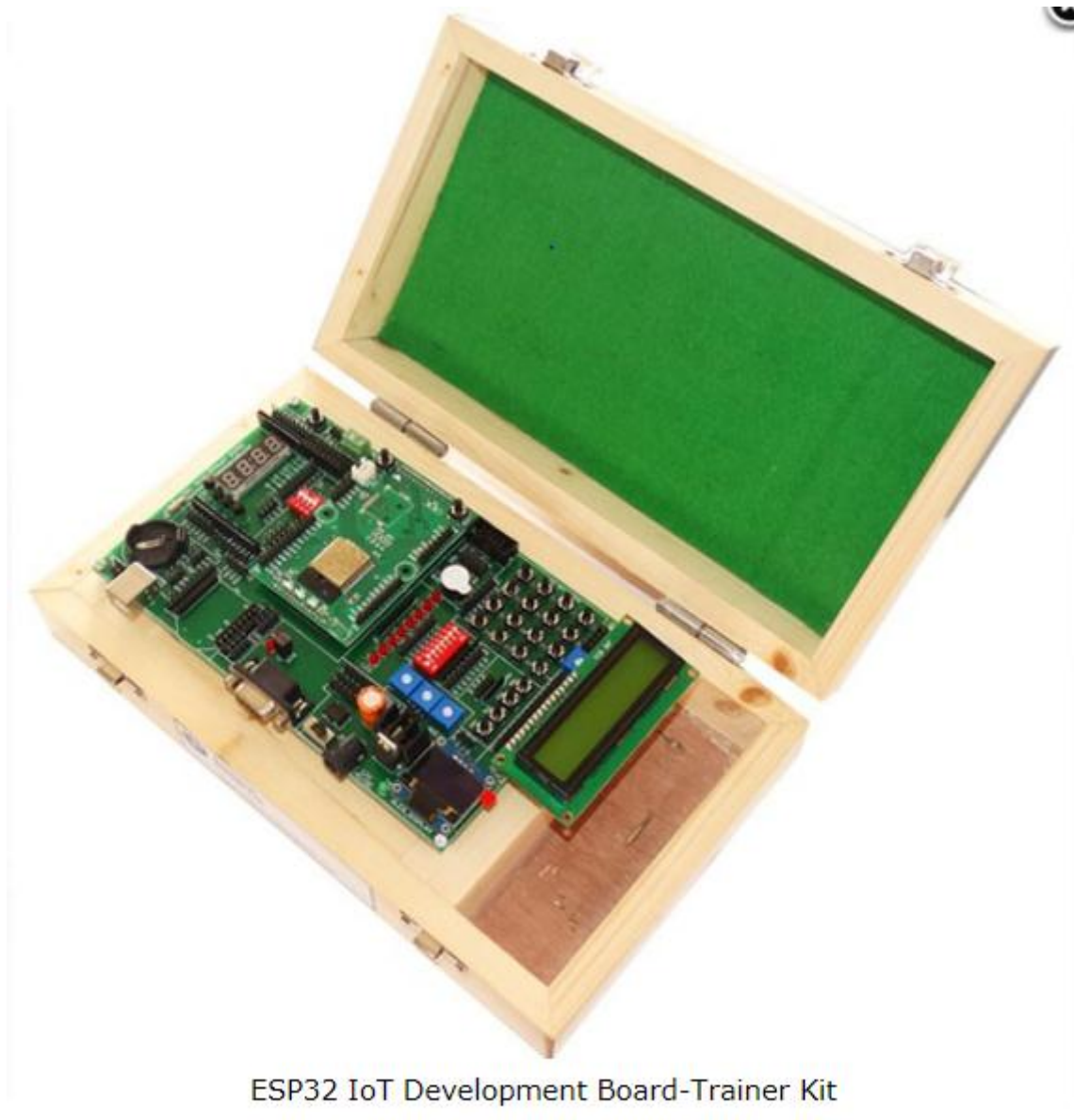


ESP32 IoT DEVELOPMENT TRAINER KIT



Features:

- High quality ESP32 development board.
- Standard I/O interface.
- Programmable with Arduino open source IDE.
- 8 independent LED.
- 1 * 4 independent keys.
- 4* 4 keypad matrix.
- RTC DS1307 with battery connector.
- I2C EEPROM Interfaces.
- The MAX232 chip RS232 communication.
- 16X2 LCD interface (character display).
- All IO Ports pin have extraction and clearly marked.
- On Board Power supply 3.3V, 5V 12V,GND.
- 8 pin DIP switches.

- The board also has inbuilt Xbee footprint.
- 3 ADC potentiometers.
- ON/OFF Slide switch.
- I2C Bus.
- SPI Bus.
- OLED interface.
- SD Card holder.
- RS232 Connector.
- Reset button.
- Power plug in DC socket.
- Power supply indicator LED.
- Test led for Tx, Rx.
- 4 digit 7 seg Multiplexed display
- Pin outs for 3.3V, 12V, 5V, GND.

ESP32 Transceiver Features:

- Ultra low power consumption
- Low cost SoC
- Built in antenna switches, power modules and filters
- Operating temperature – 40°C to +125°C
- Power supply 2.2V to 3.6V
- Supports UART/I²C/I²S/SPI protocol
- 448 KByte ROM, 520 KByte SRAM
- Wi-Fi - 802.11 b/g/n
- Bluetooth - v4.2 BR/EDR and BLE
- Xtensa® Dual-Core 32-bit LX6 microprocessors, up to 600 DMIPS

ESP32 Applications:

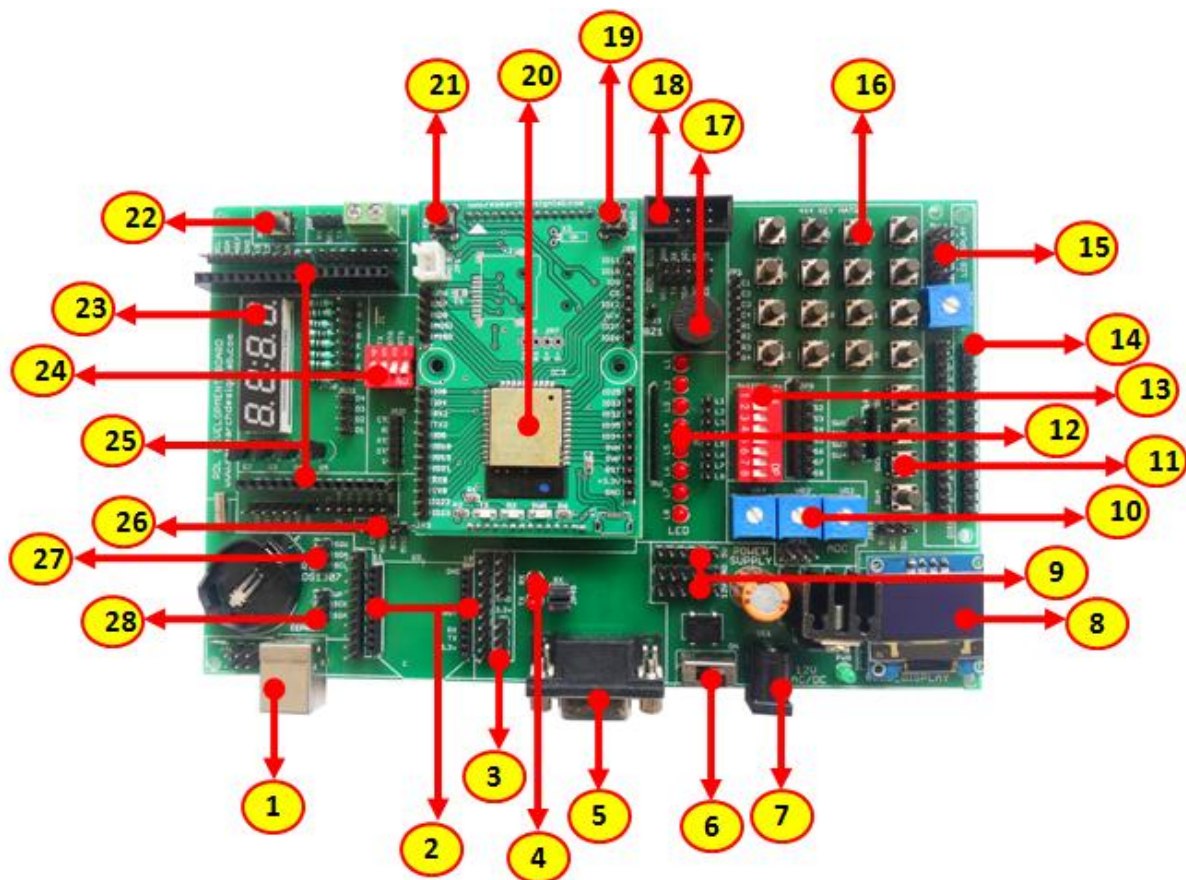
- Creating IOT Hub
- Low power IoT applications and data loggers
- Used in music players and audio streaming electronics
- Wi-Fi enabled proximity sensing
- Wi-Fi enabled home automation, smart agriculture, on-off control
- Real time wireless industrial plant condition monitoring
- Finds application in health and retail sector

Scope of Learning Experiments:

- Creating IOT Hub
- MQTT & Postman
- Monitoring Nodes
- Device to Cloud Connection (D2C)
- Device to Device Connection (D2D)
- Cloud to Device Connection (C2D)
- Creating GUI for Nodes
- Cloud Analytics

- Machine Learning
- LED blinking, shift operation
- 1X4 Keypad interfacing.
- 16X2 LCD interfacing.
- 4X4 Keypad interfacing.
- ADC interfacing.
- 7 Segment interfacing.
- DC Motor and Stepper motor interfacing.
- RTC DS1307 I2C protocol interfacing.
- AT24C04 EEPROM I2C protocol interfacing.
- Bluetooth interfacing.

ESP-32 Board Narration



1. USB Programming port
2. XBEE footprint/ XBEE Adaptor module
3. DC 3.3V connectors
4. TX and RX LED's
5. DB-9 serial female connector

6. Power ON switch
7. Power supply
8. OLED Display
9. DC 12V, 5V, GND connectors
10. Variable resistor POT
11. 4x1 Keypad
12. 8x1 LED's
13. 8 way DIP switch
14. 16x2 LCD connectors
15. Node connector
16. 4x4 Keypad matrix
17. Buzzer
18. RDL Bus FRC Connector
19. Boot Button
20. ESP32 Wi-Fi
21. Reset button
22. Reset button
23. 4x1 7 Segment display
24. 4 way DIP switch (always ON)
25. Stackable header for Arduino Shields
26. SD card pinouts **
27. RTC pinouts
28. EPROM pinouts

** SD card holder is placed at the bottom of the PCB.

Contents

1. BLINKING AN LED	7
2. CONTROLLING LED USING SWITCH.....	9
3. SEVEN SEGMENT DISPLAYS.....	11
4. HEXKEYPAD	15
5. LIQUID CRYSTAL DISPLAY	17
6. CONTROLLING LED BRIGHTNESS	19
7. RTC (Real Time Clock)	21
8. SD CARD	24
9. OLED.....	28
10.VIBRATION SENSOR.....	30
11.TEMPERATURE SENSOR.....	32
12.IR(Infrared) SENSOR	35
13.MQTT	38
14.JSON	47
15.FTP.....	52
16.OTA (Over the air) programming.....	61

EXPERIMENT NO 1

BLINKING AN LED

Aim:

Turn ON and OFF an LED after Particular delay

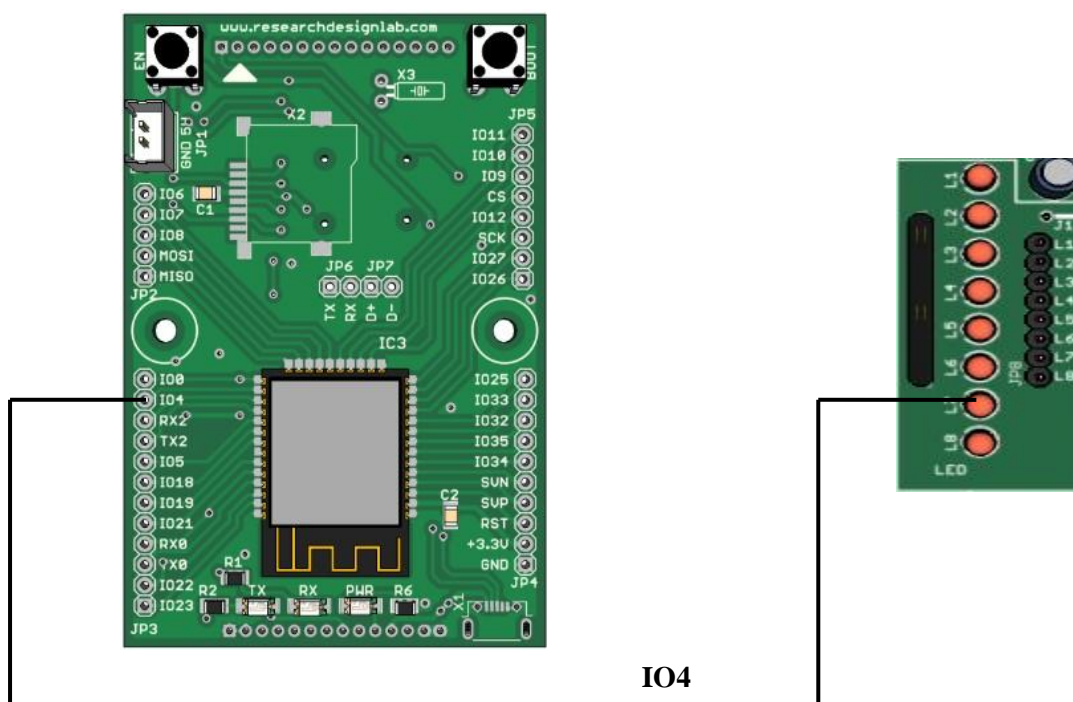
Description:

To learn how to connect LED to digital pins of an ESP32 Microcontroller and program to blink a LED.

Hardware Requirement:

ESP32 Microcontroller Development board.

Pin Connections:



Pin Mapping:

LED	ESP32
LED	IO4

Procedure:

1. The above pin connection shows the connection between LED and ESP32 development board.
2. Connect IO4 pin of ESP32 development board to any of the LED pin.
3. Connect the USB cable to the board.
4. Open Arduino IDE. Select DOIT ESP32 DEVKIT V1 in boards and select COM port.
5. Now verify the program and Upload it.
6. Now you can see the LED blink on the ESP32 development board.

Program:

```
void setup(void)
{
  /* Initialize the output pin */
  pinMode(4,OUTPUT);
}

/* the loop function runs over and over again forever */
void loop(void)
{
  /* Turn the LED on */
  digitalWrite(4,HIGH);

  /* Wait for a second */
  delay(1000);

  /* Turn OFF LED */
  digitalWrite(4,LOW);

  /* Wait for a second */
  delay(1000);
}
```


EXPERIMENT NO 2

CONTROLLING LED USING SWITCH

Aim:

Controlling LED using a Switch.

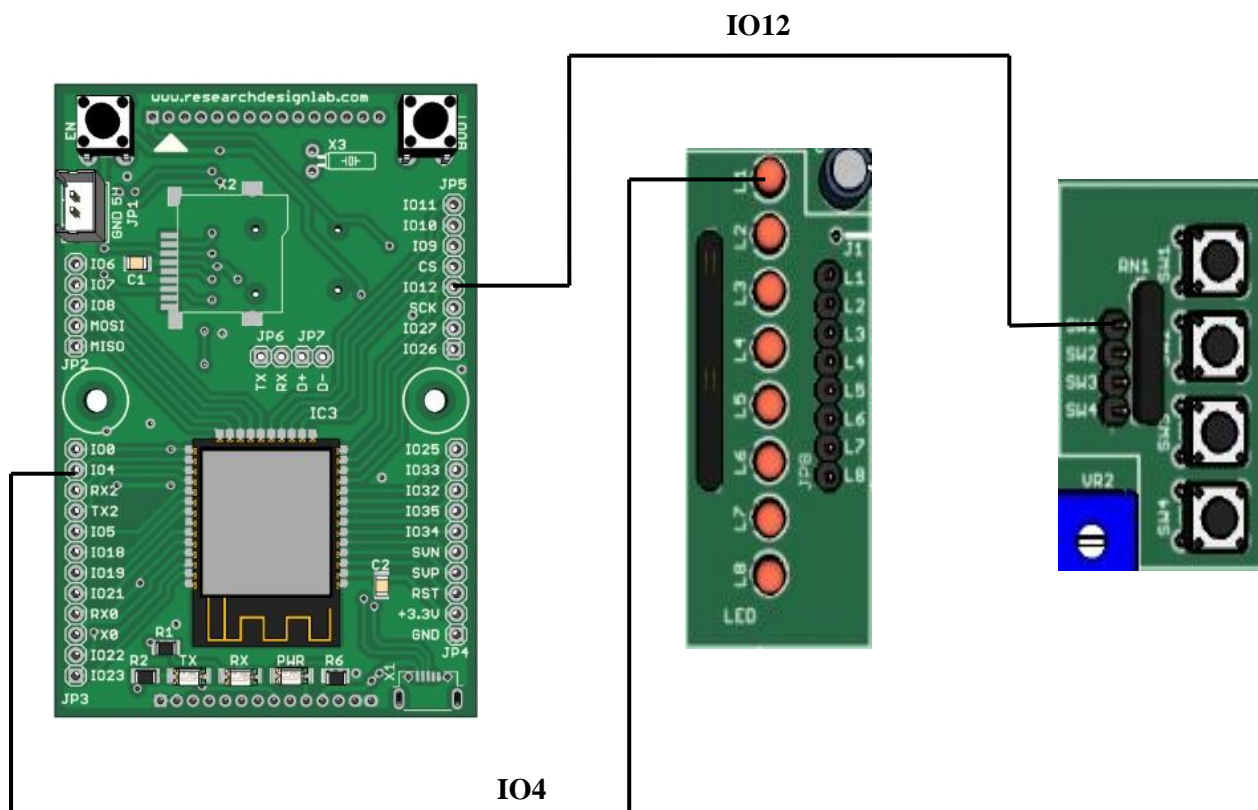
Description:

Understanding the working of switch. Turns ON the LED when switch is pressed and Turns OFF when it is released.

