potentiometer** is used to **adjust the LCD contrast**.

- The **RW pin** is connected to **GND** to keep the LCD in **write mode**.

Result

EXP.NO : 5.i **IR Sensor Analog Input With Arduino**

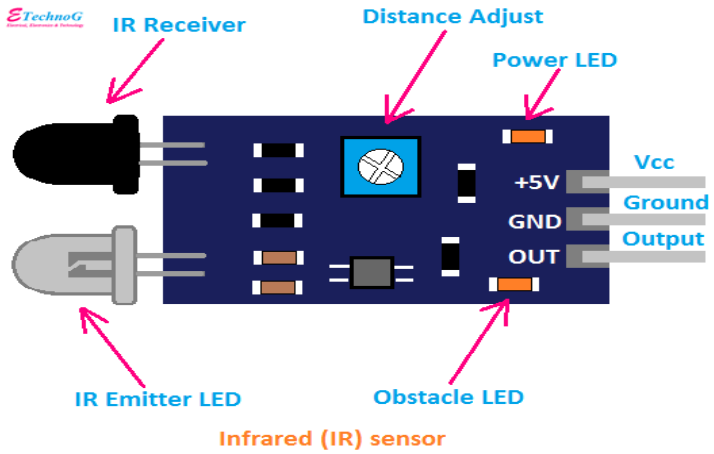
Aim :

Perform Interfacing of IR Sensor with Arduino and evaluate the response of variations.

THEORY:

An infrared sensor (IR sensor) is a radiation-sensitive optoelectronic component with a spectral sensitivity in the infrared wavelength range 780 nm … 50 µm. IR sensors are now widely used in motion detectors, which are used in building services to switch on lamps or in

alarm systems to detect unwelcome guests.



Program

```
const int ProxSensor=A0;
int inputVal = 0;

void setup()
{
  pinMode(13, OUTPUT);          // Pin 13 has an LED connected on most Arduino
boards:
  pinMode(ProxSensor,INPUT);    //Pin A0 is connected to the output of proximity
sensor
  Serial.begin(9600);
}

void loop()
{
  if(digitalRead(ProxSensor)==HIGH)      //Check the sensor output
  {
    digitalWrite(13, HIGH);   // set the LED on
  }
  else
  {
    digitalWrite(13, LOW);    // set the LED off
  }
inputVal = analogRead(ProxSensor);
Serial.println(inputVal);
delay(1000);              // wait for a second
}
```

Output

If your proximity sensor is **not detecting** an object:

230

245

250

240

(Values fluctuate around a low range.)
If an object is **near the sensor**:

CopyEdit

670

690

710

720

Result

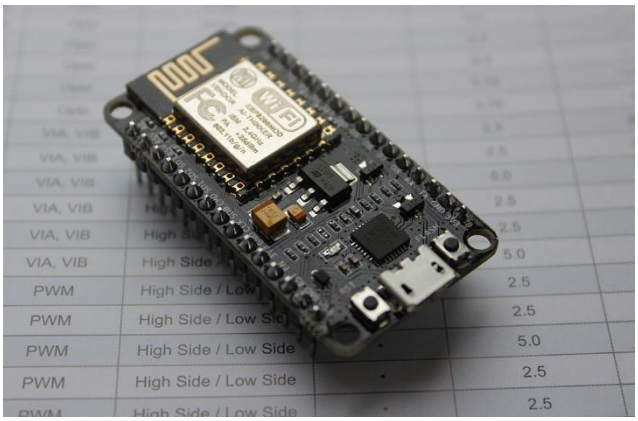
EX.NO: 6 Explore different communication methods with IOT Device

Node MCU is a low-cost open source IoT platform. And its used to communicate sensors data with the cloud platform. It initially included firmware which runs on the ESP8266 Wi-

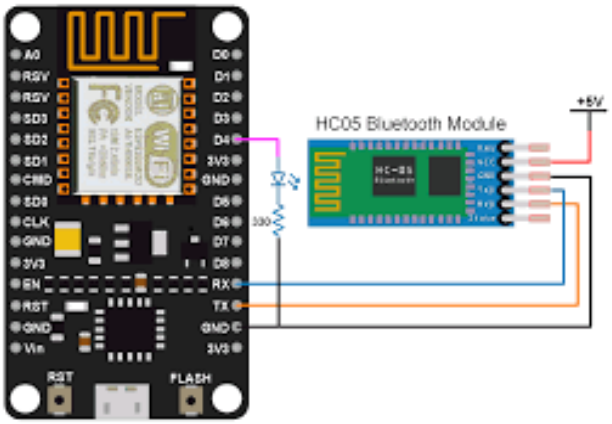
Fi SoC from Espressif Systems, and hardware which was based on the ESP-12 module. Later support for the ESP32 32-bit MCU was added.

Both the firmware and prototyping board designs are open source.

The firmware uses the Lua scripting language. The firmware is based on the eLua project, and built on the Espressif Non-OS SDK for ESP8266. It uses many open source projects, such as lua-cjson[9] and SPIFFS. Due to resource constraints, users need to select the modules relevant for their project and build a firmware tailored to their needs. Support for the 32-bit ESP32 has also been implemented.



/O index	ESP8266 pin
0 [*]	GPIO16
1	GPIO5
2	GPIO4
3	GPIO0
4	GPIO2
5	GPIO14
6	GPIO12
7	GPIO13
8	GPIO15
9	GPIO3
10	GPIO1
11	GPIO9
12	GPIO10



PROGRAM :

```
int LED = D1;

void setup() {
 pinMode(LED, OUTPUT);
 Serial.begin(9600); /* Define baud rate for serial communication */
}

void loop() {

 if (Serial.available())  /* If data is available on serial port */
    {
      char data_received;
      data_received = Serial.read();  /* Data received from bluetooth */
      if (data_received == '1')
      {
       digitalWrite(LED, HIGH);
       Serial.write("LED turned ON\n");
      }
      else if (data_received == '2')
      {
       digitalWrite(LED, LOW);
       Serial.write("LED turned OFF\n");
      }
      else
      {
       Serial.write("Select either 1 or 2");
      }
    }
}
```

Output

- Send '1' over Serial (or Bluetooth) → LED turns ON, responds:

graphql

LED turned ON

- Send '2' over Serial (or Bluetooth) → LED turns OFF, responds:

vbnet

LED turned OFF

- Send anything else → Responds with:

pgsql

Invalid input! Send '1' to turn ON, '2' to turn OFF.

Result

Ex .no : 7

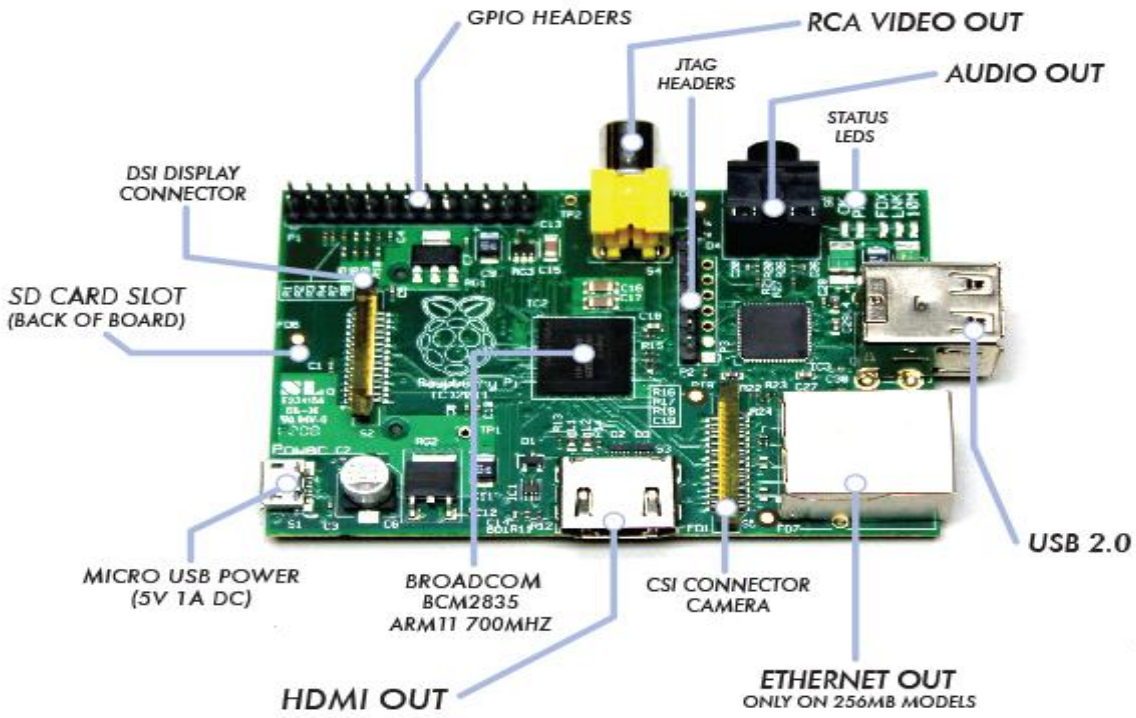
Introduction raspberry pi platform and python programming

Aim:- Introduction to RASPBERRY PI Platform and Python Programming.

Theory:-

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. It was created by Guido van Rossum during 1985- 1990. Like Perl, Python source code is also available under the GNU General Public License (GPL). This tutorial gives enough understanding on Python programming language.

Raspberry PI



Software platform:

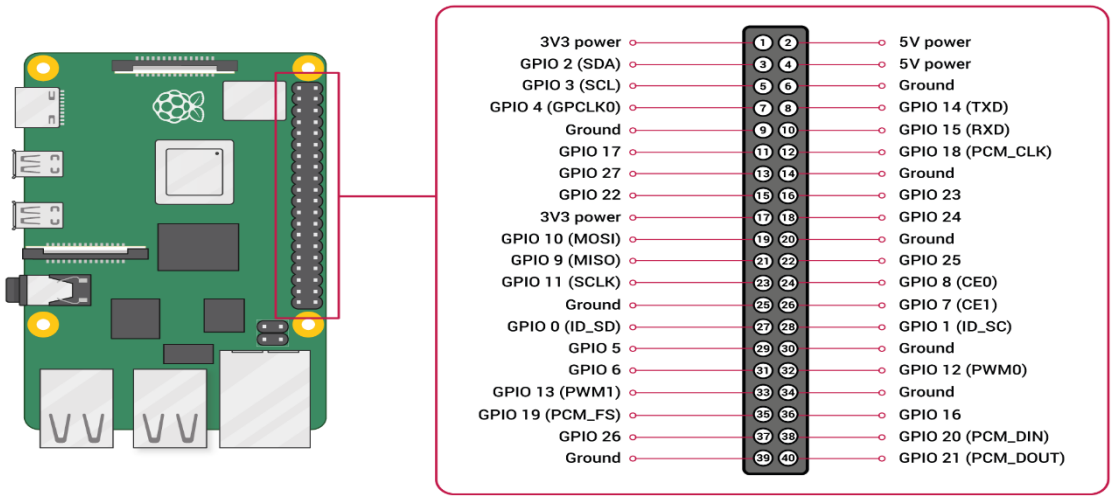
- Noobs
- Raspbian
- 3rd OS
- CentOS
- OpenMedia
- Vault
- Fedora
- LibreELEC
- DietPi
- RecalBox
- Gentoo
- Kano OS
- Kali Linux
- Lakka
- OSMC (Open Source Media Center)
- Manjaro
- Retropie
- Ubuntu

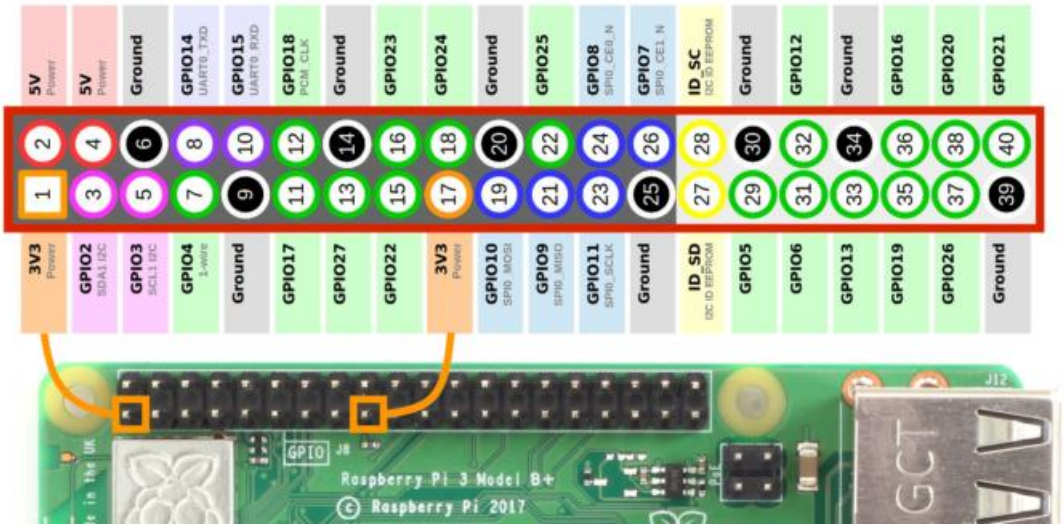
Kit Components:

- **Essential:**
  - Raspberry Pi board
  - Prepared Operating System SD Card
  - USB keyboard
  - Display (with HDMI, DVI, or Composite input)
  - Power Supply
- **Highly suggested extras include:**
  - USB mouse
  - Internet connectivity - LAN cable
  - Powered USB Hub

Programming Languages :

- **The Raspberry Pi Foundation recommends Python**
- **Any language which will compile for ARMv6 can be used**
- **Installed by default on the Raspberry Pi:**
  - C
  - C++
  - Java
  - Scratch
  - Ruby
  - Python





RASPBERRY PI TRAINER KIT

Sensors	Pin Details	Pi3 GPIO	Pi4 GPIO
Led and Switch	Led - pin7 Sw – Pin 11	Led – Gpio 7 Sw - Gpio 0	Led – Gpio 4 Sw - Gpio17
Relay	Pin 36	Gpio 27	Gpio16

Buzzer	Pin 38	Gpio 28	Gpio20
DHT	Pin 7	Gpio 7	Gpio 4
IR	Pin 36	Gpio 27	Gpio 16
LDR	Ldr-Pin 7 Sw – Pin 11	LDR-Gpio 7 Led -Gpio 0	LDR- Gpio 4 Led – Gpio 17
Servo Motor	Pin 29	Gpio 21	Gpio 5
Ultra sonic	Trig –pin 35 Echo –Pin 37	Trig –Gpio24 Echo –Gpio 25	Trig – Gpio 19 Echo –Gpio26
7 segment Display	Data- pin 7 Latch – pin 31 Clock –Pin 29	Data- Gpio 7 Latch –Gpio 22 Clock –Gpio 21	Data- Gpio 4 Latch –Gpio 6 Clock –Gpio 5
LCD	Rs - Pin 15 EN - Pin 12 D4 - Pin 23 D5 - Pin 21 D6 - Pin 19 D7 - Pin 24	Rs - Gpio 3 EN - Gpio 1 D4 - Gpio 14 D5 - Gpio 13 D6 - Gpio 12 D7 - Gpio 10	Rs - Gpio 22 EN - Gpio 18 D4 - Gpio 11 D5 - Gpio 9 D6 - Gpio 10 D7 - Gpio 8
Gas Sensor **A0 - MCP 3008 cho**	Clk - Pin 23 Dout – Pin 21 Din – Pin 19 Cs - Pin 24	Clk - Gpio 14 Dout – Gpio 13 Din – Gpio 12 Cs - Gpio 10	Clk - Gpio 11 Dout – Gpio 9 Din – Gpio 10 Cs - Gpio 8
Soil Moisture **A0 - MCP 3008 cho**	Clk - Pin 23 Dout – Pin 21 Din – Pin 19 Cs - Pin 24	Clk - Gpio 14 Dout – Gpio 13 Din – Gpio 12 Cs - Gpio 10	Clk - Gpio 11 Dout – Gpio 9 Din – Gpio 10 Cs - Gpio 8
Pulse Sensor **A0 - MCP 3008 cho**	Clk - Pin 23 Dout – Pin 21 Din – Pin 19 Cs - Pin 24	Clk - Gpio 14 Dout – Gpio 13 Din – Gpio 12 Cs - Gpio 10	Clk - Gpio 11 Dout – Gpio 9 Din – Gpio 10 Cs - Gpio 8

Python Program :

1. **LED BLINK PROGRAM :**

```
import RPi.GPIO as GPIO
import time
 GPIO.setwarnings(False)
 GPIO.setmode(GPIO.BOARD)
 GPIO.setup(11,GPIO.OUT, initial=GPIO.LOW)
while True:
 GPIO .output(11, GPIO.HIGH)
   time.sleep(1)
```

```
GPIO.output(11, GPIO.LOW)
  time.sleep(1)
```

Prerequisites

You should have a basic understanding of Computer Programming terminologies. A basic understanding of any of the programming languages is a plus.

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

• **Python is Interpreted** − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.

• **Python is Interactive** − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

• **Python is Object-Oriented** − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.

• **Python is a Beginner's Language** − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

History of Python:

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands. Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, and Unix shell and other scripting languages. Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL). Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

Python Features Python's features include –

• Easy-to-learn − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.

• Easy-to-read − Python code is more clearly defined and visible to the eyes.

• Easy-to-maintain − Python's source code is fairly easy-to-maintain.

• A broad standard library − Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.

• Interactive Mode − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.

• Portable − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.

• Extendable − You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.

• Databases − Python provides interfaces to all major commercial databases.

• GUI Programming − Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems,such as Windows MFC, Macintosh, and the X Window system of Unix.

• Scalable − Python provides a better structure and support for large programs than shell scripting. Apart from the above-mentioned features, Python has a big list of good features, few are listed below –

• It supports functional and structured programming methods as well as OOP.

• It can be used as a scripting language or can be compiled to byte-code for building large

applications.
• It provides very high-level dynamic data types and supports dynamic type checking.
• It supports automatic garbage collection.
• It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

First Python Program:

1. Open notepad and type following program Print ("Hello World")
2. Save above program with name.py
3. Open command prompt and change path to python program location
4. Type "python name.py" (without quotes) to run the program. Operators are the constructs which can manipulate the value of operands.

Consider the expression 4 + 5 = 9.
Here, 4 and 5 are called operands and + is called operator.

Python Variables: Declare, Concatenate, Global & Local

What is a Variable in Python?

A Python variable is a reserved memory location to store values. In other words, a variable in a python program gives data to the computer for processing.
Every value in Python has a datatype. Different data types in Python are Numbers, List, Tuple, Strings, Dictionary, etc.
Variables can be declared by any name or even alphabets like a, aa, abc, etc.

Types of Operator Python language supports the following types of operators.

• Arithmetic Operators
• Comparison (Relational) Operators
• Assignment Operators
• Logical Operators
• Bitwise Operators
• Membership Operators
• Identity Operators Let us have a look on all operators one by one.

How to Run It on Raspberry Pi

1. Connect LED to GPIO 11 and GND (with a 330Ω resistor).
2. Open a terminal and create a script:

nano led_blink.py

3. Paste the corrected code and save (Ctrl + X, then Y, then Enter).
4. Run the script:

python3 led_blink.py

5. Stop the script anytime with Ctrl + C.

Result

EX.NO :8

Interfacing Sensors with Raspberry Pi

Aim :

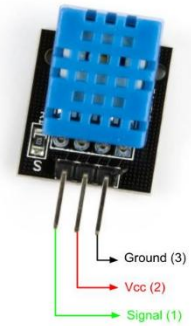
Perform Interfacing of DTH 11 Temperature and Humidity sensor with Raspberry Pi and evaluate the response of variations.