Hardware Required:

ESP32-Microcontroller Development board.

Pin Connections:



Pin Mapping:

Development Board	ESP32
Switch	IO12
LED	IO4

Procedure:

1. The above pin connection shows how to control an LED using switch.
2. Connect IO4 pin of ESP32 development board to any of the LED pin and IO12 pin to switch.
3. Connect the USB cable to the board.
4. Open Arduino IDE .Select DOIT ESP32 DEVKIT V1 in boards and select COM port.
5. Now verify the program and Upload it.
6. Now you can see that when switch is pressed LED starts to glow on the ESP32 development board.

Program:

```
void setup(void)
{
  /* Initialize the output pin */
  pinMode(4,OUTPUT);

  /* Switch is connected to pin 12 */
  pinMode(12,INPUT);

  /* to set baud rate */
  Serial.begin(9600);
}

/* the loop function runs over and over again forever */
void loop(void)

  if(digitalRead(12)==HIGH)
  {
    /* Turn the LED on */
    digitalWrite(4,HIGH);

    /* Wait for a second */
    delay(1000);

    /* Turn OFF LED */
    digitalWrite(4,LOW);

    /* Wait for a second */
    delay(1000);
  }
}
```

EXPERIMENT NO 3 SEVEN SEGMENT DISPLAYS

Aim:

Interfacing ESP32-Microcontroller with seven segment display and to display the numbers.

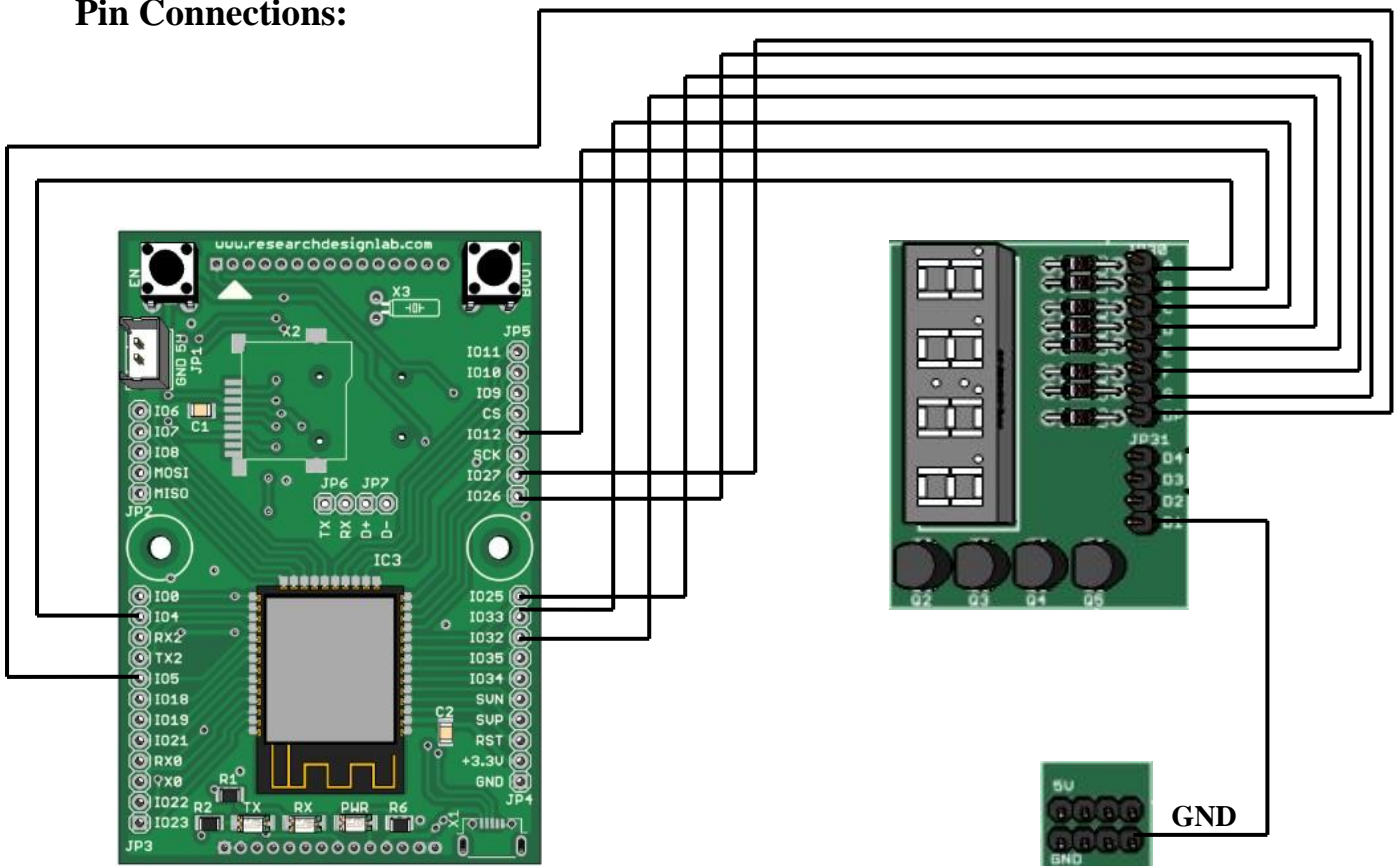
Description:

To display numbers in the seven segment in increasing order.

Hardware Required:

ESP32-Microcontroller Development board.

Pin Connections:



Pin Mapping:

Development Board	ESP32
A	I04
B	I012
C	I033
D	I032
E	I025
F	I026
DP	I05
D1	GND

Procedure:

1. The above pin connection shows how to display numbers in seven segments.
2. Do the connections as shown in the pin diagram and pin mapping.
3. Connect the USB cable to the board.
4. Open Arduino IDE .Select DOIT ESP32 DEVKIT V1in boards and select COM port.
5. Now verify the program and Upload it.
6. Now you can see that number starts displaying on the seven segments on the ESP32 development board.

Program:

```
void setup(void)
{
  /* define pin modes */
  /* Set pin D2-D9 as output pins */
  pinMode(4,OUTPUT);
  pinMode(12,OUTPUT);
  pinMode(33,OUTPUT);
  pinMode(32,OUTPUT);
  pinMode(25,OUTPUT);
  pinMode(26,OUTPUT);
  pinMode(27,OUTPUT);
  pinMode(5,OUTPUT);
}

void loop(void)
{
  /* print 0 */
  {
    digitalWrite(4,LOW);
    digitalWrite(12,LOW);
    digitalWrite(33,LOW);
    digitalWrite(32,LOW);
    digitalWrite(25,LOW);
    digitalWrite(26,LOW);
    digitalWrite(27,HIGH);
    digitalWrite(5,LOW);
    delay(1000);
  }
  /* print 1 */
  {
    digitalWrite(4,HIGH);
    digitalWrite(12,LOW);
    digitalWrite(33,LOW);
    digitalWrite(32,HIGH);
    digitalWrite(25,HIGH);
    digitalWrite(26,HIGH);
    digitalWrite(27,HIGH);
    digitalWrite(5,LOW);
  }
}
```

```
    delay(1000);
}
/* print 2 */
{
    digitalWrite(4,LOW);
    digitalWrite(12,LOW);
    digitalWrite(33,HIGH);
    digitalWrite(32,LOW);
    digitalWrite(25,LOW);
    digitalWrite(26,HIGH);
    digitalWrite(27,LOW);
    digitalWrite(5,LOW);
    delay(1000);
}
/* print 3 */
{
    digitalWrite(4,LOW);
    digitalWrite(12,LOW);
    digitalWrite(33,LOW);
    digitalWrite(32,LOW);
    digitalWrite(25,HIGH);
    digitalWrite(26,HIGH);
    digitalWrite(27,LOW);
    digitalWrite(5,LOW);
    delay(1000);
}
/* print 4 */
{
    digitalWrite(4,HIGH);
    digitalWrite(12,LOW);
    digitalWrite(33,LOW);
    digitalWrite(32,HIGH);
    digitalWrite(25,HIGH);
    digitalWrite(26,LOW);
    digitalWrite(27,LOW);
    digitalWrite(5,LOW);
    delay(1000);
}
/* print 5 */
{
    digitalWrite(4,LOW);
    digitalWrite(12,HIGH);
    digitalWrite(33,LOW);
    digitalWrite(32,LOW);
    digitalWrite(25,HIGH);
    digitalWrite(26,LOW);
    digitalWrite(27,LOW);
    digitalWrite(5,LOW);
    delay(1000);
}
```

```
/* print 6 */
{
  digitalWrite(4,LOW);
  digitalWrite(12,HIGH);
  digitalWrite(33,LOW);
  digitalWrite(32,LOW);
  digitalWrite(25,LOW);
  digitalWrite(26,LOW);
  digitalWrite(27,LOW);
  digitalWrite(5,LOW);
  delay(1000);
}
/* print 7 */
{
  digitalWrite(4,LOW);
  digitalWrite(12,LOW);
  digitalWrite(33,LOW);
  digitalWrite(32,HIGH);
  digitalWrite(25,HIGH);
  digitalWrite(26,HIGH);
  digitalWrite(27,HIGH);
  digitalWrite(5,LOW);
  delay(1000);
}
/* print 8 */
{
  digitalWrite(4,LOW);
  digitalWrite(12,LOW);
  digitalWrite(33,LOW);
  digitalWrite(32,LOW);
  digitalWrite(25,LOW);
  digitalWrite(26,LOW);
  digitalWrite(27,LOW);
  digitalWrite(5,LOW);
  delay(1000);
}
/* print 9 */
{
  digitalWrite(4,LOW);
  digitalWrite(12,LOW);
  digitalWrite(33,LOW);
  digitalWrite(32,LOW);
  digitalWrite(25,HIGH);
  digitalWrite(26,LOW);
  digitalWrite(27,LOW);
  digitalWrite(5,LOW);
  delay(1000);
}
}
```

EXPERIMENT NO 4

HEXKEYPAD

Aim:

To interface 4x4 HexKeypad with ESP32-Microcontroller module and display the pressed numbers on the Serial monitor.

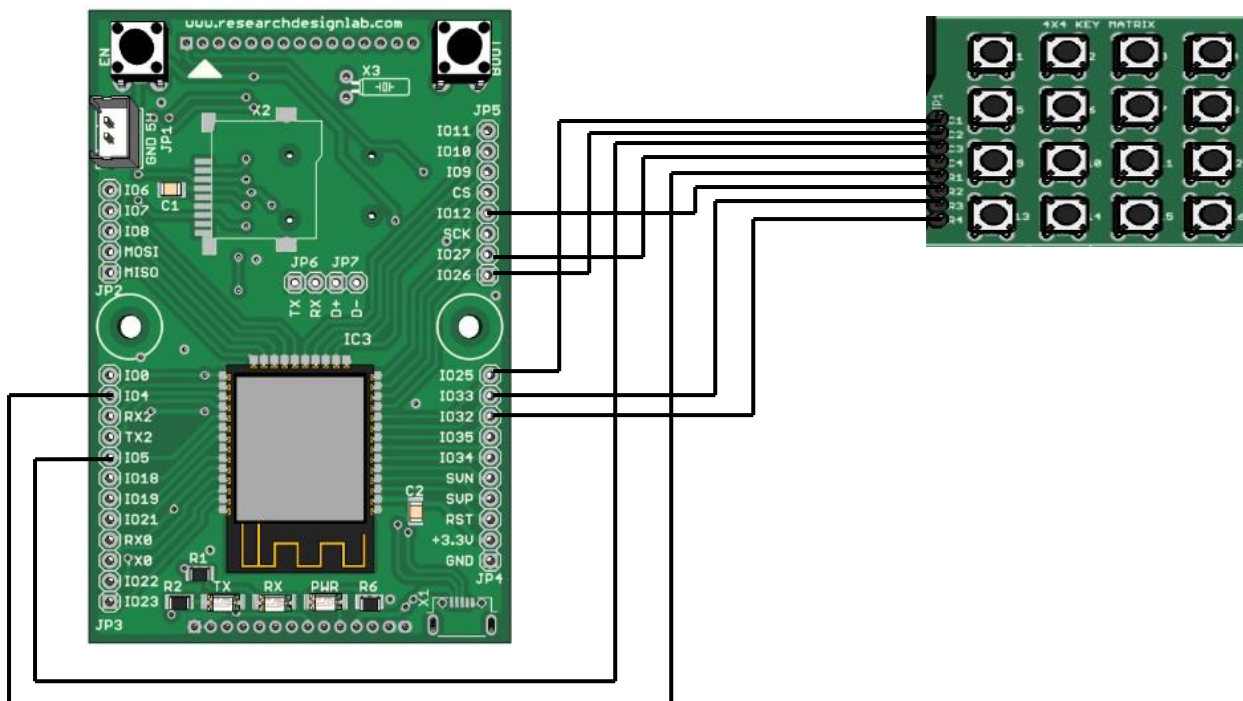
Description:

To display the pressed key on the serial monitor.

Hardware Required:

ESP32-Microcontroller Development board

Pin Connection:



Pin Mapping:

Development Board	ESP32
C1	I025
C2	I026
C3	I05
C4	I027
R1	I04
R2	I012
R3	I033
R4	I032

Procedure:

1. The above pin connection shows how to interface Hexkeypad with ESP32 board.
2. Do the connections as shown in the pin diagram and pin mapping.
3. Connect the USB cable to the board.
4. Open Arduino IDE .Select DOIT ESP32 DEVKIT V1 in boards and select COM port.
5. Now verify the program and Upload it.
6. After uploading is done open serial monitor to observe the output.
7. For a particular switch the same number displays on our serial monitor.

Program:

```
/* Include the Keypad header file */
#include<Keypad.h>
/* four rows */
const byte ROWS=4;
/* four columns */
const byte COLS=4;
/* define the symbols on the buttons of the keypads */
char hexaKeys[ROWS][COLS]={
  {'0','1','2','3'},
  {'4','5','6','7'},
  {'8','9','A','B'},
  {'C','D','E','F'}
};
/* connect to the row pinouts of the keypad */
byte rowPins[ROWS]={4,12,33,32};
/* connect to the column pinouts of the keypad */
byte colPins[COLS]={25,26,5,27};
/* initialize an instance of class NewKeypad */
Keypad customKeypad=Keypad(makeKeymap(hexaKeys),rowPins,colPins,ROWS,COLS);
/* counter initialization */
int counter=0;

void setup(void)
{
  Serial.begin(9600);
}

void loop(void)
{
  /* Key pressed is stored in customkey */
  char customKey=customKeypad.getKey();
  if(customKey)
  {
    /* Key pressed is stored in customkey */
    Serial.println(customKey);
  }
}
```


EXPERIMENT NO 5

LIQUID CRYSTAL DISPLAY

Aim:

This experiment shows how to display the message on LCD using ESP32-Microcontroller.