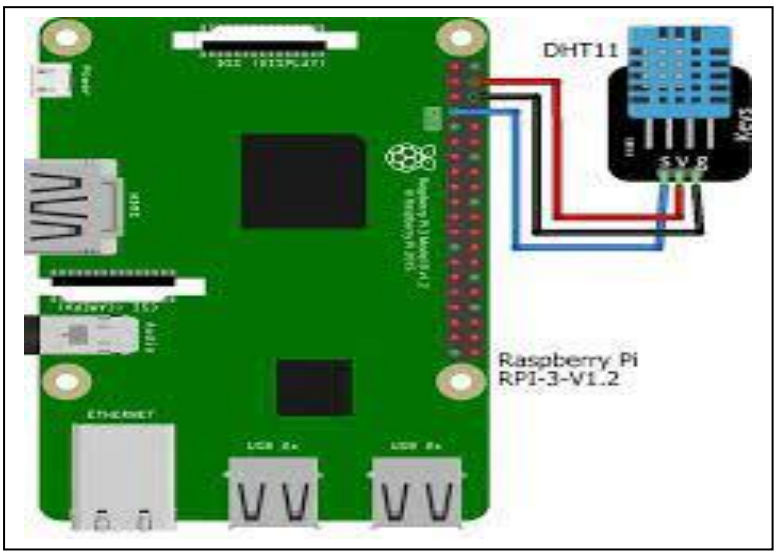
APPARATUS REQUIRED :

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

THEORY:

Humidity is the measure of water vapour present in the air. The level of humidity in air affects various physical, chemical and biological processes. In industrial applications, humidity can affect the business cost of the products, health and safety of the employees. So, in semiconductor industries and control system industries measurement of humidity is very important. Humidity measurement determines the amount of moisture present in the gas that can be a mixture of water vapour, nitrogen, argon or pure gas etc… Humidity sensors are of two types based on their measurement units. They are a relative humidity sensor and Absolute humidity sensor. DHT11 is a digital temperature and humidity sensor.





DHT11 humidity and temperature sensor is available as a sensor and as a module. The difference between this sensor and module is the pull-up resistor and a power-on LED. DHT11 is a relative humidity sensor. To measure the surrounding air this sensor uses a thermistor and a capacitive humidity sensor.

Working Principle of DHT11 Sensor

DHT11 sensor consists of a capacitive humidity sensing element and a thermistor for sensing temperature. 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.

For measuring temperature this sensor uses a Negative Temperature coefficient thermistor, which causes a decrease in its resistance value with increase in temperature. To get larger resistance value even for the smallest change in temperature, this sensor is usually made up of semiconductor ceramics or polymers.

The temperature range of DHT11 is from 0 to 50 degree Celsius with a 2-degree accuracy. Humidity range of this sensor is from 20 to 80% with 5% accuracy. The sampling rate of this sensor is 1Hz .i.e. it gives one reading for every second. DHT11 is small in size with operating voltage from 3 to 5 volts. The maximum current used while measuring is 2.5mA.

PROCEDURE :

1. Connect power cables , HDMI Converter cable, Mouse and Keyboard
2. Connect power adapter and switch ON
3. Open Rasbian OS and go to file
4. Create new file and write Python code
5. Save and Run
6. Output will be indicate on the screen

Program :

```
import time
import board
import adafruit_dht
dhtDevice = adafruit_dht.DHT11(board.D4, use_pulseio=False)  ## Raspberry pi 4 - gpio4
while True:
    try:
        # Print the values to the serial port
        temperature_c = dhtDevice.temperature
        temperature_f = temperature_c * (9 / 5) + 32
        humidity = dhtDevice.humidity
        print(
            "Temp: {:.1f} F / {:.1f} C    Humidity: {}% ".format(
                temperature_f, temperature_c, humidity
            )
        )

    except RuntimeError as error:
        # Errors happen fairly often, DHT's are hard to read, just keep going
        print(error.args[0])
        time.sleep(2.0)
        continue
    except Exception as error:
        dhtDevice.exit()
        raise erro
 time.sleep(2.0)
```

Result:

EX.NO: 9 Communicate between Arduino and Raspberry Pi using any wireless medium

Aim :

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

APPARATUS REQUIRED :

1. Arduino IOT Kit
2. USB Cable
3. Arduino SDK Software tool
4. VGA to mini HDMI Converer cable
5. Power Adapter
6. Short USB power cable
7. Patch cards

PROCEDURE :

1. **Arduino:**
   Write the arduino program and upload it
   As per connection bluetooth to arduino TX-RX , RX-TX

2. **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 :

EXP.NO : 10

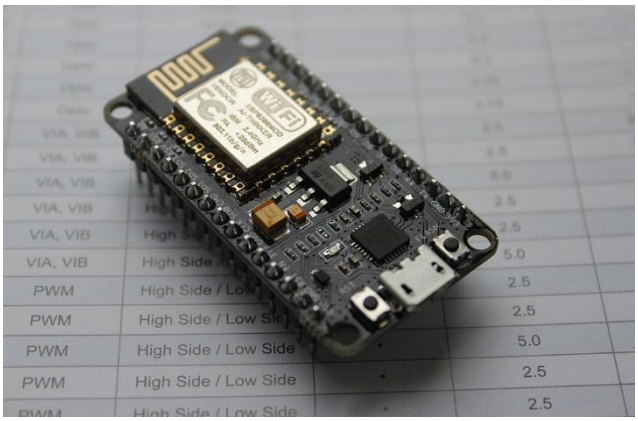
Setup a cloud Platform to log the data

Using NODE MCU module and Think Speak Cloud, writing and reading the temperature data

Node MCU is a low-cost open source IoT platform. It initially included firmware which runs on the ESP8266 Wi-Fi SoC from Espressif Systems, and hardware which was based on the ESP-12 module. Later support for the ESP32 32-bit MCU was added.

Both the firmware and prototyping board designs are open source.

The firmware uses the Lua scripting language. The firmware is based on the eLua project, and built on the Espressif Non-OS SDK for ESP8266. It uses many open source projects, such as lua-cjson[9] and SPIFFS. Due to resource constraints, users need to select the modules relevant for their project and build a firmware tailored to their needs. Support for the 32-bit ESP32 has also been implemented.



/O index	ESP8266 pin
0 [*]	GPIO16
1	GPIO5
2	GPIO4
3	GPIO0
4	GPIO2
5	GPIO14
6	GPIO12
7	GPIO13
8	GPIO15

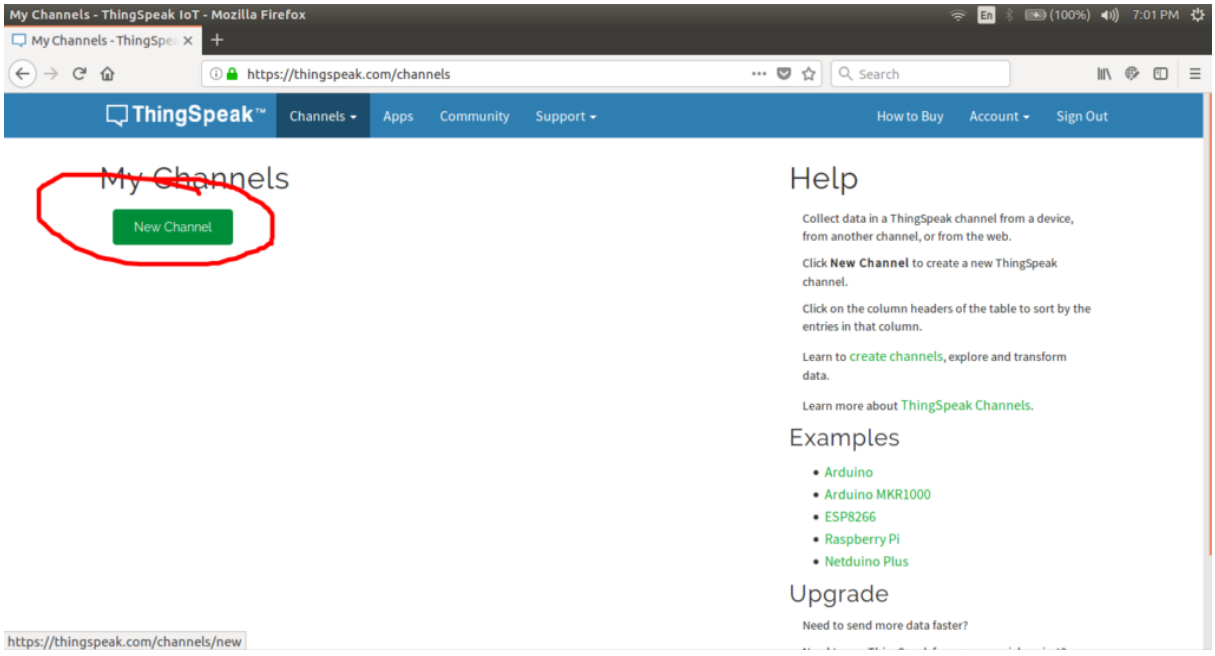
9	GPIO3
10	GPIO1
11	GPIO9
12	GPIO10

ThingSpeak is a cloud space owned by Matlab. Initally, we have to get an account and create a new channel with say one field and save the channel. Then next screen appears in which write and read api keys are available. These keys are noted and used later in the program.

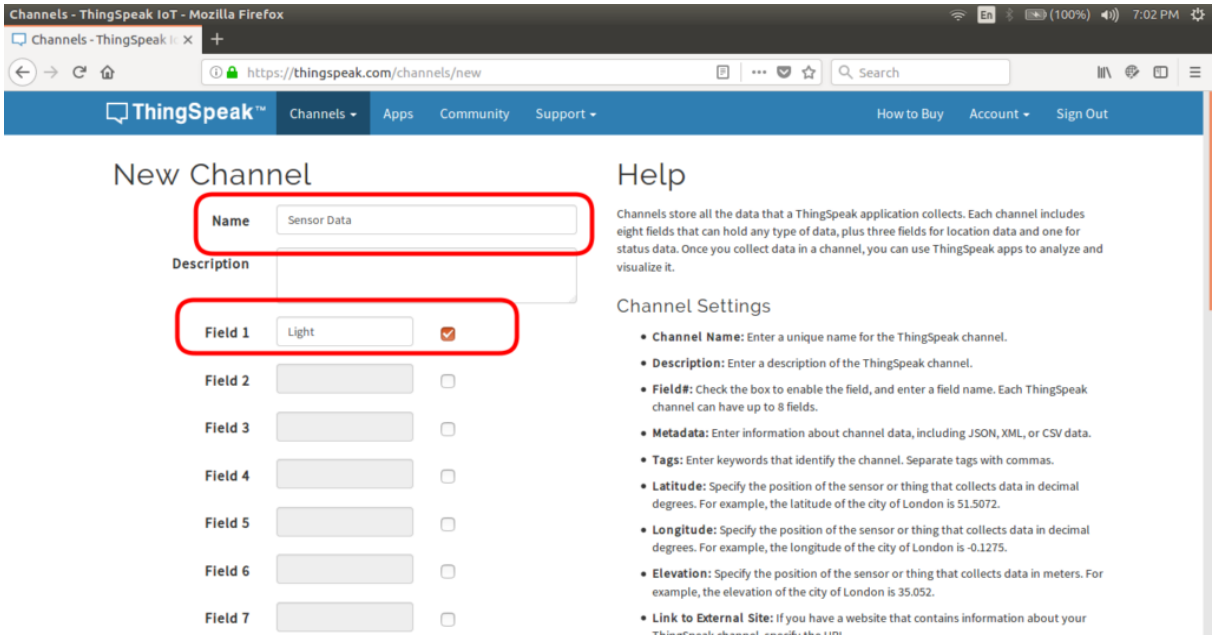
In this experiment, temperature is read and it is stored in the ThingSpeak cloud space.

Step 1: Sign up ThingSpeak
It is simple just enter your email id and verify your account.

Step 2: Configuring ThingSpeak
Configuration is just few clicks job
Open the ThingSpeak
Step 2.1: Create New Channel
Click on New Channel

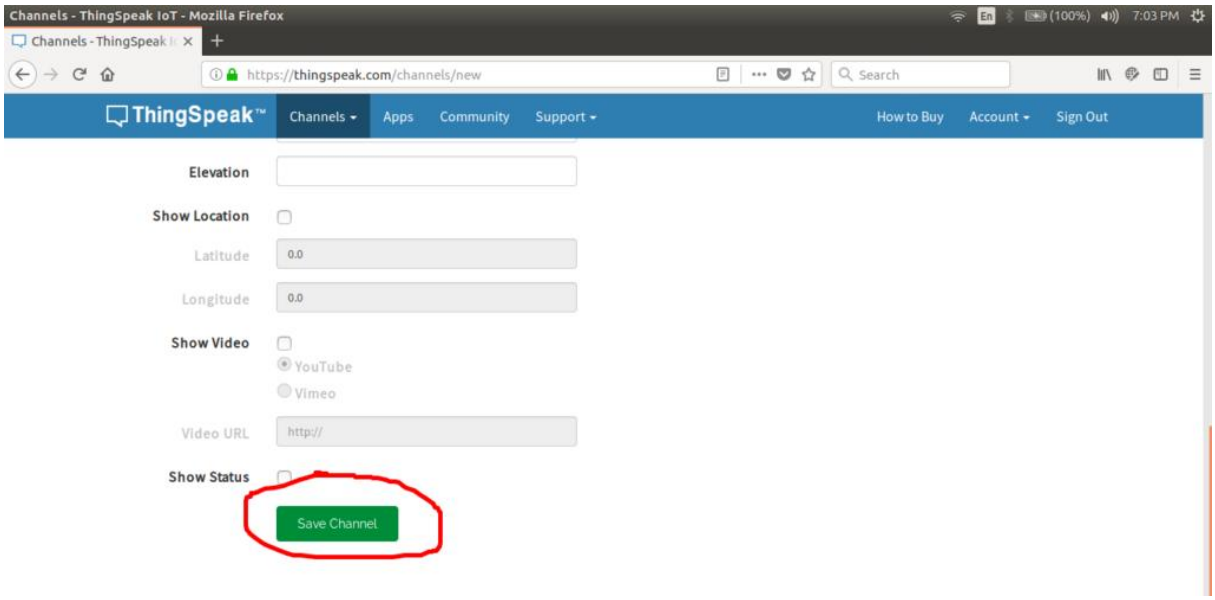


It creates a new channel in Thing speak for ESP8266

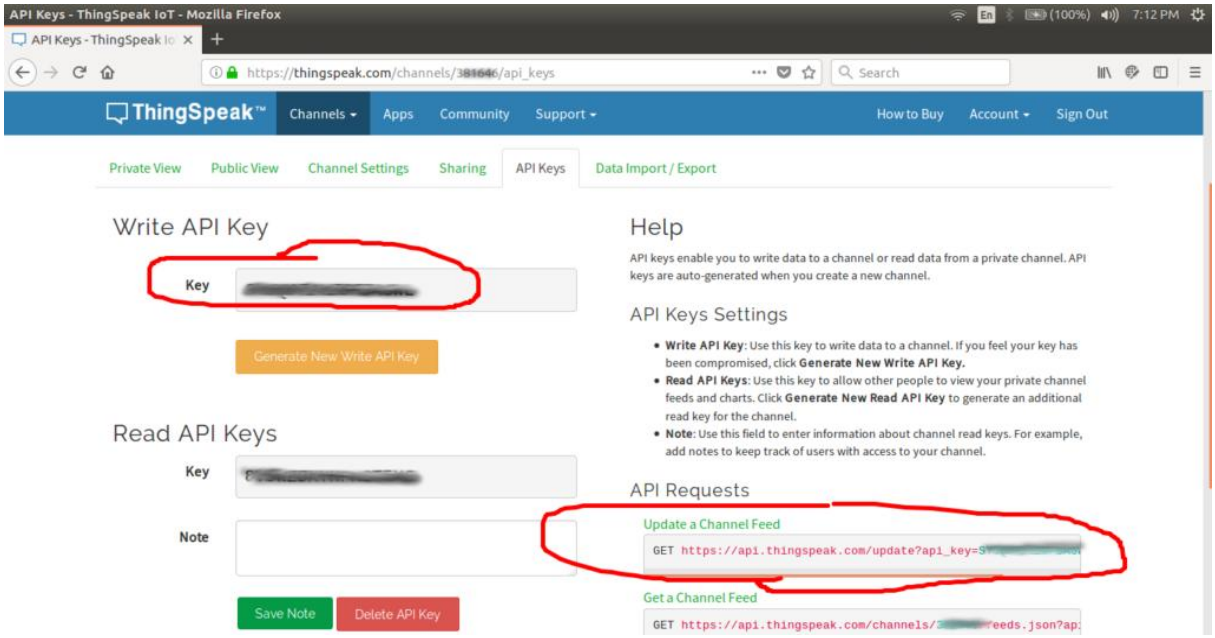


Enter Name and Field. You may have multiple Fields depending on number of sensor create multiple fields such as Light, Temperature, Humidity, etc.
Enter Name and Label thing speak esp8266

Keep everything else as it is. Blank or default values. and click on Save Channel. Thing Speak save channel



Step 2.2: Getting API Key
Click on API Keys Tab and look for these two fields Write Api Key and Update channel feed line.



Note Write api key and update a channel feed

This line is important for data upload to cloud server
GET https://api.thingspeak.com/update?api_key=akjshfajkhfowei&field1=0

AT commands: The following are the AT commands used in the program

AT – Attention
AT+ RST - Reset
AT+CWMODE – 1- station , 2 – Access point , 3 - both
AT+ CWJAP+ SSID+PW – join network
AT+CIPMUX – 0 single , 1 – multi
AT+CIPSTART+type+host+port – 0 single, 1 – multiple – (TCP/UDP)
AT+CIPSEND+datalength+bytes – 0 –single , 1 - multiple
AT+CIPCLOSE – 0 – single , 1 – multiple

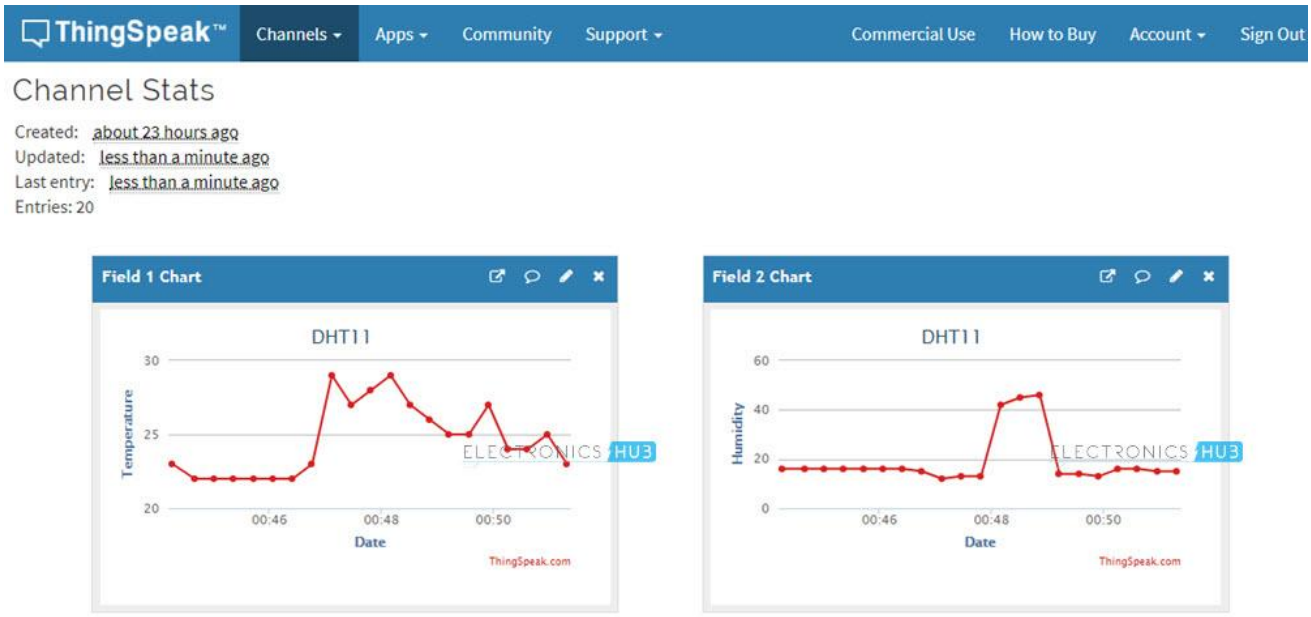
Program:

```
#include <DHT.h>  // Including library for dht

#include <ESP8266WiFi.h>

String apiKey = "               ";     //  Enter your Write API key from ThingSpeak

const char *ssid =  "         ";     // replace with your wifi ssid and wpa2 key
const char *pass =  "           ";
const char* server = "api.thingspeak.com";

#define DHTPIN 0          //pin where the dht11 is connected

DHT dht(DHTPIN, DHT11);

WiFiClient client;

void setup()
```

```
{
      Serial.begin(115200);
      delay(10);
      dht.begin();

      Serial.println("Connecting to ");
      Serial.println(ssid);


      WiFi.begin(ssid, pass);

     while (WiFi.status() != WL_CONNECTED)
     {
          delay(500);
          Serial.print(".");
     }
      Serial.println("");
      Serial.println("WiFi connected");

}

void loop()
{

     float h = dht.readHumidity();
     float t = dht.readTemperature();

           if (isnan(h) || isnan(t))
             {
                Serial.println("Failed to read from DHT sensor!");
                 return;
             }

                  if (client.connect(server,80))   //   "184.106.153.149" or api.thingspeak.com
                 {

                      String postStr = apiKey;
                      postStr +="&field1=";
                      postStr += String(t);
                      postStr +="&field2=";
                      postStr += String(h);
                      postStr += "\r\n\r\n";

                      client.print("POST /update HTTP/1.1\n");
                      client.print("Host: api.thingspeak.com\n");
                      client.print("Connection: close\n");
                      client.print("X-THINGSPEAKAPIKEY: "+apiKey+"\n");
                      client.print("Content-Type: application/x-www-form-urlencoded\n");
                      client.print("Content-Length: ");
                      client.print(postStr.length());
                      client.print("\n\n");
                      client.print(postStr);

                      Serial.print("Temperature: ");
                      Serial.print(t);
```

```
                    Serial.print(" degrees Celcius, Humidity: ");
                    Serial.print(h);
                    Serial.println("%. Send to Thingspeak.");
                }
        client.stop();

        Serial.println("Waiting...");

  // thingspeak needs minimum 15 sec delay between updates
  delay(1000);
}
```