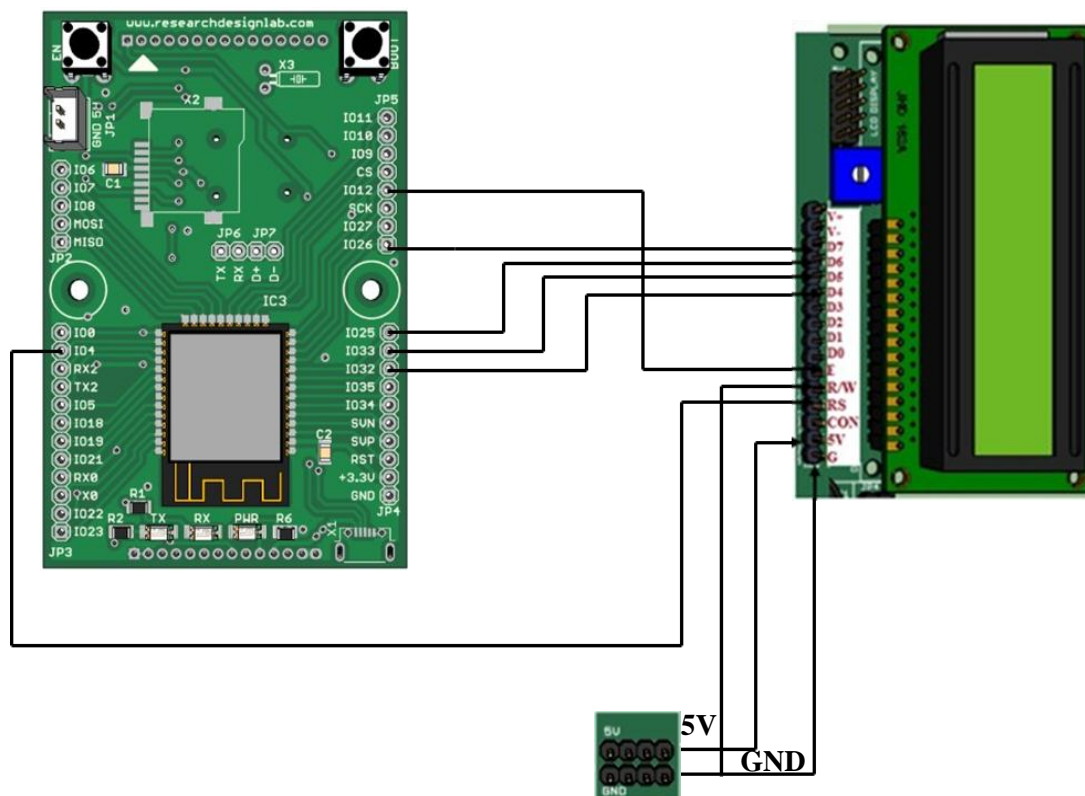
Description:

To display the message on the LCD screen.

Hardware required:

ESP32-Microcontroller Development board.

Pin connection:



Pin Mapping:

LCD	ESP32
RS	I04
E	I012
D4	I032
D5	I033
D6	I025
D7	I026
R/W	GND

Procedure:

1. The above pin connection shows how to interface LCD with ESP32 board.
2. Do the connections as shown in the pin diagram and pin mapping.
3. Connect the USB cable to the board.
4. Open Arduino IDE .Select DOIT ESP32 DEVKIT V1 in boards and select COM port.
5. Verify the program and Upload it.
6. Now you can see the output displaying the message on LCD of ESP32 microcontroller board.

Program:

```
/* Include the LCD library */
#include<LiquidCrystal.h>

/* Mapping the pins with library
rs=4,en=12,d4=32,d5=33,d6=25,d7=26 */
LiquidCrystallcd(4,12,32,33,25,26);

void setup(void)
{
  /* set baud rate */
  Serial.begin(9600);
  /* set up the LCD's number of columns and rows */
  lcd.begin(16,2);
}

void loop(void)
{
  /* 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 a message to the LCD */
  lcd.print("*WELCOME TO RDL*");

  /*print the number of seconds since reset */
  delay(5000);
  /* clear the message */
  lcd.clear();
}
```

EXPERIMENT NO 6

CONTROLLING LED BRIGHTNESS

Aim:

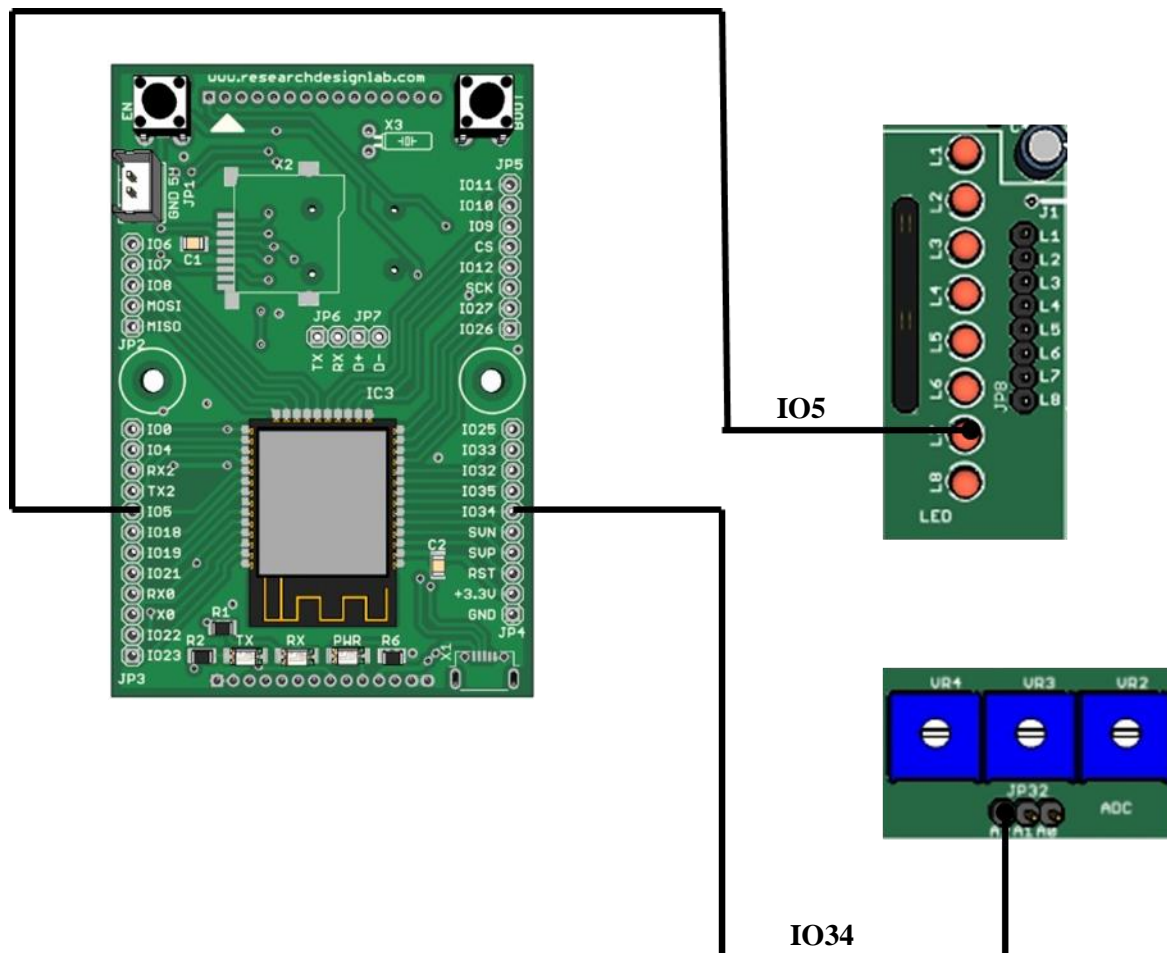
To vary the intensity of LED using potentiometer.

Description:

This experiment shows how to read an analog input value connected to onboard pot and display it on the serial monitor of the Arduino Software (IDE).

Hardware required:

ESP32-Microcontroller development board

Pin connection:

Pin Mapping:

Development Board	ESP32
Potentiometer	IO34
LED	IO5

Procedure:

1. The above pin connection shows how to control the brightness of LED using potentiometer in ESP32 board.
2. Do the connections as shown in the pin diagram and pin mapping.
3. Connect the USB cable to the board.
4. Open Arduino IDE .Select DOIT ESP32 DEVKIT V1 in boards and select COM port.
5. Verify the program and Upload it.
6. Open the serial monitor to observe the values.
7. Now using a screw driver rotate the potentiometer. The LED will be very bright at the highest point and low as you turn the potentiometer in the opposite direction.

Program:

```
void setup(void)
{
  /* set baud rate */
  Serial.begin(9600);

  /* Initialize the input pin */
  pinMode(34,INPUT);

  /* Initialize the output pin */
  pinMode(5,OUTPUT);
}

/* the loop function runs over and over again forever */
void loop(void)
{
  int outputvalue=analogRead(34);
  int value=map(outputvalue,0,1023,0,255);
  analogWrite(5,outputvalue);
  /* Print a value on the serial monitor */
  Serial.println(outputvalue);
  delay(1000);
}
```

EXPERIMENT NO 7

RTC (Real Time Clock)

Aim:

To display Date and Time on the serial monitor using ESP 32 microcontroller development board.

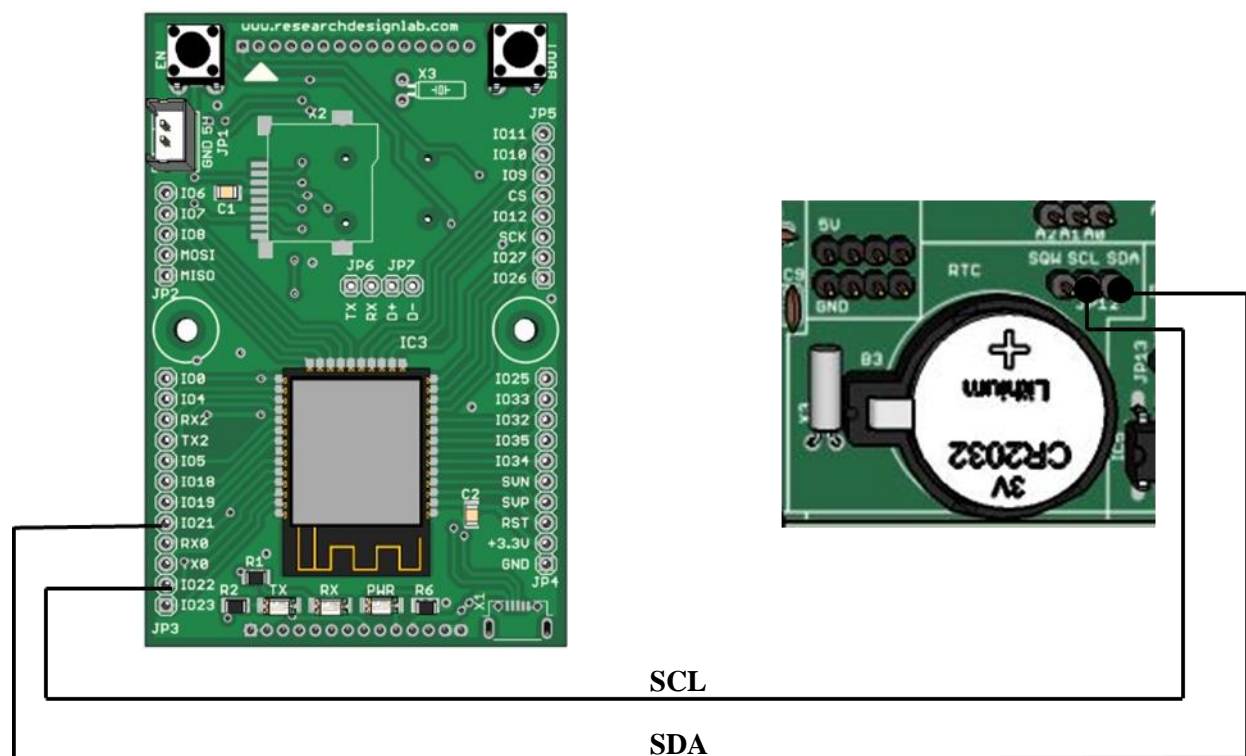
Description:

Interfacing Real Time Clock module with ESP 32 to display date and time on the serial monitor.

Hardware required:

ESP32-Microcontroller Development board

Pin connection:



Pin Mapping:

RTC	ESP32
SDA	I021
SCL	I022

Procedure:

1. The above pin connection shows how to display date and time using RTC in ESP32 board.
2. Do the connections as shown in the pin diagram and pin mapping.
3. Connect the USB cable to the board.
4. Open Arduino IDE .Select DOIT ESP32 DEVKIT V1 in boards and select COM port.
5. Verify the program and Upload it.
6. Open the serial monitor to observe the output.

Program:

```
/* Date and time functions using a DS1307 RTC connected via I2C and Wire lib */
#include<Wire.h>
#include"RTCLib.h"
RTC_DS1307RTC;
charrec;

void setup(void)
{
  Serial.begin(9600);
  Wire.begin();
  RTC.begin();
  delay(1000);
  //while(!Serial.available());
  //{
  // rec=Serial.read();
  // if(rec=='2')
  /* This line sets the RTC with an explicit date & time, for example to set
  January 16, 2019 at 12.32pm you would call:
  rtc.adjust(DateTime(2019, 1, 16, 12, 32, 45)) */
  RTC.adjust(DateTime("Mar 22 2019","10:32:45"));

  //}
}

void loop(void)
{
  /*Getting the current Time and storing it into a DateTime object */
  DateTime now=RTC.now();

  Serial.print(now.year(),DEC);
  Serial.print('/');
  Serial.print(now.month(),DEC);
  Serial.print('/');
  Serial.print(now.day(),DEC);
  Serial.print(' ');
```

```

Serial.print(now.hour(),DEC);
Serial.print(':');
Serial.print(now.minute(),DEC);
Serial.print(':');
Serial.print(now.second(),DEC);
Serial.println();
delay(1000);
}

```

Output:

```

F<  qM;  8  2019/1/13 17:11:45
2019/1/13 17:11:46
2019/1/13 17:11:47
2019/1/13 17:11:48
2019/1/13 17:11:49
2019/1/13 17:11:50
2019/1/13 17:11:51
2019/1/13 17:11:52
2019/1/13 17:11:53
2019/1/13 17:11:54
2019/1/13 17:11:55
2019/1/13 17:11:56
2019/1/13 17:11:57
2019/1/13 17:11:58
2019/1/13 17:11:59
2019/1/13 17:12:0
`&e. +pF 4} ! 2019/1/13 17:12:5
2019/1/13 17:12:6
2019/1/13 17:12:7
2019/1/13 17:12:8
2019/1/13 17:12:9
@;*  l  Œ9D&#p ^2019/1/13 17:11:45
2019/1/13 17:11:46
2019/1/13 17:11:47
2019/1/13 17:11:48
2019/1/13 17:11:49
2019/1/13 17:11:50
P+++++ c, ;F E3 

```


EXPERIMENT NO 8

SD CARD

Aim:

To read the stored directories in SD card using ESP 32 microcontroller development board.