OUT PUT :



EX.NO : 11 Log data using Raspberry Pi and upload to the cloud platform

IOT

INTRODUCTION TO IoT

Today the Internet has become ubiquitous, has touched almost every corner of the globe, and is affecting human life in unimaginable ways.

- We are now entering an era of even more pervasive connectivity where a very wide variety of appliances will be connected to the web.

- One year after the past edition of the Cluster book 2012 it can be clearly stated that the Internet of Things (IoT) has reached many different players and gained further recognition. Out of the potential Internet of Things application areas, Smart Cities (and regions), Smart Car and mobility, Smart Home and assisted living, Smart Industries, Public safety, Energy & environmental protection, Agriculture and Tourism as part of a future IoT Ecosystem (Figure 1.1) have acquired high attention.

We are entering an era of the "Internet of Things" (abbreviated as IoT).
There are 2 definitions:
First one is defined by Vermesan and second by Pe˜na-L´opez
1. The Internet of Things as simply an interaction between the physical and digital worlds. The digital world interacts with the physical world using a plethora of sensors and actuators.

2. Another is the Internet of Things is defined as a paradigm in which computing and networking capabilities are embedded in any kind of conceivable object.

- We use these capabilities to query the state of the object and to change its state if possible.
- In common parlance, the Internet of Things refers to a new kind of world where almost all the devices and appliances that we use are connected to a network.
- We can use them collaboratively to achieve complex tasks that require a high degree of intelligence.
- For this intelligence and interconnection, IoT devices areequipped with embedded sensors, actuators, processors, andtransceivers.
- IoT is not a single technology; rather it is anagglomeration of various technologies that work together intandem.
- Sensors and actuators are devices, which help in interactingwith the physical environment.
- The data collected by thesensors has to be stored and processed intelligently in order toderive useful inferences from it.
- Note that we broadly definethe term sensor; a mobile phone or even a microwave ovencan count as a sensor as long as it provides inputs about itscurrent state (internal state + environment).
- An actuator is adevice that is used to effect a change in the environment suchas the temperature controller of an air conditioner.
- The storage and processing of data can be done on theedge of the network itself or in a remote server.
- If any preprocessingof data is possible, then it is typically done at eitherthe sensor or some other proximate device.
- The processeddata is then typically sent to a remote server.
- The storageand processing capabilities of an IoT object are also restrictedby the resources available, which are often very constraineddue to limitations of size, energy, power, and computationalcapability.
- As a result the main research challenge is toensure that we get the right kind of data at the desired levelof accuracy.
- Along with the challenges of data collection, and handling, there are challenges in communication aswell.
- The communication between IoT devices is mainlywireless because they are generally installed at geographicallydispersed locations.
- The wireless channels often have high rates of distortion and are unreliable.
- In this scenario reliablycommunicating data without too many retransmissions is

animportant problem and thus communication technologiesare integral to the study of IoT devices.

- We can directly modify the physical world through actuators or we may do something virtually. For example,we can send some information to other smart things.

PROGRAM :

```
import httplib
import urllib
import time
key = "4IX9VESM09B9MQ7H"  # Put your API Key here
def thermometer():
   while True:
      #Calculate CPU temperature of Raspberry Pi in Degrees C
      temp = int(open('/sys/class/thermal/thermal_zone0/temp').read()) / 1e3 # Get Raspberry
Pi CPU temp
      params = urllib.urlencode({'field1': temp, 'key':key })
      headers = {"Content-typZZe": "application/x-www-form-urlencoded","Accept":
"text/plain"}
      conn = httplib.HTTPConnection("api.thingspeak.com:80")
      try:
         conn.request("POST", "/update", params, headers)
         response = conn.getresponse()
         print temp
         print response.status, response.reason
         data = response.read()
         conn.close()
      except:
         print "connection failed"
      break
if __name__ == "__main__":
      while True:
            thermometer()
```

Result

EX.NO : 12 DESIGN AN IOT BASED SYSTEM

Aim :

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

APPARATUS REQUIRED :

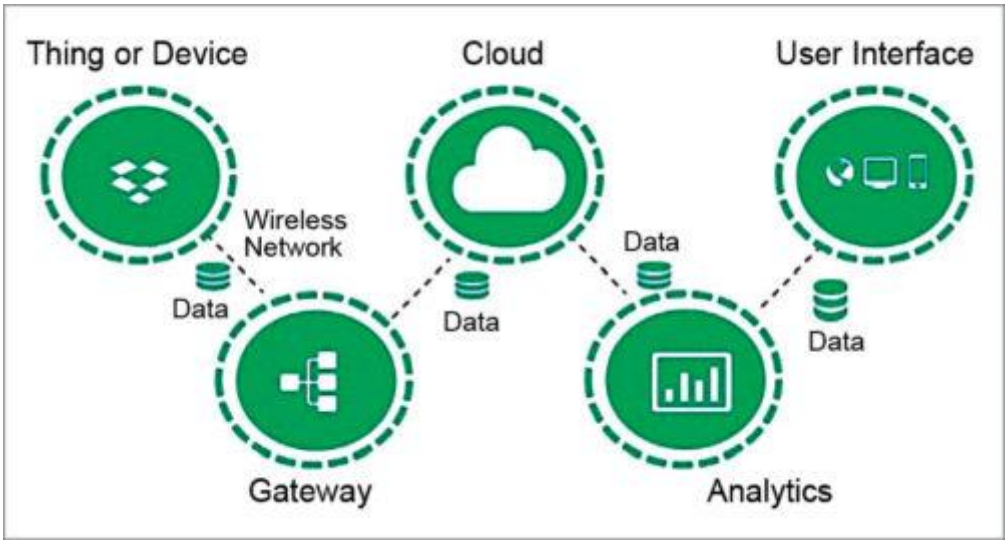
1.Embedded IOT Kit
2.USB Cable
3.Software tool
4.Patch cards
5.Power Adapter

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.

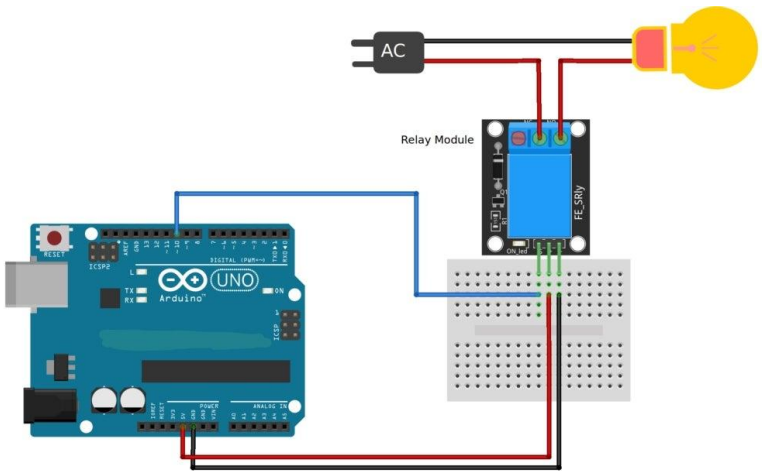
In this article, we'll explain the main features of ESP8266 modules and development boards and their application within the world of IoT.

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.