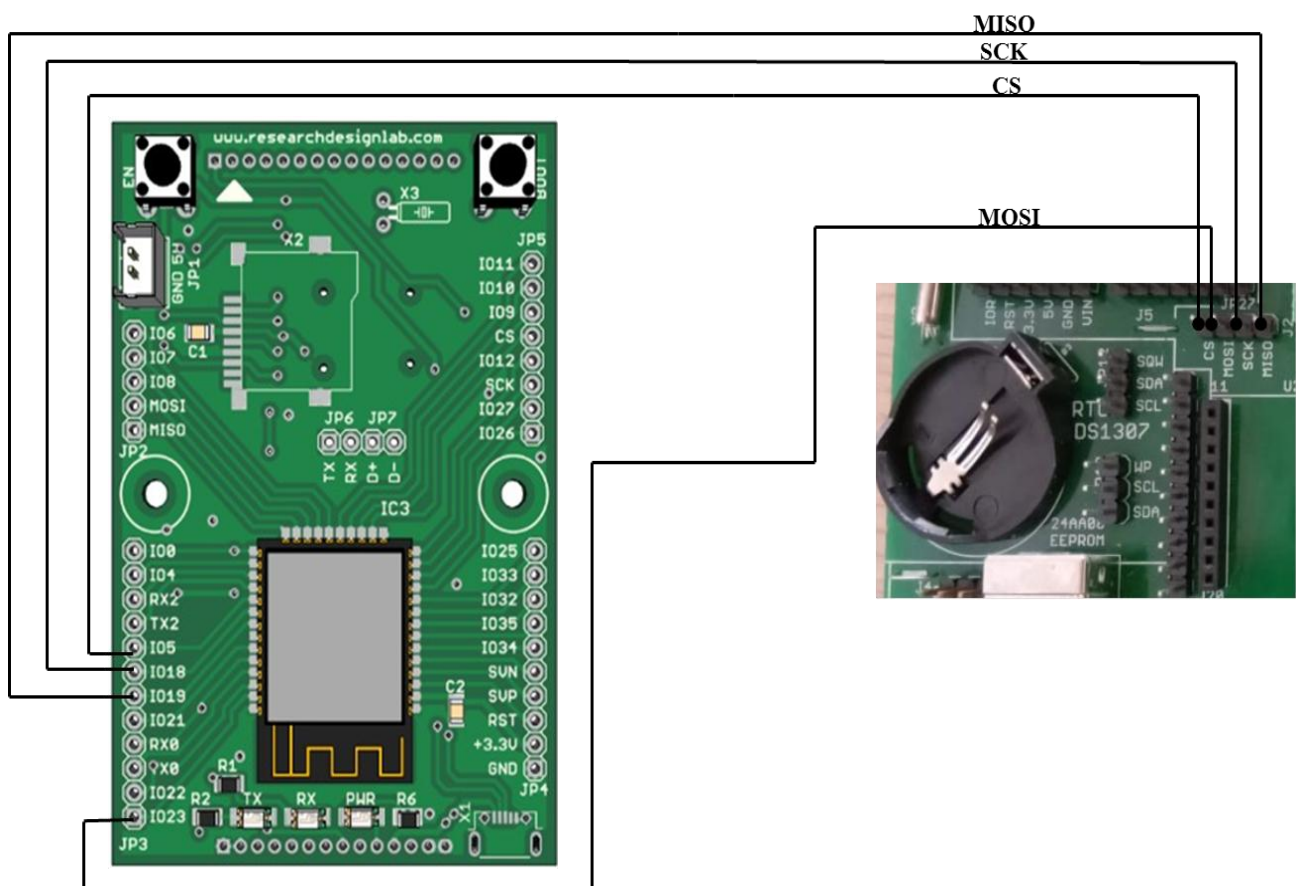
Description:

Interfacing SD card module with ESP 32 to list the directories stored in memory card.

Hardware required:

ESP32-Microcontroller Development board

Pin connection:



Pin Mapping:

SDCARD	ESP32
CS	IO5
MOSI	IO23
SCK	IO18
MISO	IO19

Procedure:

1. The above pin connection shows how to interface SD Card with ESP32 board.
2. Do the connections as shown in the pin diagram and pin mapping.
3. Insert the SD Card in the slot given below the board.
4. Connect the USB cable to the board.
5. Open Arduino IDE .Select DOIT ESP32 DEVKIT V1 in boards and select COM port.
6. Verify the program and Upload it.
7. Open the serial monitor to observe the output.

Program :

```
/*
```

```
Listfiles
```

This example shows how print out the files in a directory on a SD card

The circuit:

* SD card attached to SPI bus as follows:

** MOSI - pin 11

** MISO - pin 12

** CLK - pin 13

** CS - pin 4 (for MKRZero SD: SDCARD_SS_PIN)

created Nov 2010

by David A. Mellis

modified 9 Apr 2012

by Tom Igoe

modified 2 Feb 2014

by Scott Fitzgerald

This example code is in the public domain.

```
*/
```

```
#include<SPI.h>
```

```
#include<SD.h>
```

```
Fileroot;
```

```
void setup(void)
```

```
{
```

```
/* Open serial communications and wait for port to open: */
```

```
Serial.begin(9600);
```

```
/* wait for serial port to connect. Needed for native USB port only */
```

```
while(!Serial){
```

```
}
```

```
Serial.print("Initializing SD card...");

if(!SD.begin(4))
{
    Serial.println("initialization failed!");
    return;
}
Serial.println("initialization done.");

root=SD.open("/");

printDirectory(root,0);

Serial.println("done!");
}
/*nothing happens after setup finishes */
void loop(void){

}

void print Directory(Filedir,intnumTabs){
while(true)
{
    Fileentry= dir.openNextFile();
    /* no more files */
    if(!entry)
    {
        break;
    }
    for(uint8_ti=0;i<numTabs;i++)
    {
        Serial.print("\t");
    }
    Serial.print(entry.name());
    if(entry.isDirectory()){
        Serial.println("/");
        printDirectory(entry,numTabs+1);
    }
    else
    {
        Serial.print("\t\t");
        Serial.println(entry.size(),DEC);
    }
    entry.close();
}
}
```

Output:

```

COM48

CQ09X0i 0 Initializing SD card...initialization done.
/test.txt          1048576
/foo.txt           13
/hello.txt         13
done!

E0 p00.0IH_ 4 Initializing SD card...initialization done.
/test.txt          1048576
/foo.txt           13
/hello.txt         13
done!

0'0)F 003 G Initializing SD card...initialization done.
/test.txt          1048576
/foo.txt           13
/hello.txt         13
done!

00!D 000! 000 00 Initializing SD card...initialization done.
/test.txt          1048576
/foo.txt           13
/hello.txt         13
done!

\5a> q0 0 Initializing SD card...initialization done.
/test.txt          1048576
/foo.txt           13
/hello.txt         13
done!

☒ Autoscroll

```

EXPERIMENT NO 9

OLED

Aim:

This experiment shows how to display the message on OLED using ESP32-Microcontroller.

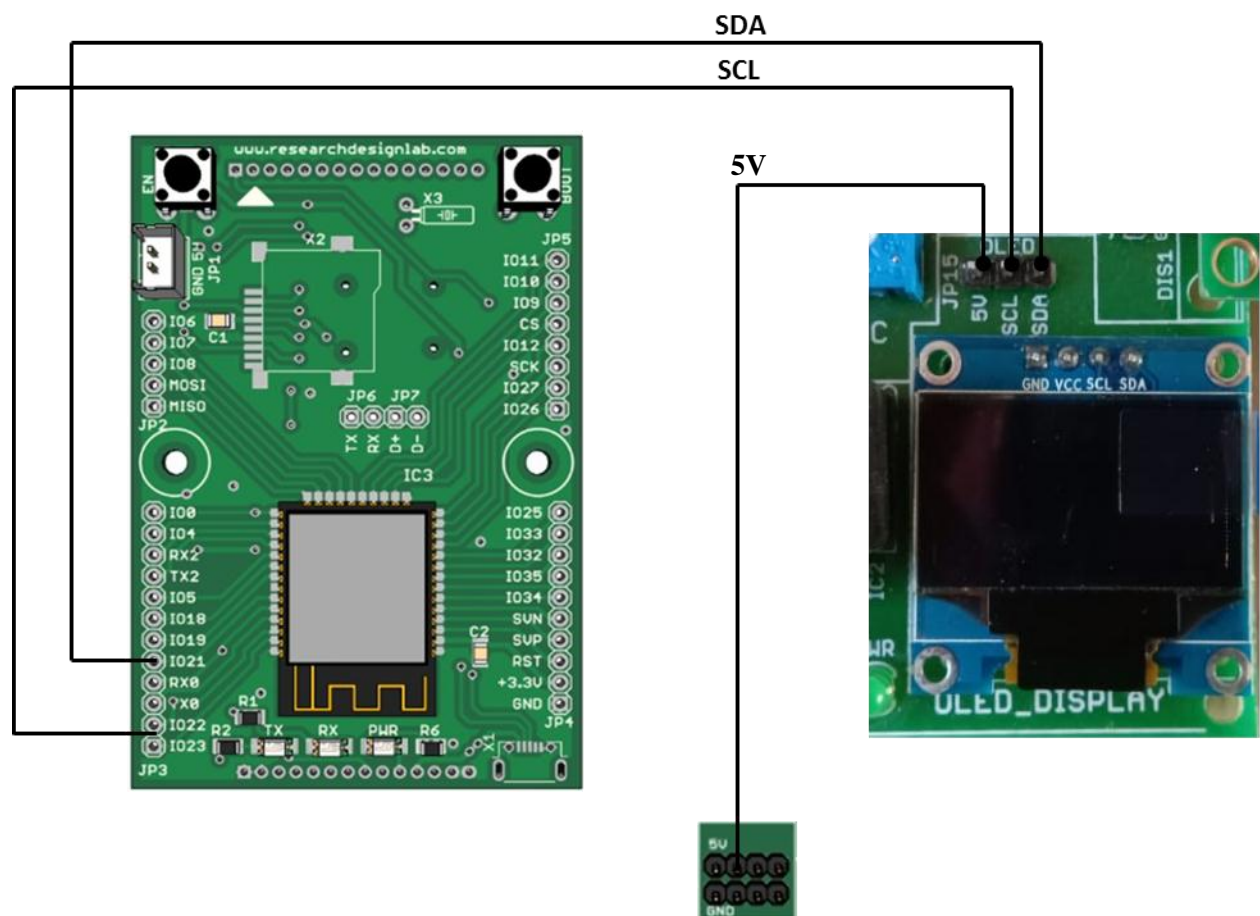
Description:

To display message on OLED screen.

Hardware required:

ESP32-Microcontroller Development board

Pin connection:



Pin Mapping:

OLED	ESP32
5V	5V
SCL	IO22
SDA	IO21

Procedure:

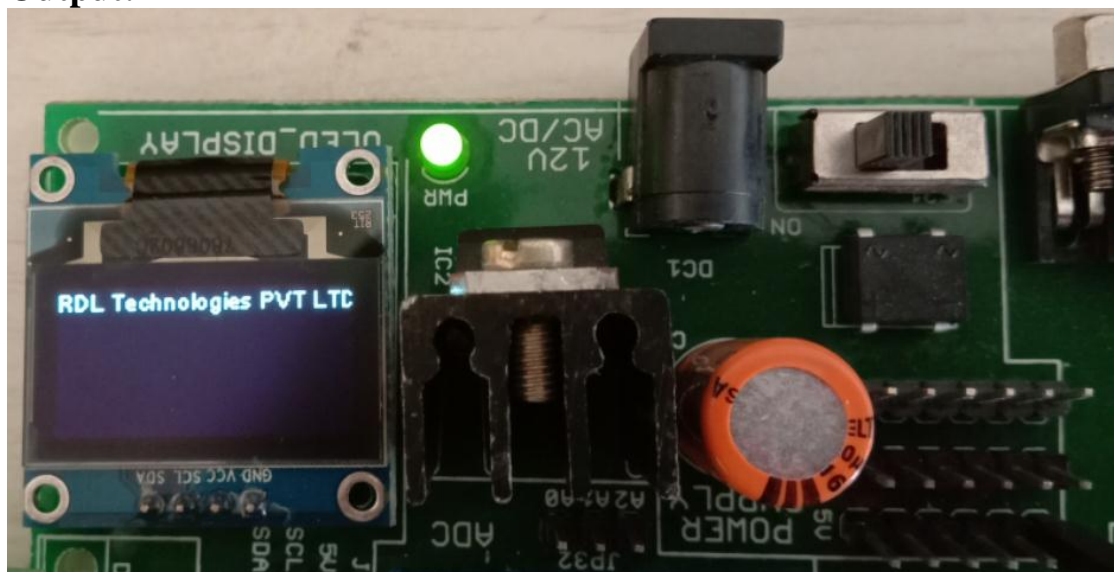
1. The above pin connection shows how to interface OLED with ESP32 board.
2. Do the connections as shown in the pin diagram and pin mapping.
3. Connect the USB cable to the board.
4. Open Arduino IDE .Select DOIT ESP32 DEVKIT V1 in boards and select COM port.
5. Verify the program and Upload it.
6. Now you can see the output displaying the message on OLED of ESP32 microcontroller board.

Program:

```
#include<Wire.h>
#include"SSD1306.h"
/* SDA=21,SCL=22 */
SSD1306 display(0x3c,21,22);

void setup(void)
{
    display.init();
    /* display message in OLED */
    display.drawString(0,0,"RDL Technologies PVT. LTD.");
    display.display();
}

void loop(void)
{
}
```

Output:

EXPERIMENT NO 10

VIBRATION SENSOR

Aim:

To extract information from vibration sensor.

Description:

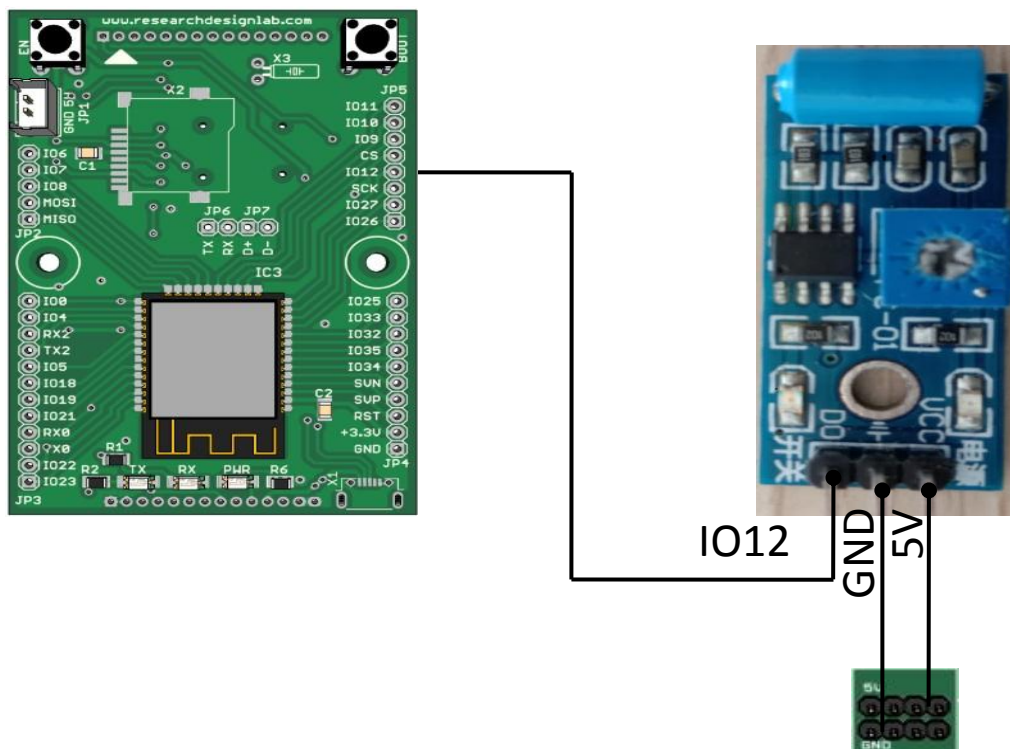
To learn how to read values from a vibration sensor connected to analog pin using ESP32-Microcontroller.

Hardware required:

ESP32-Microcontroller Development board

Vibration sensor

Pin connection:



Pin Mapping:

VIBRATION SENSOR	ESP32
5V	5V
GND	GND
A/O	IO12

Procedure:

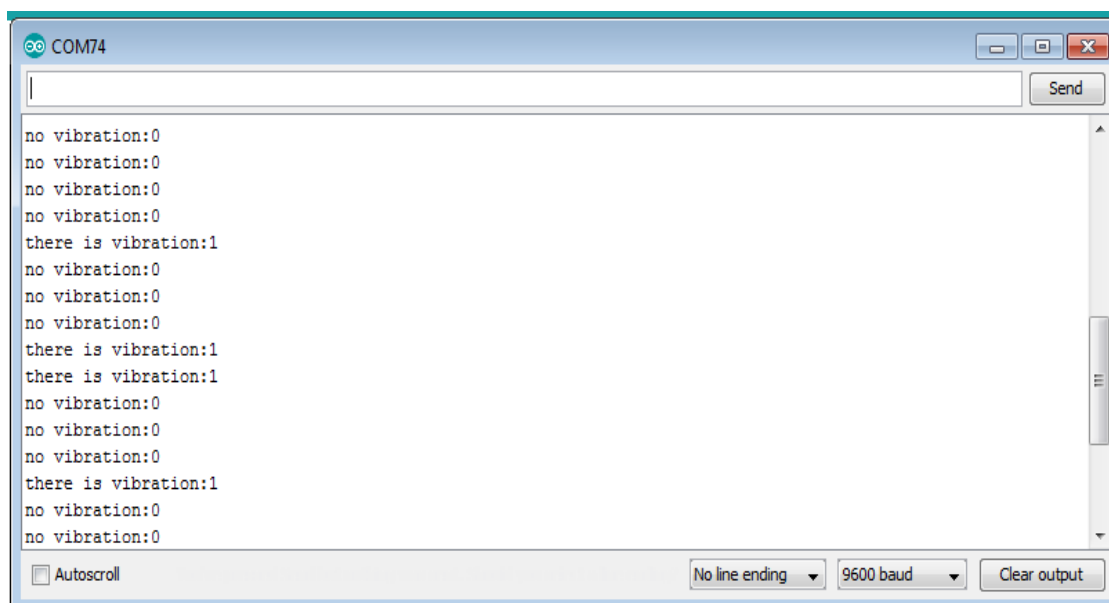
1. The above pin connection shows how to read values from a vibration sensor using ESP32 board.
2. Do the connections as shown in the pin diagram and pin mapping.
3. Connect the USB cable to the board.
4. Open Arduino IDE .Select DOIT ESP32 DEVKIT V1 in boards and select COM port.
5. Verify the program and upload it.
6. Now you can see the output on the serial monitor.

Program:

```
void setup()
{
    pinMode(35,INPUT);
    Serial.begin(9600);
}

void loop()
{
    int sensorvalue = analogRead(35);
    if(sensorvalue>1000)
    {
        Serial.print("Sensor value is : ");
        Serial.println(sensorvalue);
        delay(2000);
    }
}
```

Output:



EXPERIMENT NO 11

TEMPERATURE SENSOR

Aim:

To extract information from temperature sensor.

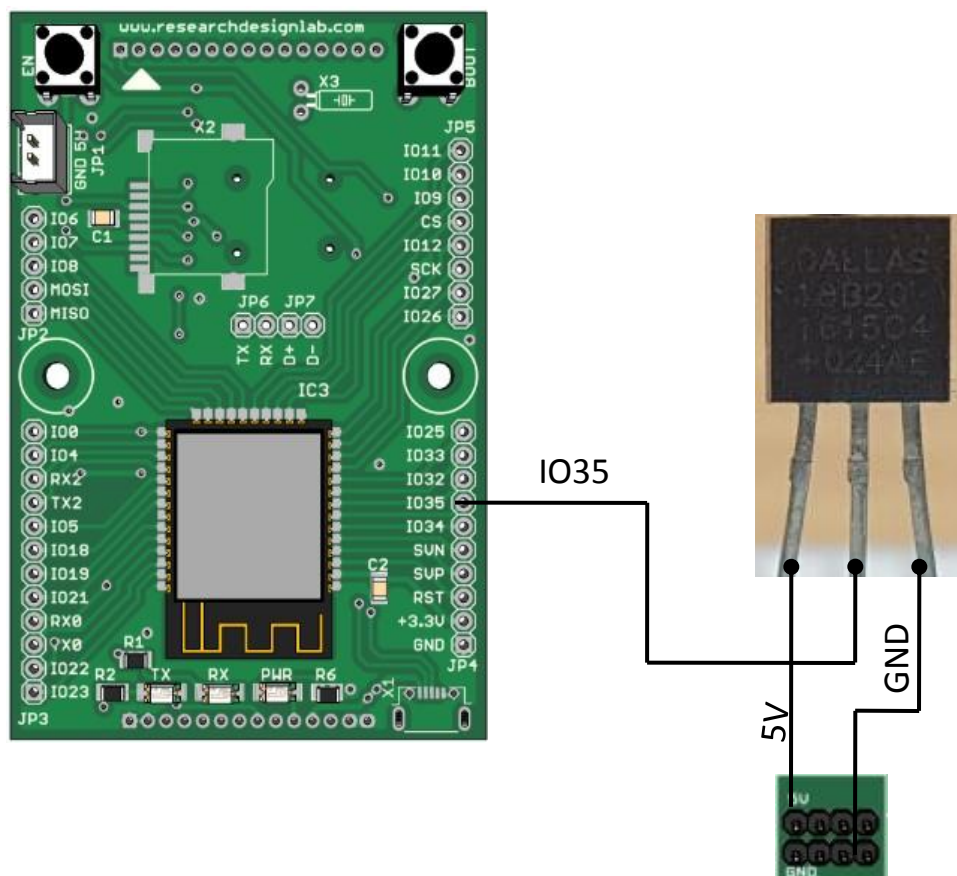
Description:

To learn how to read values from a temperature sensor connected to analog pin using ESP32-Microcontroller.

Hardware required:

ESP32-Microcontroller Development board

Temperature sensor

Pin connection:

Pin Mapping:

TEMPERARUTE SENSOR	ESP32
5V	5V
GND	GND
A/O	IO35

Procedure:

1. The above pin connection shows how to read values from a temperaturesensor using ESP32 board.
2. Do the connections as shown in the pin diagram and pin mapping.
3. Connect the USB cable to the board.
4. Open Arduino IDE .Select DOIT ESP32 DEVKIT V1in boards and select COM port.
5. Verify the program and Upload it.
6. Now you can see the output on the serial monitor.

PROGRAM:

```

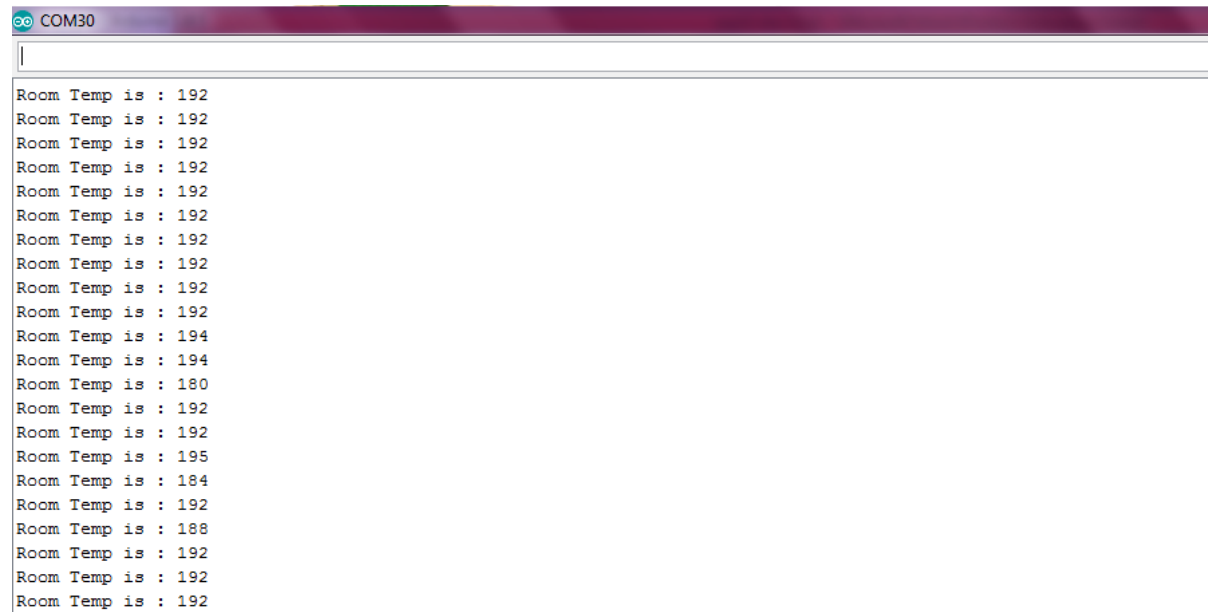
float temp;
float value;
float valuec;
int tempPin = 35;

void setup() {
  Serial.begin(9600);
}

void loop() {
  temp = analogRead(tempPin);
  // read analog volt from sensor and save to variable temp
  value=( temp/2048.0)*3300;
  valuec=value*0.1; //converts to degree celsius
  // convert the analog volt to its temperature equivalent
  Serial.print("temperature in C = ");
  Serial.println(valuec); // display temperature value
  Serial.println();
  Serial.print("temperature = ");
  Serial.println(value);
  delay(1000); // update sensor reading each one second
}

```

OUTPUT:



```
COM30
Room Temp is : 192
Room Temp is : 192
Room Temp is : 192
Room Temp is : 192
Room Temp is : 192
Room Temp is : 192
Room Temp is : 192
Room Temp is : 192
Room Temp is : 192
Room Temp is : 192
Room Temp is : 194
Room Temp is : 194
Room Temp is : 180
Room Temp is : 192
Room Temp is : 192
Room Temp is : 195
Room Temp is : 184
Room Temp is : 192
Room Temp is : 188
Room Temp is : 192
Room Temp is : 192
Room Temp is : 192
```

EXPERIMENT NO 12

IR(Infrared) SENSOR

Aim:

To extract information from IR sensor.

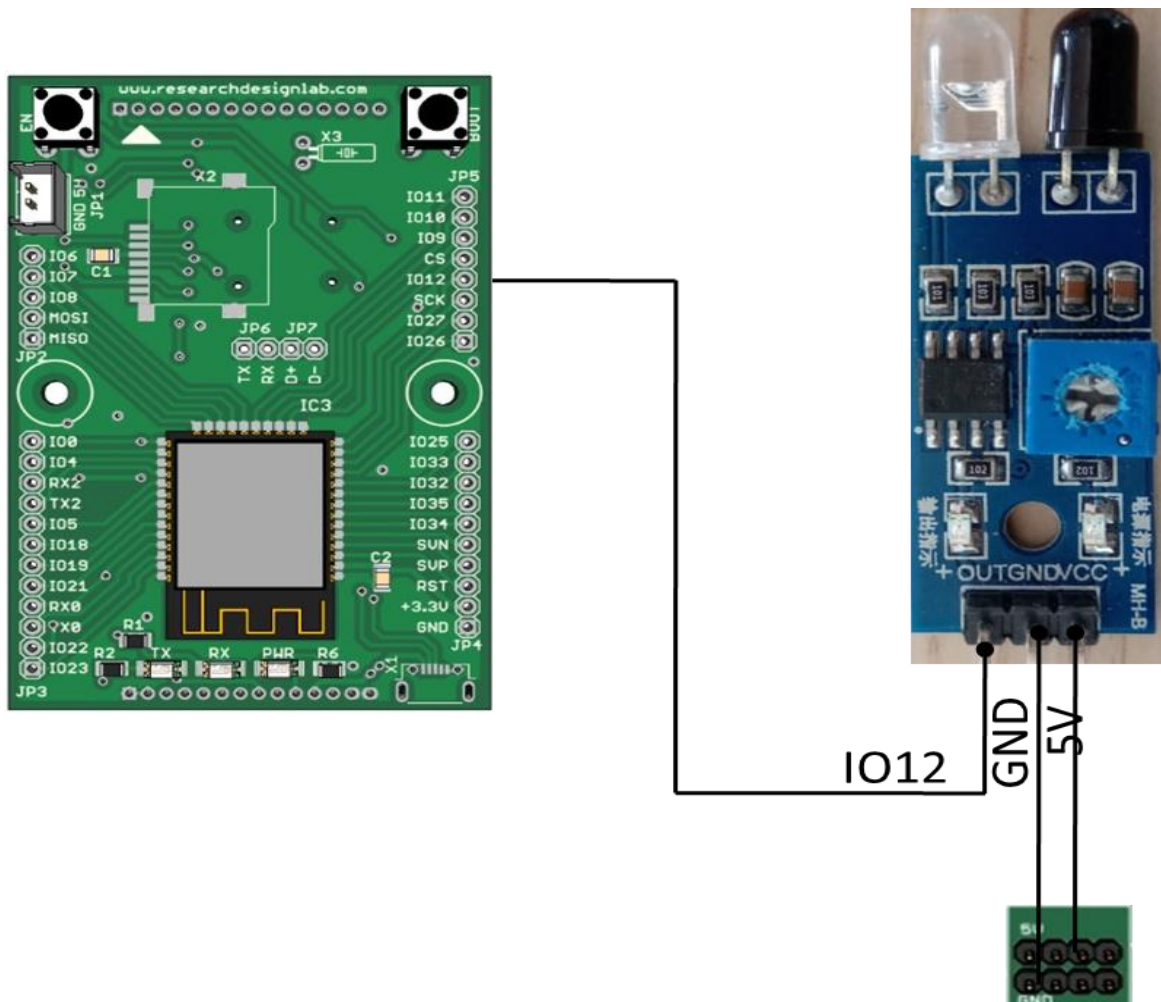
Description:

To learn how to read values from an IR sensor using ESP32-Microcontroller.

Hardware required:

ESP32-Microcontroller Development board

PIR sensor

Pin connection:

Pin Mapping:

IR SENSOR	ESP32
5V	5V
GND	GND
A/O	IO12

Procedure:

1. The above pin connection shows how to read values from a IR sensor using ESP32 board.
2. Do the connections as shown in the pin diagram and pin mapping.
3. Connect the USB cable to the board.
4. Open Arduino IDE .Select DOIT ESP32 DEVKIT V1 in boards and select COM port.
5. Verify the program and upload it.
6. Now you can see the output on the serial monitor.

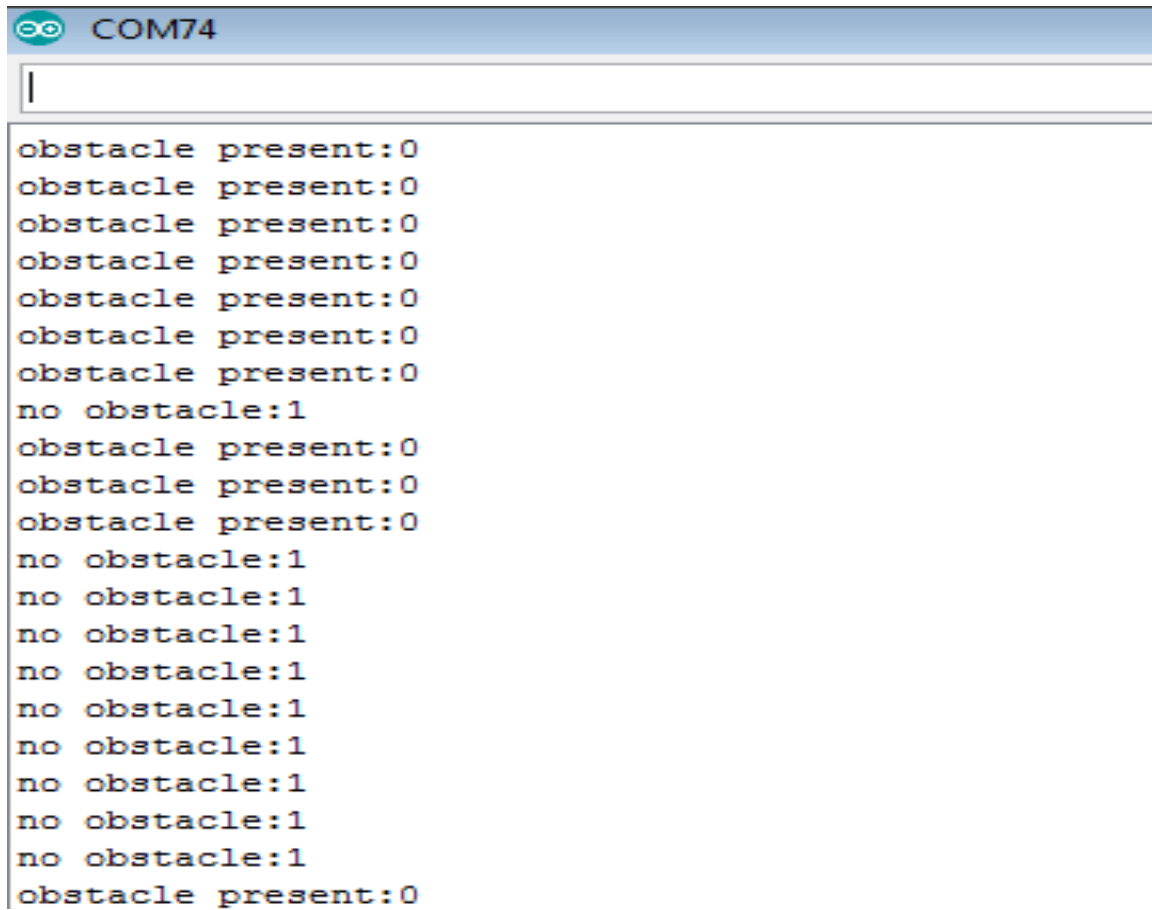
Program:

```

/* the setup function runs once when you press reset or power the board */
void setup(void)
{
  /* initialize digital pin 12 as an input */
  pinMode(12,INPUT);
  Serial.begin(9600);
}

/* the loop function runs over and over again forever */
void loop(void)
{
  int value=digitalRead(12);
  if(digitalRead(12)==HIGH)
  {
    Serial.print("no obstacle:");
    Serial.println(value);
    delay(1000);
  }
  else if(digitalRead(12)==LOW)
  {
    Serial.print("obstacle present:");
    Serial.println(value);
    delay(1000);
  }
}

```

Output:

```
obstacle present:0
obstacle present:0
obstacle present:0
obstacle present:0
obstacle present:0
obstacle present:0
obstacle present:0
no obstacle:1
obstacle present:0
obstacle present:0
obstacle present:0
no obstacle:1
no obstacle:1
no obstacle:1
no obstacle:1
no obstacle:1
no obstacle:1
no obstacle:1
no obstacle:1
no obstacle:1
obstacle present:0
```

EXPERIMENT NO 13

MQTT

Aim:

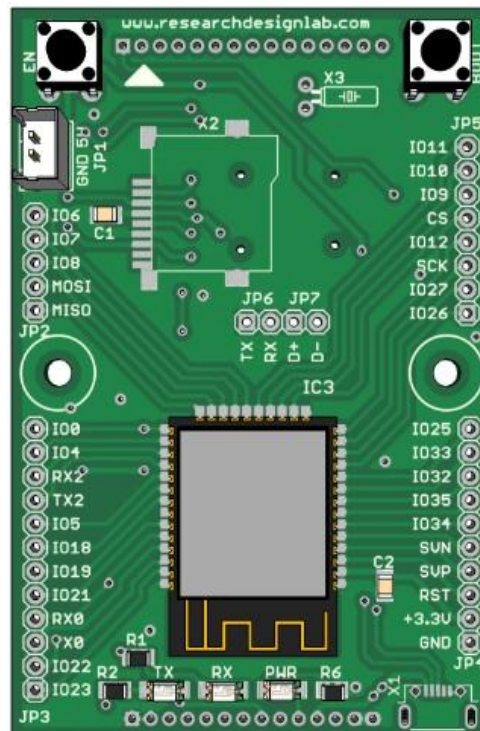
To interface Wi-Fi module with ESP32 to receive data from cloud using RDL ESP32 Development board.