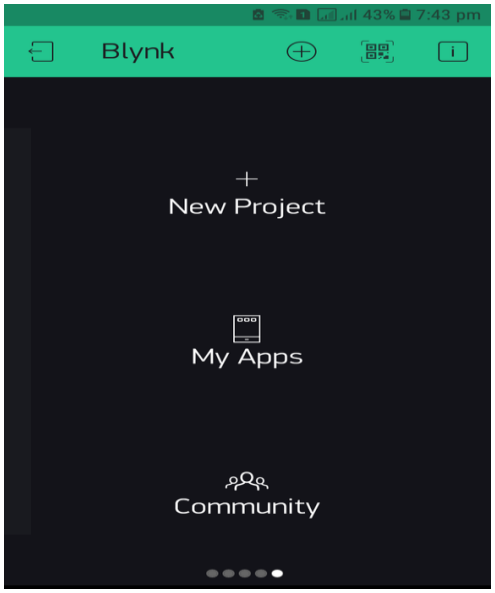


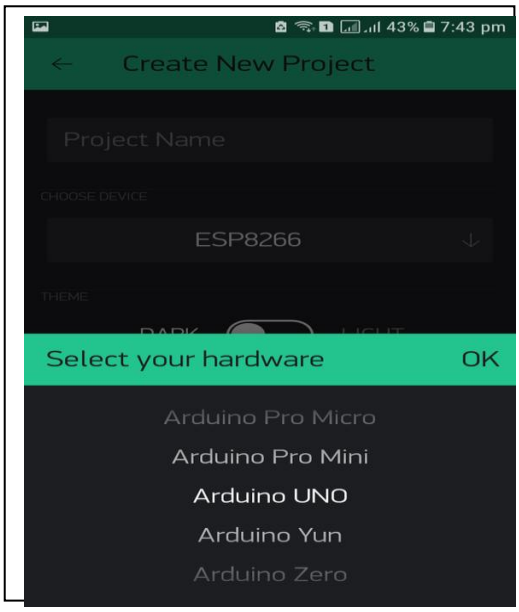
PROGRAMM :

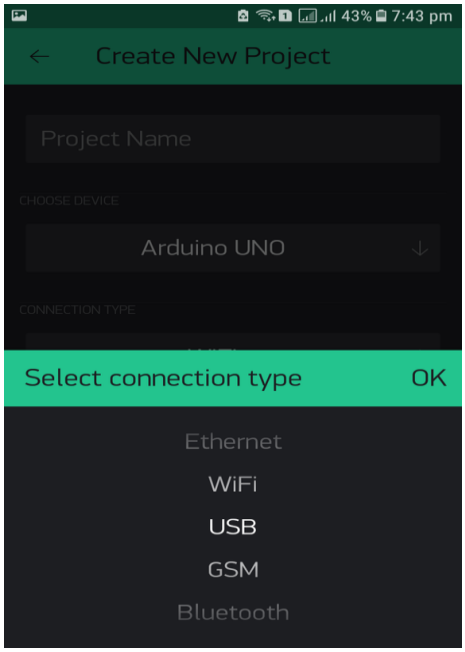
```
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>

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

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

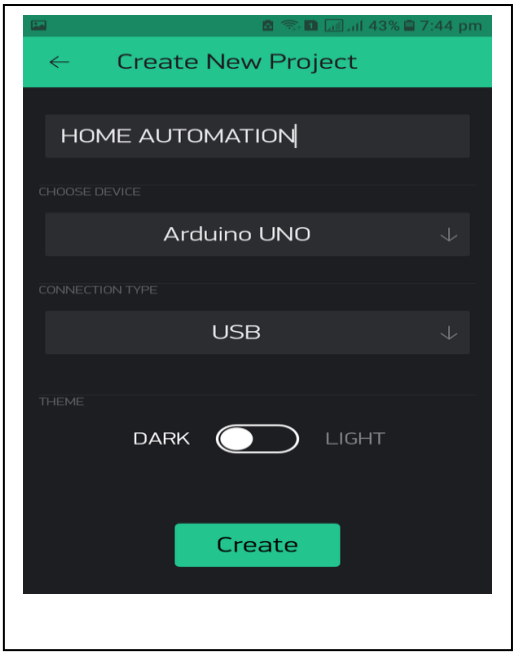
```
  // Connect to Wi-Fi
  Blynk.begin(auth, ssid, pass);