Description:

Interfacing wi-fi module with the cloud MQTT to receive and send data using esP32 microcontroller.

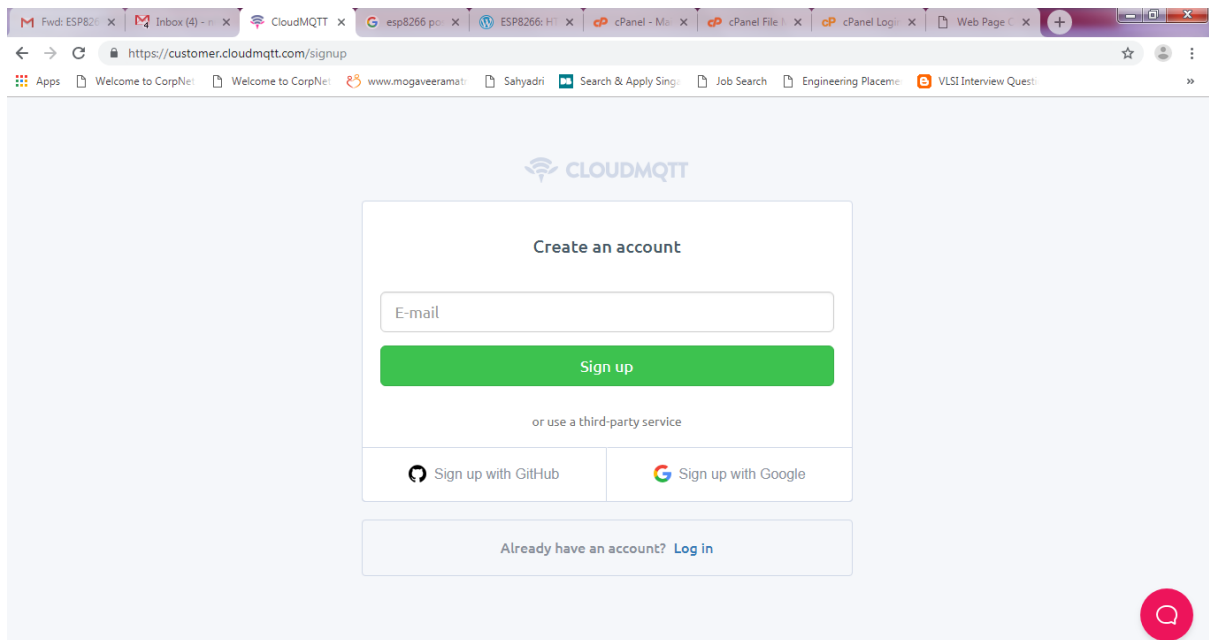
Hardware required:

ESP32-Microcontroller development board.

Pin connection:

Procedure:**Uploading and Receiving Data from Cloud using ESP32**

1. Create an Account in www.cloudmqtt.com



CloudMQTT

Create an account

E-mail

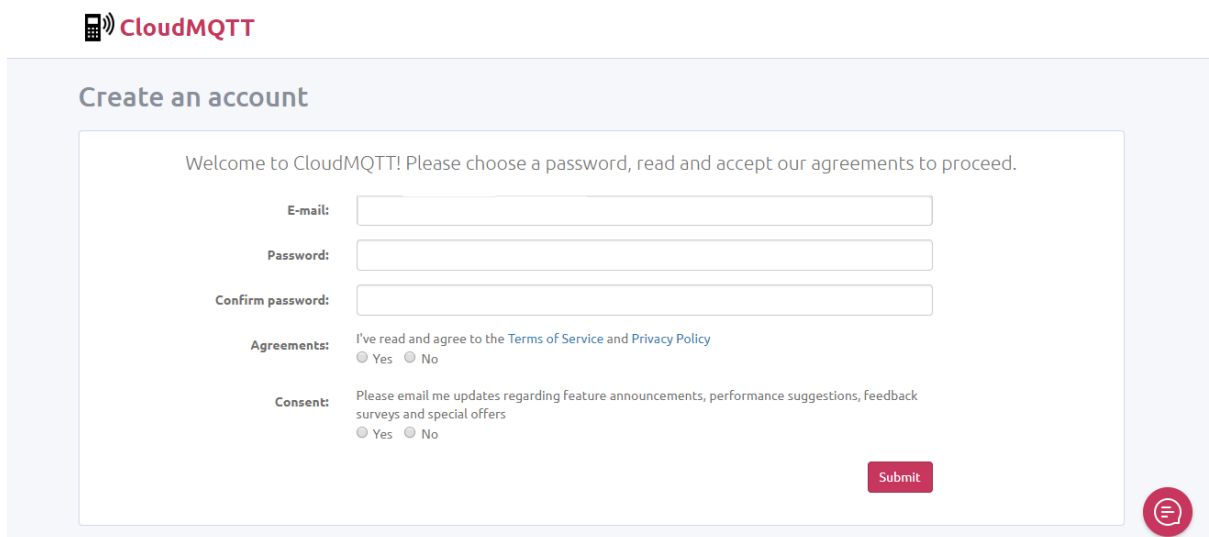
Sign up

or use a third-party service

Sign up with GitHub

Sign up with Google

Already have an account? [Log in](#)



CloudMQTT

Create an account

Welcome to CloudMQTT! Please choose a password, read and accept our agreements to proceed.

E-mail:

Password:

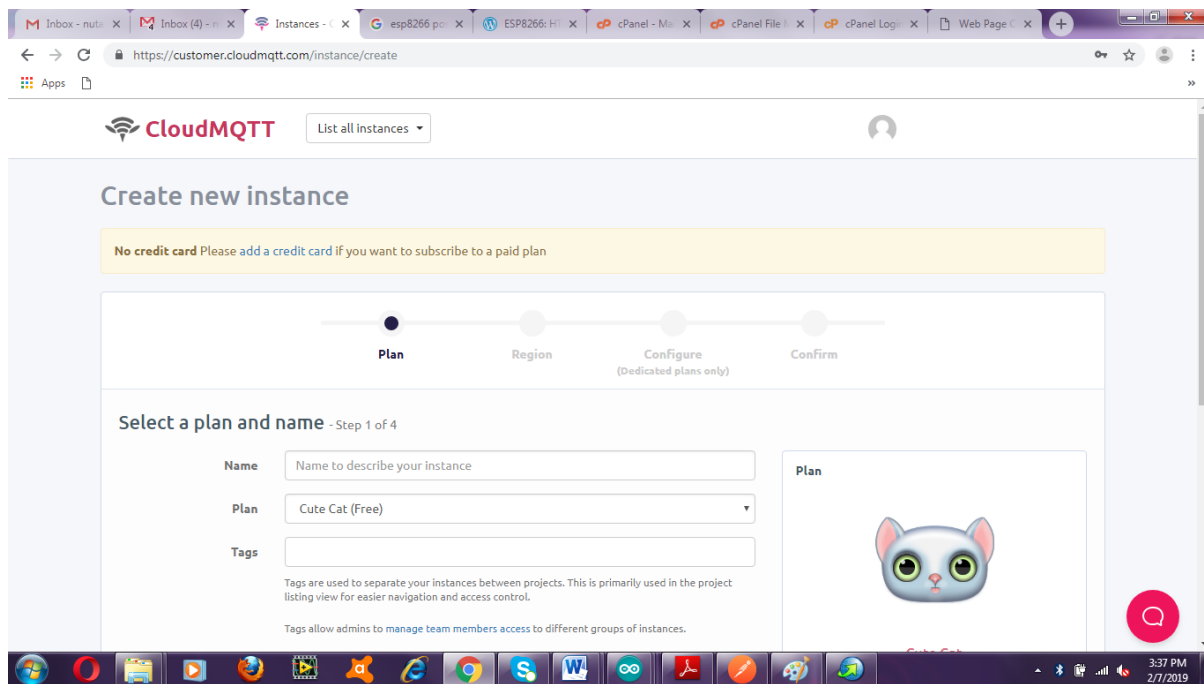
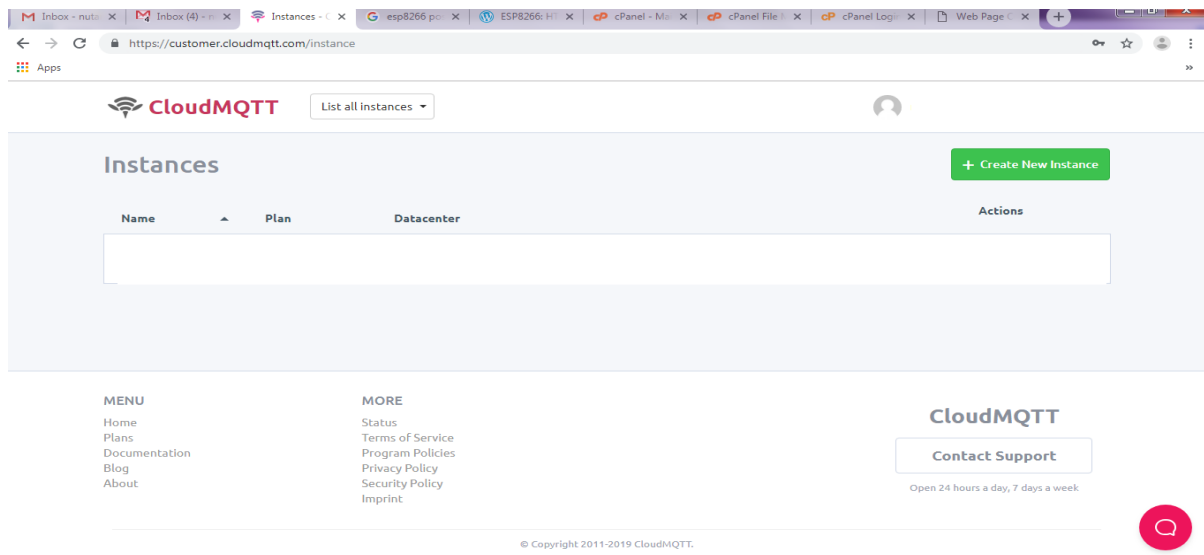
Confirm password:

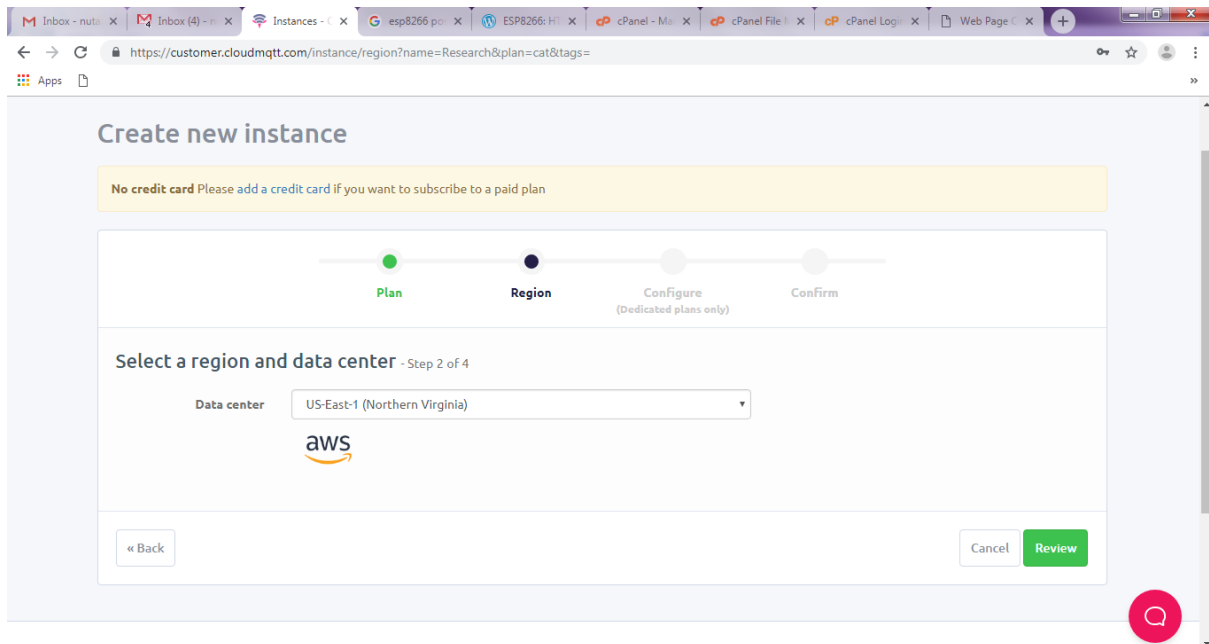
Agreements: I've read and agree to the [Terms of Service](#) and [Privacy Policy](#)
☐ Yes ☐ No

Consent: Please email me updates regarding feature announcements, performance suggestions, feedback surveys and special offers
☐ Yes ☐ No

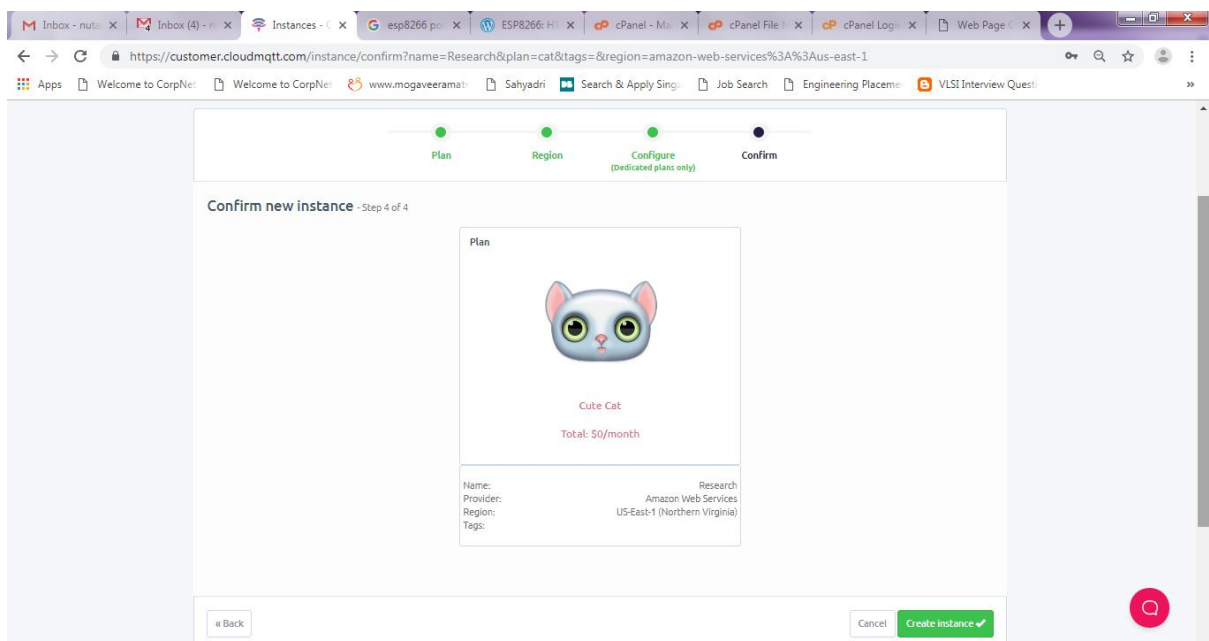
Submit

2. Create an instance which will be used in the program to publish data to the cloud.

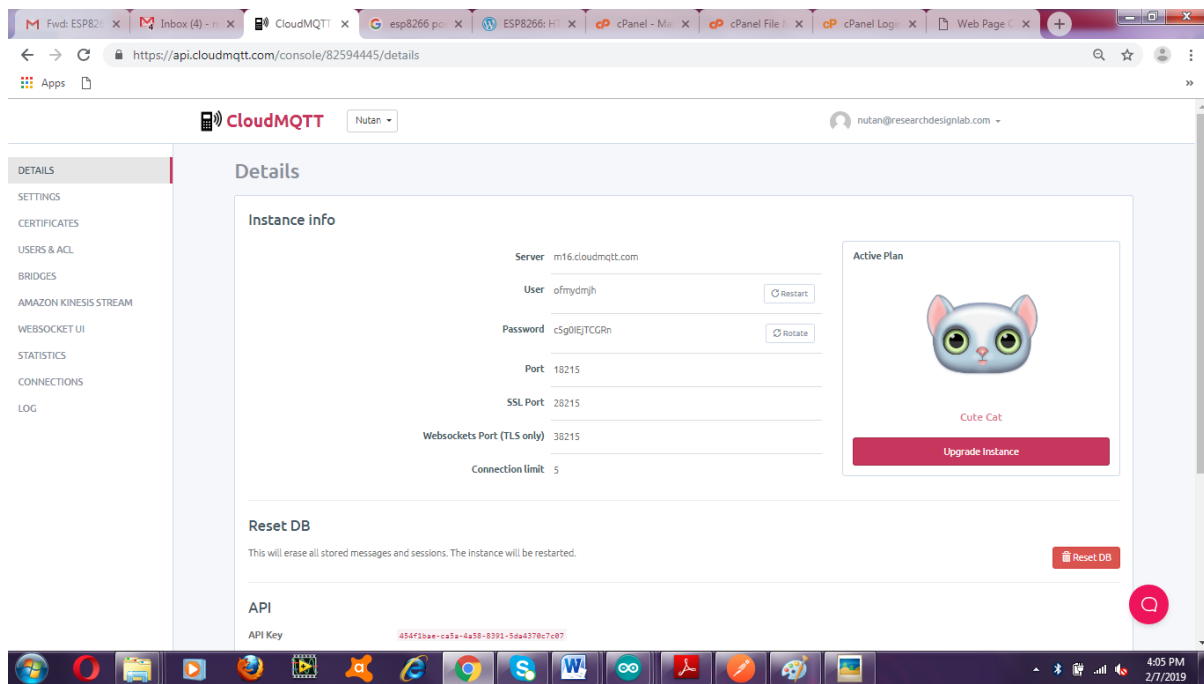




3. Select the Instance name (test_1) and use the details of the same in the code. Include the code.



- 4.

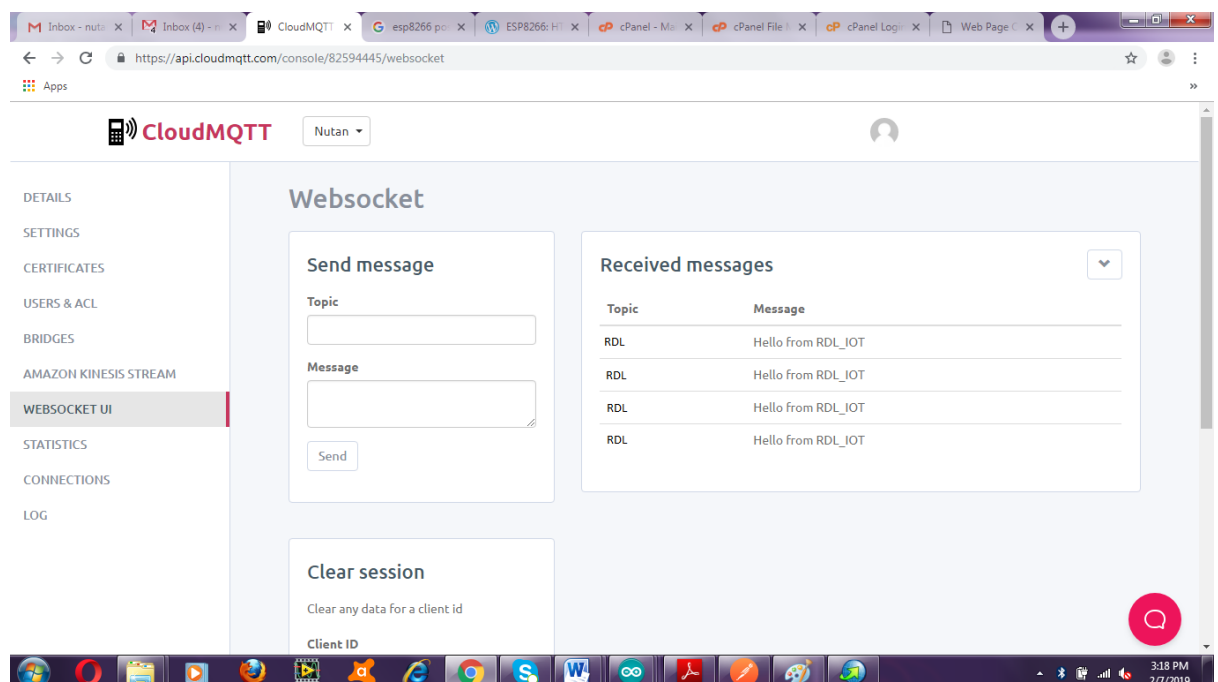


5. Enter the details given under server, port, user, ssl port in the given program.

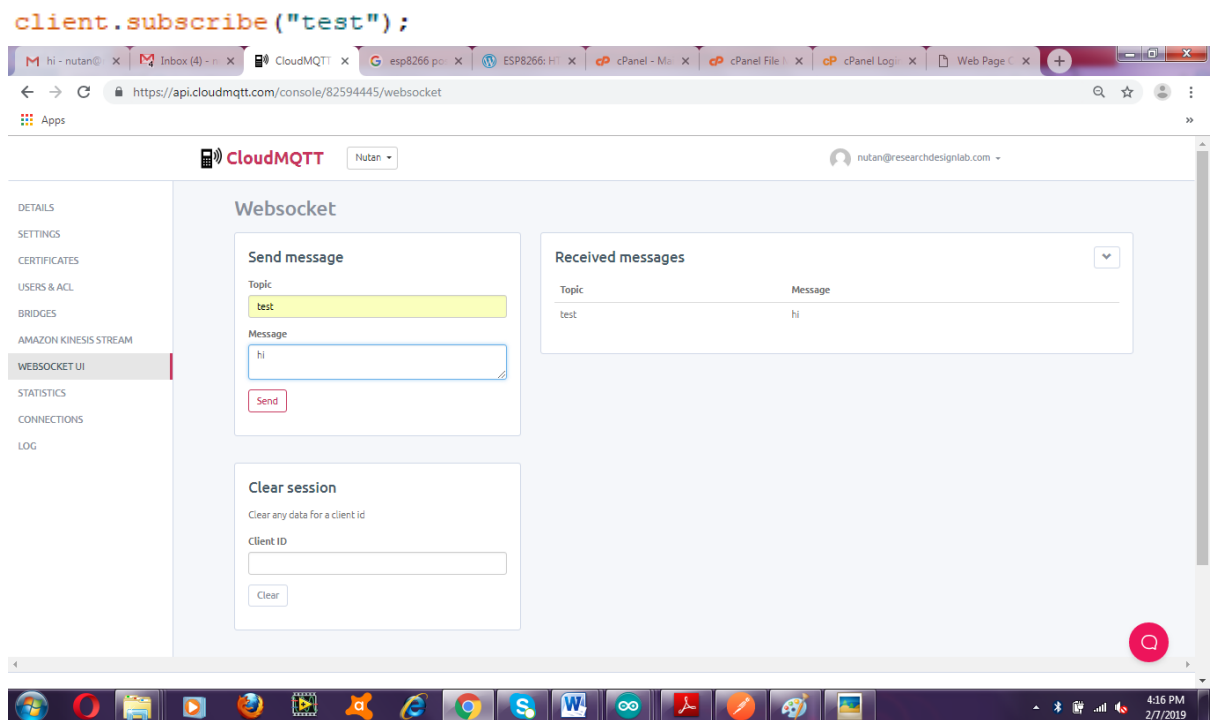
```
const char* mqttServer = "m16.cloudmqtt.com";
const int mqttPort = 18215;
const char* mqttUser = "ofmydmjh";
const char* mqttPassword = "c5g0IEjTCGRn";
```

6. In order to send a message to the CloudMQTT, include the line

```
client.publish("topic_name", "Message");
```

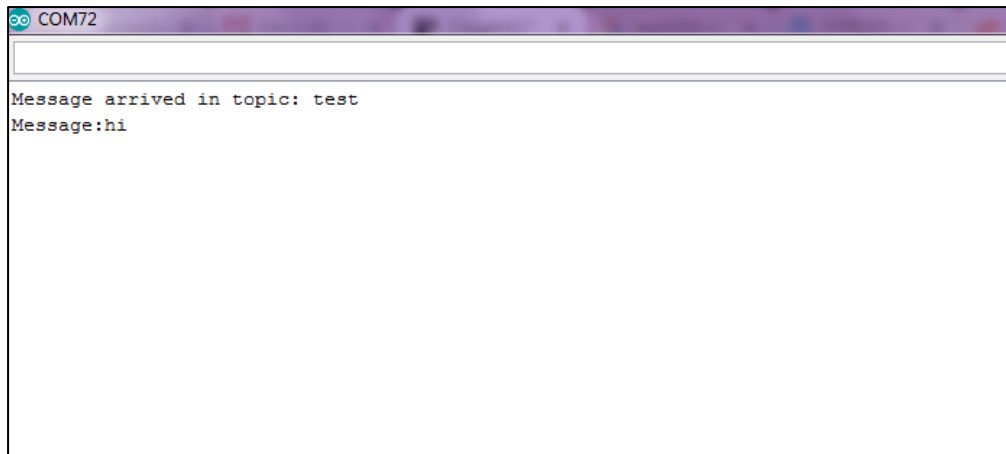


7. To send data from CloudMQTT ,include the line `client.subscribe("Topic_name");`



8. The following code prints the received data from the cloud in the Serial Monitor

```
void callback(char* topic, byte* payload, unsigned int length)
{
  uint8_t s;
  Serial.print("Message arrived in topic: ");
  Serial.println(topic);
  Serial.print("Message:");
  for (int i = 0; i < length; i++)
  {
    s= payload[i];
    Serial.write(s);
  }
}
```



9. The final Program:

```
#include <WiFi.h>
#include <PubSubClient.h>
const char* ssid = "userid";
const char* password = "*****";
/* Details from the instance created */
const char* mqttServer = "m16.cloudmqtt.com";
const int mqttPort = 18215;
const char* mqttUser = "ofmydmjh";
const char* mqttPassword = "c5g0IEjTCGRn";
WiFiClient espClient;
PubSubClient client(espClient);
void setup(void)
{
  Serial.begin(115200);
  WiFi.begin(ssid, password);
  /* Connecting ESP8266 to WiFi */
  while (WiFi.status() != WL_CONNECTED)
  {
    delay(500);
    Serial.write('.');
  }
  Serial.println("Connected to the WiFi network");
  client.setServer(mqttServer, mqttPort);
  client.setCallback(callback);
  /* Connecting to CloudMqtt */
  while (!client.connected())
  {
    Serial.println("Connecting to MQTT...");
    if (client.connect("ESP32Client", mqttUser, mqttPassword))
    {
      Serial.println("connected");
    }
    else
  }
```

```
{  
Serial.print("failed with state ");  
Serial.print(client.state());  
delay(2000);  
}  
}  
/* Sending message to Topic "test1" */  
client.publish("ABC", "Hello from RDL_IOT");  
client.subscribe("test"); //Receives message sent to the topic "test"  
}  
/* This function is used to print the incoming data sent to the topic "test" */  
void callback(char* topic, byte* payload, unsigned int length)  
{  
uint8_t s;  
Serial.print("Message arrived in topic: ");  
Serial.println(topic);  
Serial.print("Message:");  
for (int i = 0; i < length; i++)  
{  
s= payload[i];  
Serial.write(s);  
}  
}  
  
void loop(void)  
{  
client.loop();  
}
```

Output:

```
COM72
|
|
no 0 tail 12 room 4
load:0x40078000,len:9280
load:0x40080400,len:5848
entry 0x40080698
...Connected to the WiFi network
Connecting to MQTT...
connected
ets Jun  8 2016 00:22:57

rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:1
load:0x3fff0018,len:4
load:0x3fff001c,len:928
no 0 tail 12 room 4
load:0x40078000,len:9280
load:0x40080400,len:5848
entry 0x40080698
.....ets Jun  8 2016 00:22:57

rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:1
load:0x3fff0018,len:4
load:0x3fff001c,len:928
no 0 tail 12 room 4
load:0x40078000,len:9280
load:0x40080400,len:5848
entry 0x40080698
...Connected to the WiFi network
Connecting to MQTT...
connected
Message arrived in topic: test
Message:hello good morning

☒ Autoscrol
```

EXPERIMENT NO 14

JSON

Aim:

To interface Wi-Fi module with ESP32 to send data to server using RDL ESP32 Development board.

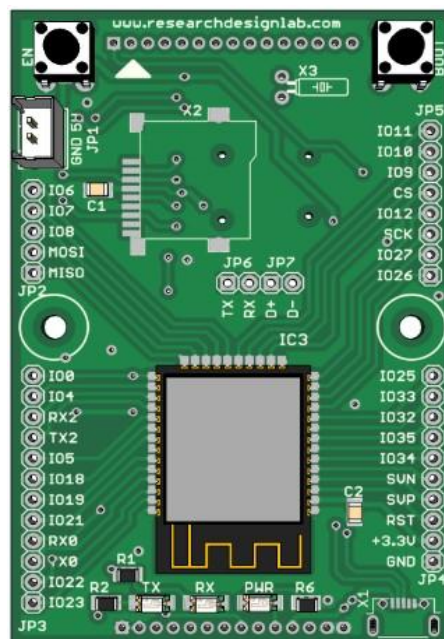
Description:

Interfacing wi-fi module to send data to the server using ESP32 microcontroller.

Hardware required:

ESP32-Microcontroller development board.

Pin connection:



Pin Mapping:

TEMPERARUTE SENSOR	ESP32
3.3V	3.3V
GND	GND
A/O	IO35

SOIL MOISTURE SENSOR	ESP32
5V	5V
GND	GND
A/O	IO34

Procedure:

1. Connect the USB cable to the ESP32 development board.
2. Do the connections as shown in the pin diagram and pin mapping.
3. Open Arduino IDE .Select DOIT ESP32 DEVKIT V1 in boards and select COM port.
4. Verify the program and upload it.
5. Now you can see wifi connected along with the IP address on the serial monitor.
6. To check the inserted values of gas and temperature use the server address and check it on the browser.