  // 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);
}
```

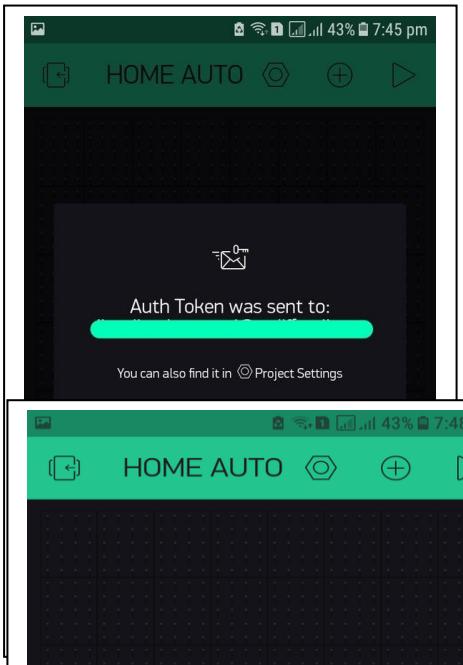
WORKING PRINCIPLE :

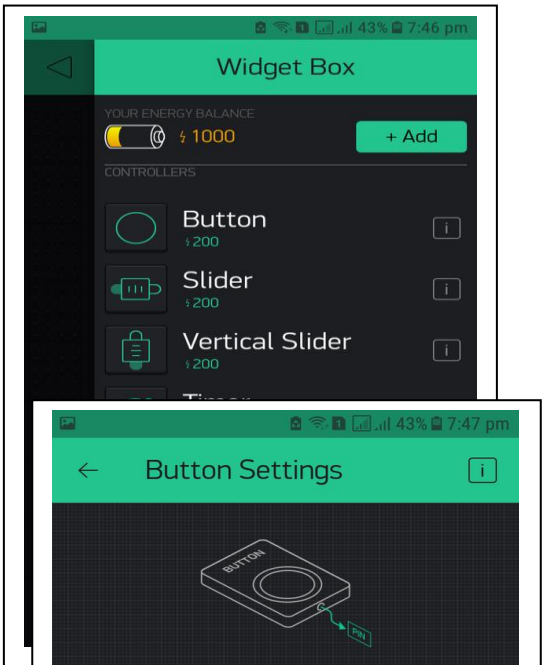


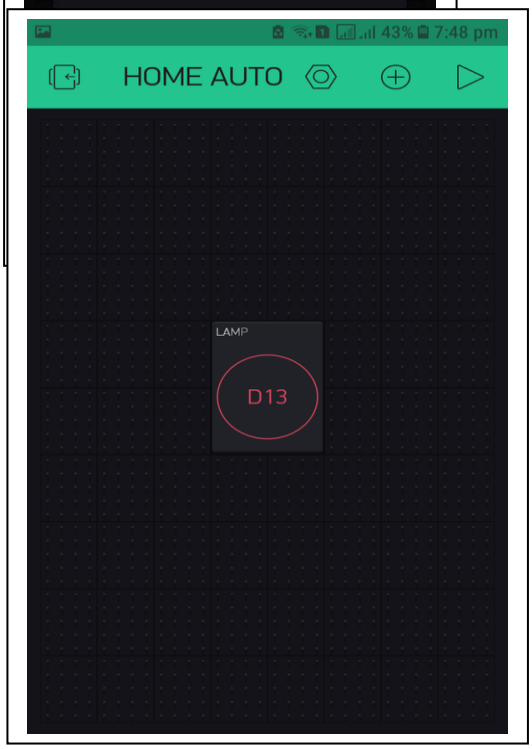


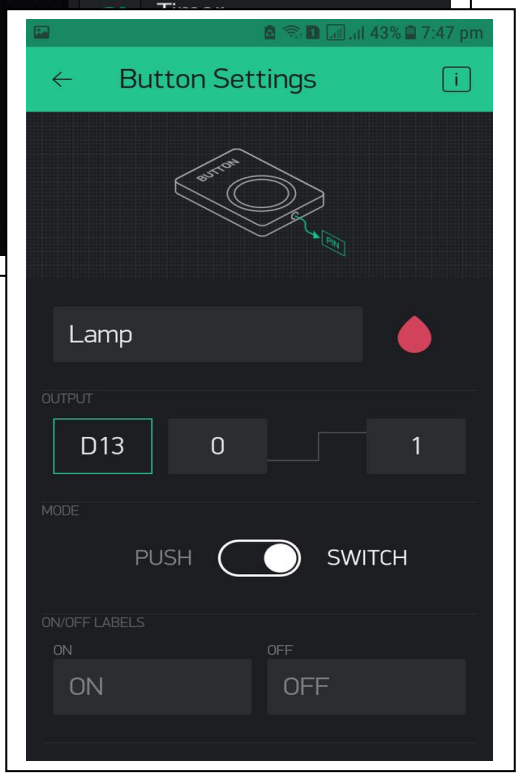

43% 7:43 pm
Create New Project
Project Name
CHOOSE DEVICE
Arduino UNO
CONNECTION TYPE
Select connection type
OK
Ethernet
WiFi
USB
GSM
Bluetooth


43% 7:44 pm
Create New Project
HOME AUTOMATION
CHOOSE DEVICE
Arduino UNO
CONNECTION TYPE
USB
THEME
DARK
LIGHT
Create

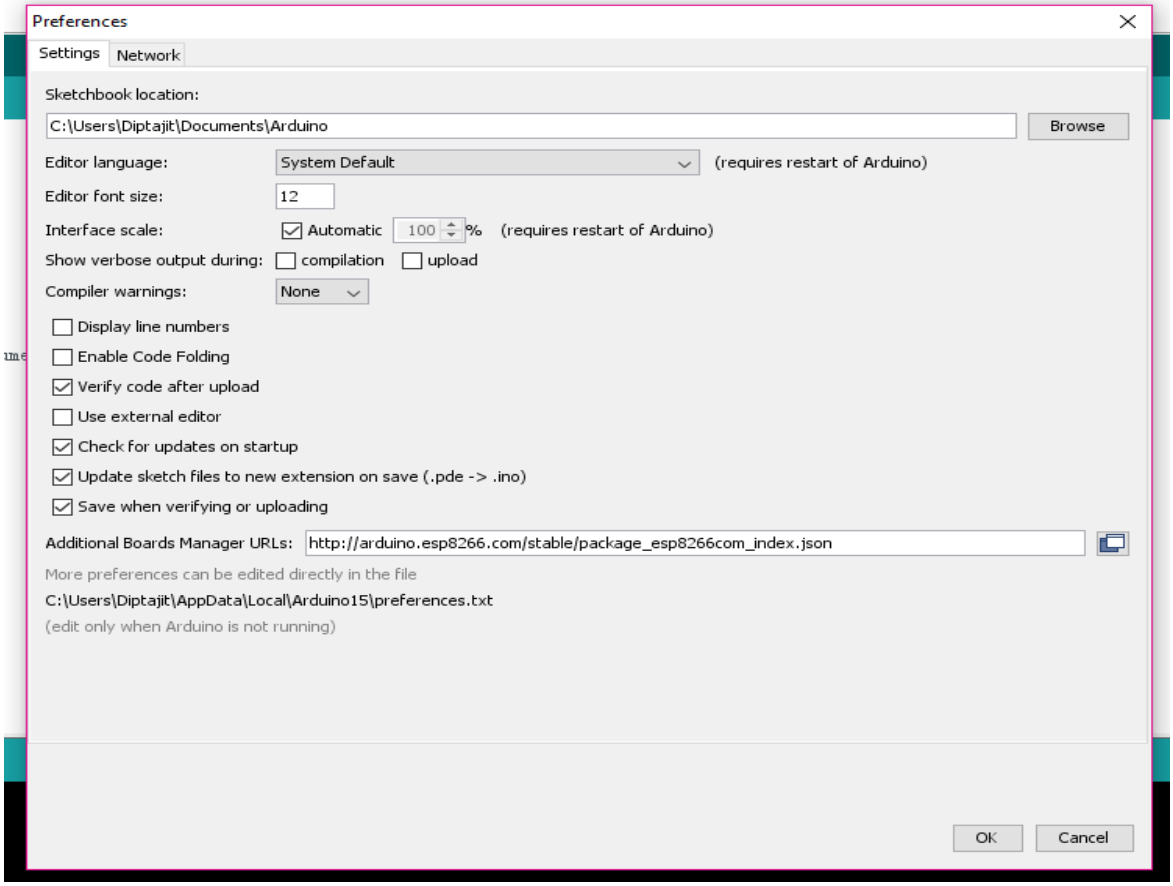

43% 7:45 pm
HOME AUTO
Auth Token was sent to:
You can also find it in Project Settings

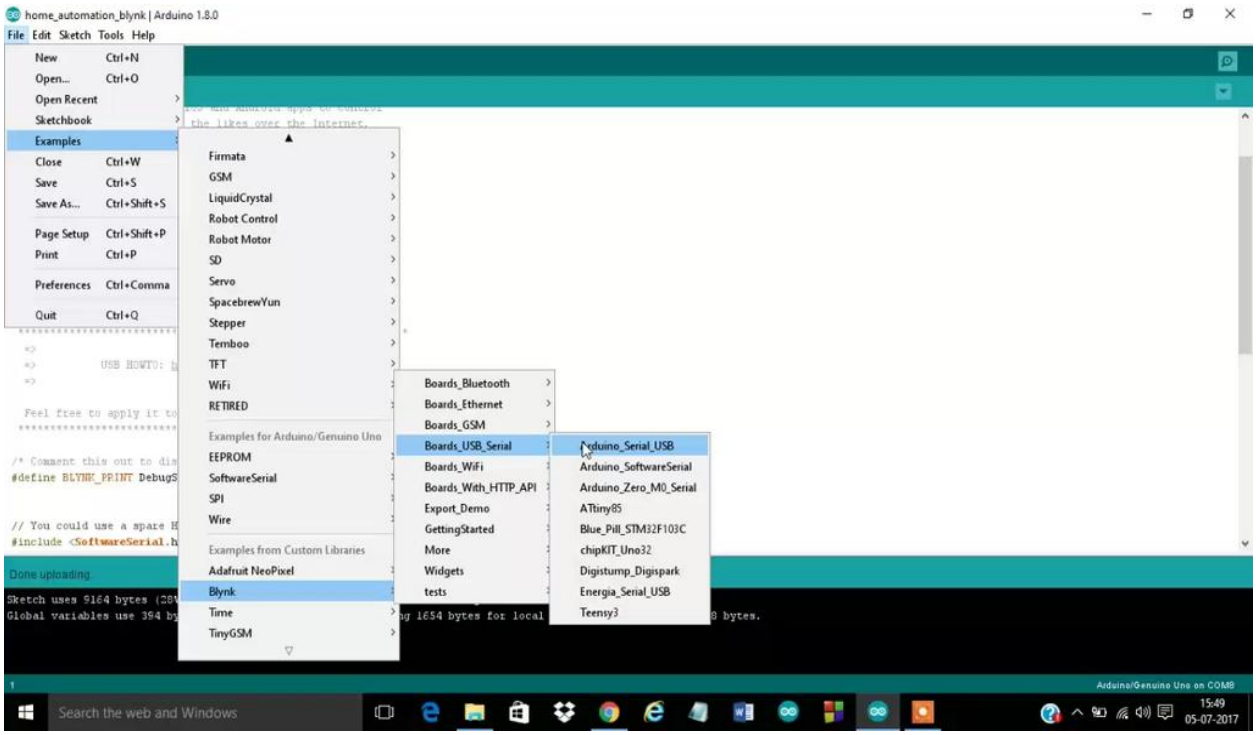

43% 7:46 pm
Widget Box
YOUR ENERGY BALANCE
1000
+ Add
CONTROLLERS
Button
200
Slider
200
Vertical Slider
200

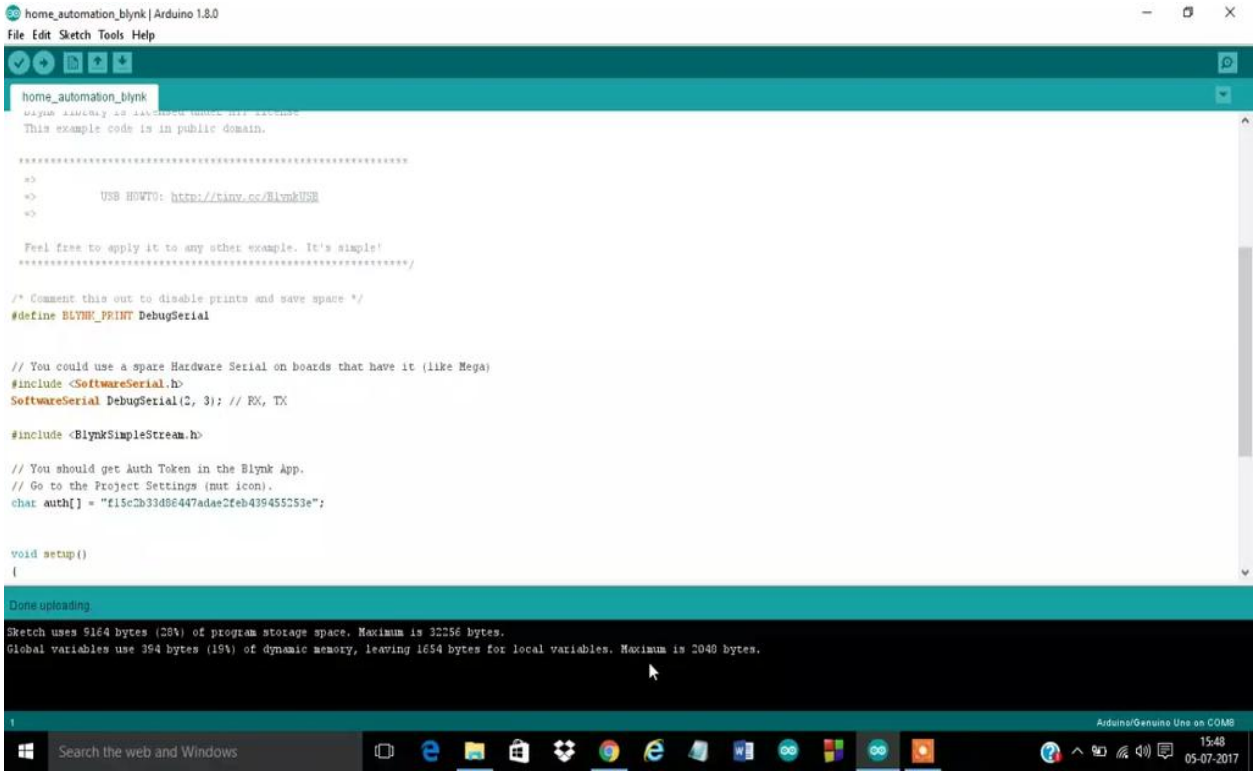

43% 7:48 pm
HOME AUTO
LAMP
D13

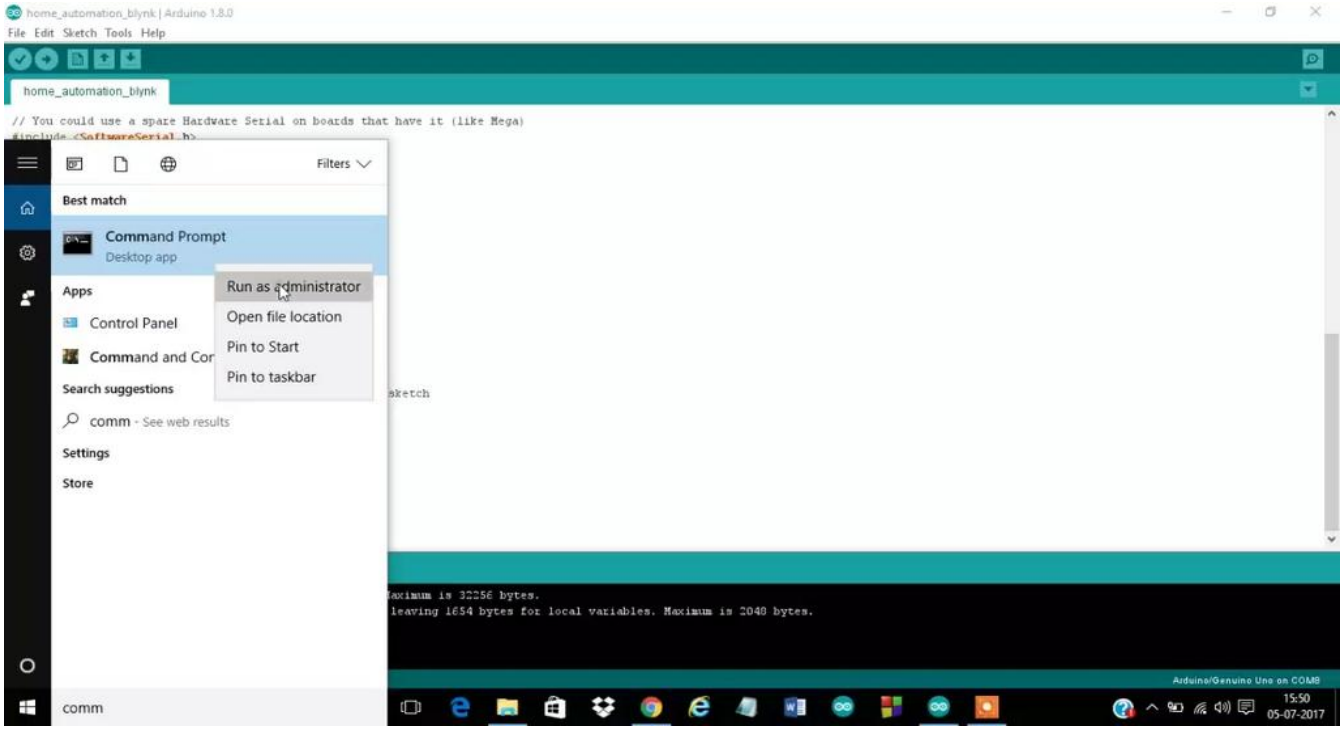

43% 7:47 pm
Button Settings
BUTTON
PIN
Lamp
OUTPUT
D13
0
1
MODE
PUSH
SWITCH
ON/OFF LABELS
ON
OFF
ON
OFF


43% 7:48 pm

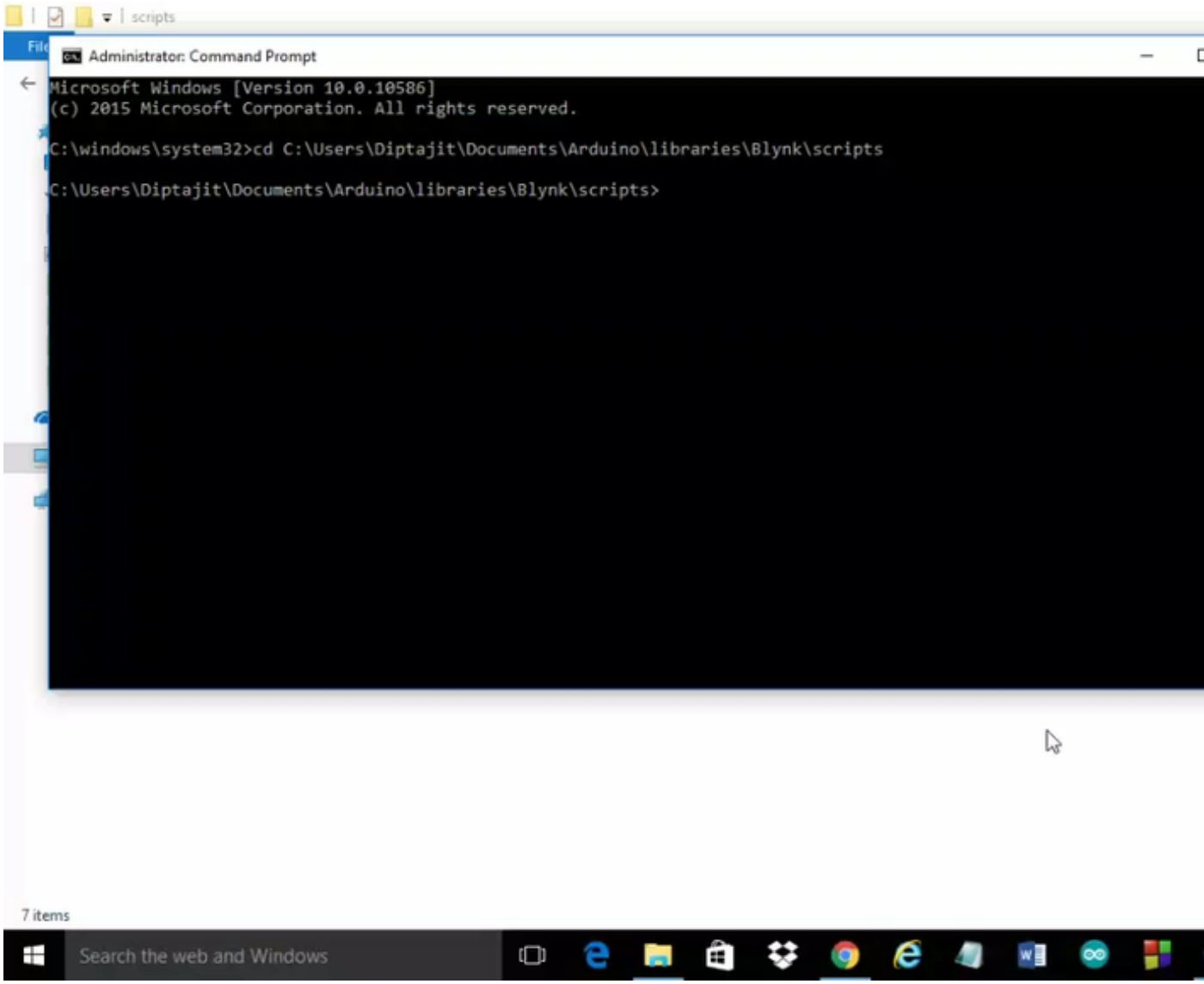

Preferences
Settings
Network
Sketchbook location:
C:\Users\Diptajit\Documents\Arduino
Browse
Editor language:
System Default
(requires restart of Arduino)
Editor font size:
12
Interface scale:
Automatic
100
%
(requires restart of Arduino)
Show verbose output during:
compilation
upload
Compiler warnings:
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/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 9164 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
leaving 1654 bytes for local variables. Maximum is 2048 bytes.
Arduino/Genuino Uno on COM8
15:50
05-07-2017


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

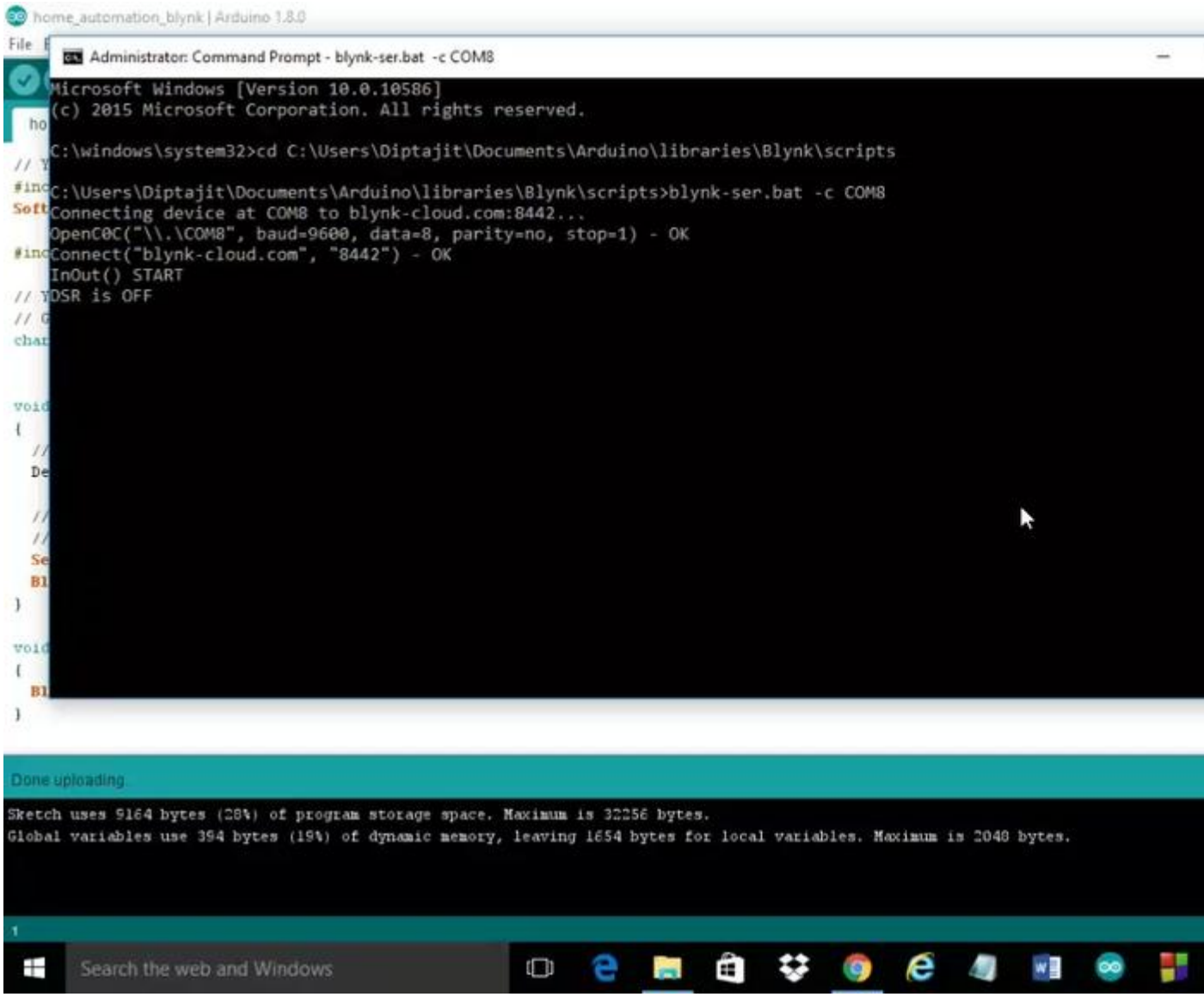


Write your path to blynk-ser.bat folder. For example:

```
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 :