Program:

```
/*
 * This sketch sends data via HTTP POST requests to data.sparkfun.com service.
 *
 * You need to get stream Id and private Key at data.sparkfun.com and paste them
 * below. Or just customize this script to talk to other HTTP servers.
 */

#include <WiFi.h>

char ssid[] = "your username";
char password[] = "your password";

const char* host = "xyz.com";
const char* streamId = ".....";
const char* privateKey = ".....";
//http://www.xyz.com/abcd/def/postone/postinsert.php

WiFiClient esp32client;
void setup() {
  Serial.begin(115200);
  delay(10);
  pinMode(35,INPUT); //temperature
  pinMode(34,INPUT); //soil
  // We start by connecting to a WiFi network

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

  /* Explicitly set the ESP8266 to be a WiFi-client, otherwise, it by default,
   would try to act as both a client and an access-point and could cause
   network-issues with your other WiFi-devices on your WiFi-network. */
  //WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
```



```
    delay(500);
    Serial.print(".");
}

Serial.println("");
Serial.println("WiFi connected");
Serial.println("IP address: ");
Serial.println(WiFi.localIP());
}

int value = 0;

void loop() {

String url="/abcd/def/postone/postinsert.php?";

String jason_string="";

int temp=analogRead(35);
float value=(( temp/2048.0)*3300)*0.1;
jason_string="{\"temp\":";
jason_string+=value;
jason_string+=\", \"soil\":";
int soil=analogRead(34);
jason_string+=soil;
jason_string+=\", \"vibrate\":";
jason_string+=\"100\"}";

while(Serial.available()>0) {Serial.read();}

Serial.print("connecting to ");
Serial.println(host);

// Use WiFiClient class to create TCP connections

const int httpPort = 80;
if (!esp32client.connect(host, httpPort)) {
    Serial.println("connection failed");
    return;
}

// We now create a URI for the request

//String url="/energyconsumption/insert.php?one=1&two=3&three=2";
Serial.print("Requesting URL:");
Serial.println(url);
//String jason_string="{\"gas\": \"40\", \"temp\": \"70\"}";
// This will send the request to the server
esp32client.print(String("POST ") + url + " HTTP/1.0\r\n" +
    "Host: " + host + "\r\n" +
```

```

    "Accept: *" + "/" + "*\r\n" +
    "Content-Length: " + json_string.length() + "\r\n" +
    "Content-Type: application/json\r\n" +
    "\r\n" + json_string
    + "\r\n");
unsigned long timeout = millis();
while (esp32client.available() == 0) {
    if (millis() - timeout > 5000) {
        Serial.println(">>> Client Timeout !");
        esp32client.stop();
        return;
    }
}

// Read all the lines of the reply from server and print them to Serial
while(esp32client.available()){
    String line = esp32client.readStringUntil('\r');
    Serial.print(line);
}

Serial.println();
delay(10000);
// Serial.println("closing connection");
}

```

Output:

1. Open serial monitor you can see Wi-Fi connected along with IP address

```

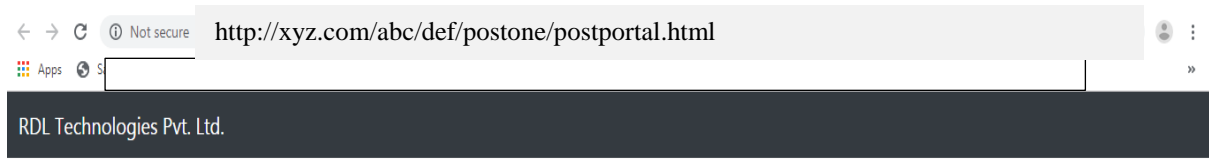
COM84
Connecting to chethz
.....
WiFi connected
IP address:
192.168.43.28
connecting to varmatrix.com
Requesting URL:/surendrardl/nitk/postone/postinsert.php?
HTTP/1.1 200 OK
Date: Fri, 21 Jun 2019 10:55:13 GMT
Server: Apache
Vary: Accept-Encoding
Connection: close
Content-Type: text/html

value inserted
connecting to varmatrix.com
Requesting URL:/surendrardl/nitk/postone/postinsert.php?
>>> Client Timeout !
connecting to varmatrix.com
Requesting URL:/surendrardl/nitk/postone/postinsert.php?
HTTP/1.1 200 OK
Date: Fri, 21 Jun 2019 10:55:30 GMT
Server: Apache
Vary: Accept-Encoding
Connection: close
Content-Type: text/html

value inserted
connecting to varmatrix.com
Requesting URL:/surendrardl/nitk/postone/postinsert.php?
HTTP/1.1 200 OK
Date: Fri, 21 Jun 2019 10:55:42 GMT
Server: Apache
Vary: Accept-Encoding
Connection: close

```

2. Enter `http://xyz.com/abc/def/postone/postportal.html` in your browser.



Temperature, Soil Moisture and Vibration

Temperature	35.93 °F
Soil Moisture	3831 %
Vibration	100 Hz

EXPERIMENT NO 15

FTP

Aim:

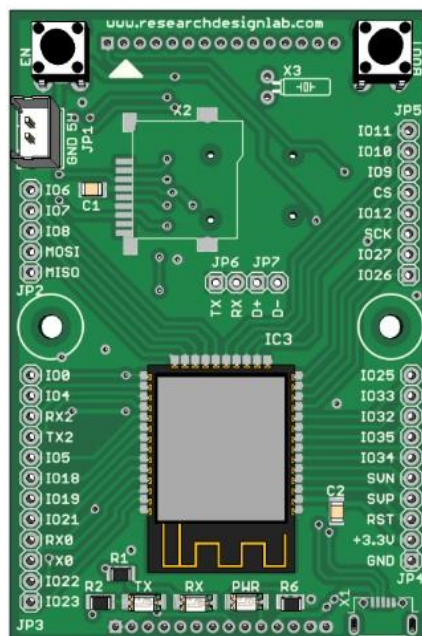
To interface wifi module and access the data using RDL ESP32 Development board.

Description:

Interfacing wifi module to access the data using ESP32 microcontroller.

Hardware required:

ESP32-Microcontroller Development board.

Pin connection:

Program:

```
#include <SD.h>
#include <SPI.h>
#include <WiFi.h>
/* comment out next line to write to SD from FTP server */
#define FTPWRITE
/* change to your network settings */
const char* ssid = "your userid";
const char* password = "your passwordl";
char server_link[] = "ftp://abcd.xyz.com/";
/* change to your server */
/* IPAddress server( 1, 2, 3, 4 ); */
WiFiClient client;
WiFiClient dclient;
char outBuf[128];
char outCount;
/* change fileName to your file (8.3 format!) */
char fileName[13] = "rdl.txt";
void setup()
{

Serial.begin(115200);

WiFi.mode(WIFI_STA);
WiFi.begin(ssid, password);

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

Serial.println("");
Serial.println("WiFi connected");
Serial.println("IP address: ");
Serial.println(WiFi.localIP());

Serial.println(F("Ready. Press f or r"));
}

void loop()
{
    byte inChar;

    inChar = Serial.read();

    if(inChar == 'f')
    {
        if(doFTP()) Serial.println(F("FTP OK"));
    }
}
```

```
    else Serial.println(F("FTP FAIL"));
  }
}
byte doFTP()
{
  Serial.println(F("SD opened"));

  if (client.connect(server_link,21)) {
    Serial.println(F("Command connected"));
  }
  else {
    // fh.close();
    Serial.println(F("Command connection failed"));
    return 0;
  }

  if(!eRcv()) return 0;

  client.println(F("USER xxxxxx"));

  if(!eRcv()) return 0;

  client.println(F("PASS xxxxxx"));

  if(!eRcv()) return 0;

  client.println(F("SYST"));

  if(!eRcv()) return 0;

  client.println(F("Type I"));

  if(!eRcv()) return 0;

  client.println(F("PASV"));

  if(!eRcv()) return 0;

  char *tStr = strtok(outBuf,(",");
  int array_pasv[6];
  for ( int i = 0; i < 6; i++) {
    tStr = strtok(NULL,(",");
    array_pasv[i] = atoi(tStr);
    if(tStr == NULL)
    {
      Serial.println(F("Bad PASV Answer"));
    }
  }
}
```

```
unsigned int hiPort,loPort;

hiPort = array_pasv[4] << 8;
loPort = array_pasv[5] & 255;

Serial.print(F("Data port: "));
hiPort = hiPort | loPort;
Serial.println(hiPort);

if (dclient.connect(server_link,hiPort)) {
    Serial.println(F("Data connected"));
}
else {
    Serial.println(F("Data connection failed"));
    client.stop();
    //fh.close();
    return 0;
}

#ifdef FTPWRITE
    client.print(F("STOR "));
    //client.print("aspiration/");
    client.println(fileName);
#else
    client.print(F("RETR "));
    client.println(fileName);
#endif

if(!eRcv())
{
    dclient.stop();
    return 0;
}

#ifdef FTPWRITE
    Serial.println(F("Writing"));

    byte clientBuf[]="I love my INDIA";
    int clientCount = 13;

    if(clientCount > 0) dclient.write(clientBuf,clientCount);

#else
    while(dclient.connected())
    {
        while(dclient.available())
        {
            char c = dclient.read();
            //fh.write(c);
        }
    }
}
```

```
    Serial.write(c);
  }
}
#endif

dclient.stop();
Serial.println(F("Data disconnected"));

if(!eRcv()) return 0;

client.println(F("QUIT"));

if(!eRcv()) return 0;

client.stop();
Serial.println(F("Command disconnected"));

// fh.close();
Serial.println(F("SD closed"));
return 1;
}

byte eRcv()
{
  byte respCode;
  byte thisByte;

  while(!client.available()) delay(1);

  respCode = client.peek();

  outCount = 0;

  while(client.available())
  {
    thisByte = client.read();
    Serial.write(thisByte);

    if(outCount < 127)
    {
      outBuf[outCount] = thisByte;
      outCount++;
      outBuf[outCount] = 0;
    }
  }

  if(respCode >= '4')
  {
    efail();
    return 0;
  }
}
```



```

    }

    return 1;
}

void efail()
{
    byte thisByte = 0;

    client.println(F("QUIT"));

    while(!client.available()) delay(1);

    while(client.available())
    {
        thisByte = client.read();
        Serial.write(thisByte);
    }

    client.stop();
    Serial.println(F("Command disconnected"));
    // fh.close();
    Serial.println(F("SD closed"));
}

```

Output:

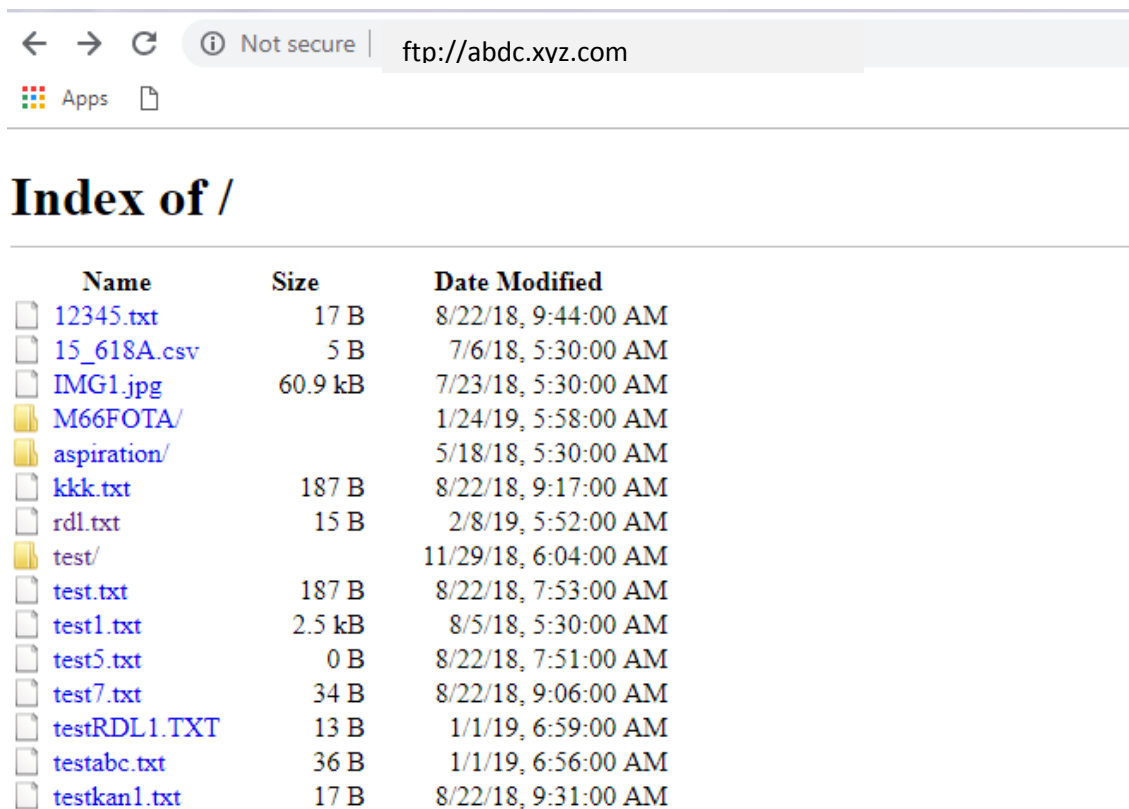
1. Gives information whether the WiFi is connected to the server.

```

COM72
load:0x3f1001c,len:928
ho 0 tail 12 room 4
load:0x40078000,len:9280
load:0x40080400,len:5848
entry 0x40080698
..
WiFi connected
IP address:
192.168.43.201
Ready. Press f or r
SD opened
Command connected
220----- Welcome to Pure-FTPd [privsep] [TLS] -----
220-You are user number 7 of 500 allowed.
220-Local time is now 00:22. Server port: 21.
220-This is a private system - No anonymous login
220 You will be disconnected after 3 minutes of inactivity.
331 User kanwall234 OK. Password required
230 OK. Current restricted directory is /
215 UNIX Type: L8
200 TYPE is now 8-bit binary
227 Entering Passive Mode (166,62,1,1,196,84)
Data port: 50260
Data connected
150 Accepted data connection
Writing
Data disconnected
226-File successfully transferred
226 0.191 seconds (measured here), 78.44 bytes per second
221-Goodbye. You uploaded 1 and downloaded 0 kbytes.
221 Logout.
Command disconnected
SD closed
FTP OK
Autoscroll

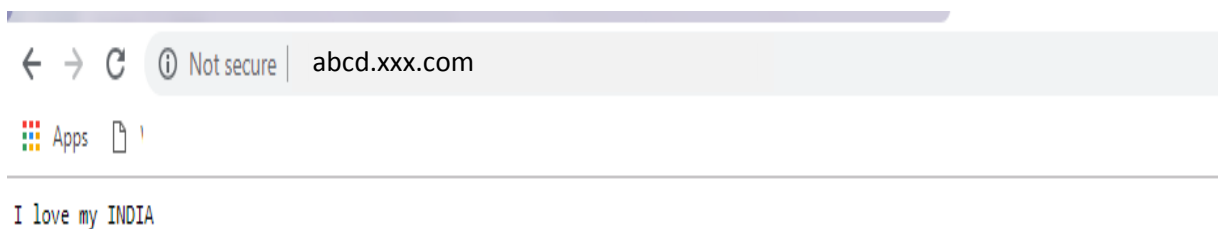
```

2. Here you can see that the contents are displayed in the server. We can also see that the folder created in the program being displayed.



Name	Size	Date Modified
12345.txt	17 B	8/22/18, 9:44:00 AM
15_618A.csv	5 B	7/6/18, 5:30:00 AM
IMG1.jpg	60.9 kB	7/23/18, 5:30:00 AM
M66FOTA/		1/24/19, 5:58:00 AM
aspiration/		5/18/18, 5:30:00 AM
kkk.txt	187 B	8/22/18, 9:17:00 AM
rd1.txt	15 B	2/8/19, 5:52:00 AM
test/		11/29/18, 6:04:00 AM
test.txt	187 B	8/22/18, 7:53:00 AM
test1.txt	2.5 kB	8/5/18, 5:30:00 AM
test5.txt	0 B	8/22/18, 7:51:00 AM
test7.txt	34 B	8/22/18, 9:06:00 AM
testRDL1.TXT	13 B	1/1/19, 6:59:00 AM
testabc.txt	36 B	1/1/19, 6:56:00 AM
testkan1.txt	17 B	8/22/18, 9:31:00 AM

3. The contents in the folder are given below.



EXPERIMENT NO 16

OTA (Over the air) programming

Aim:

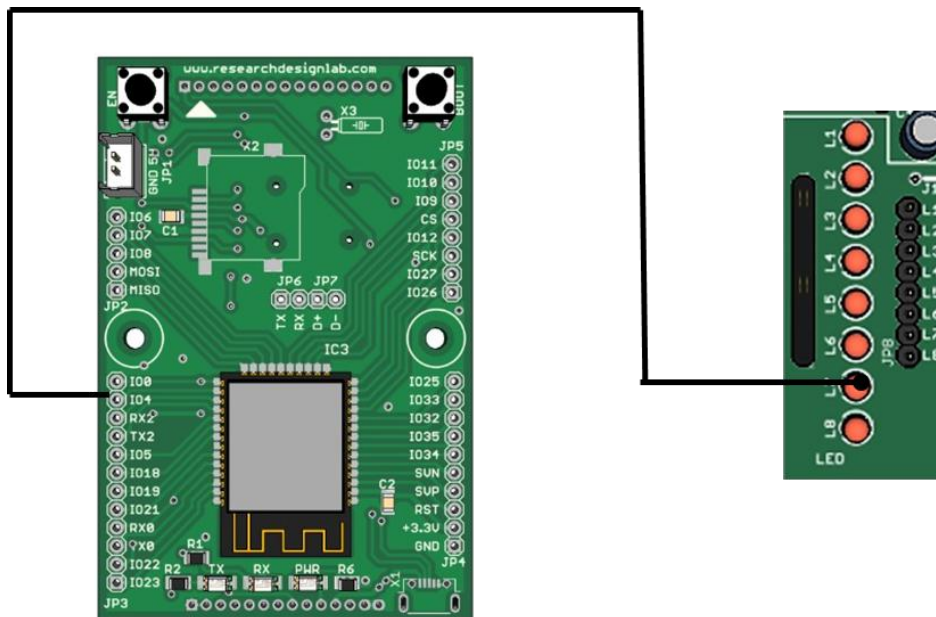
Turn ON and OFF an LED after Particular delay using OTA web server

Description:

To learn how to connect LED to digital pins of an ESP32 Microcontroller and program to blink an LED using OTA web server.

Hardware required:

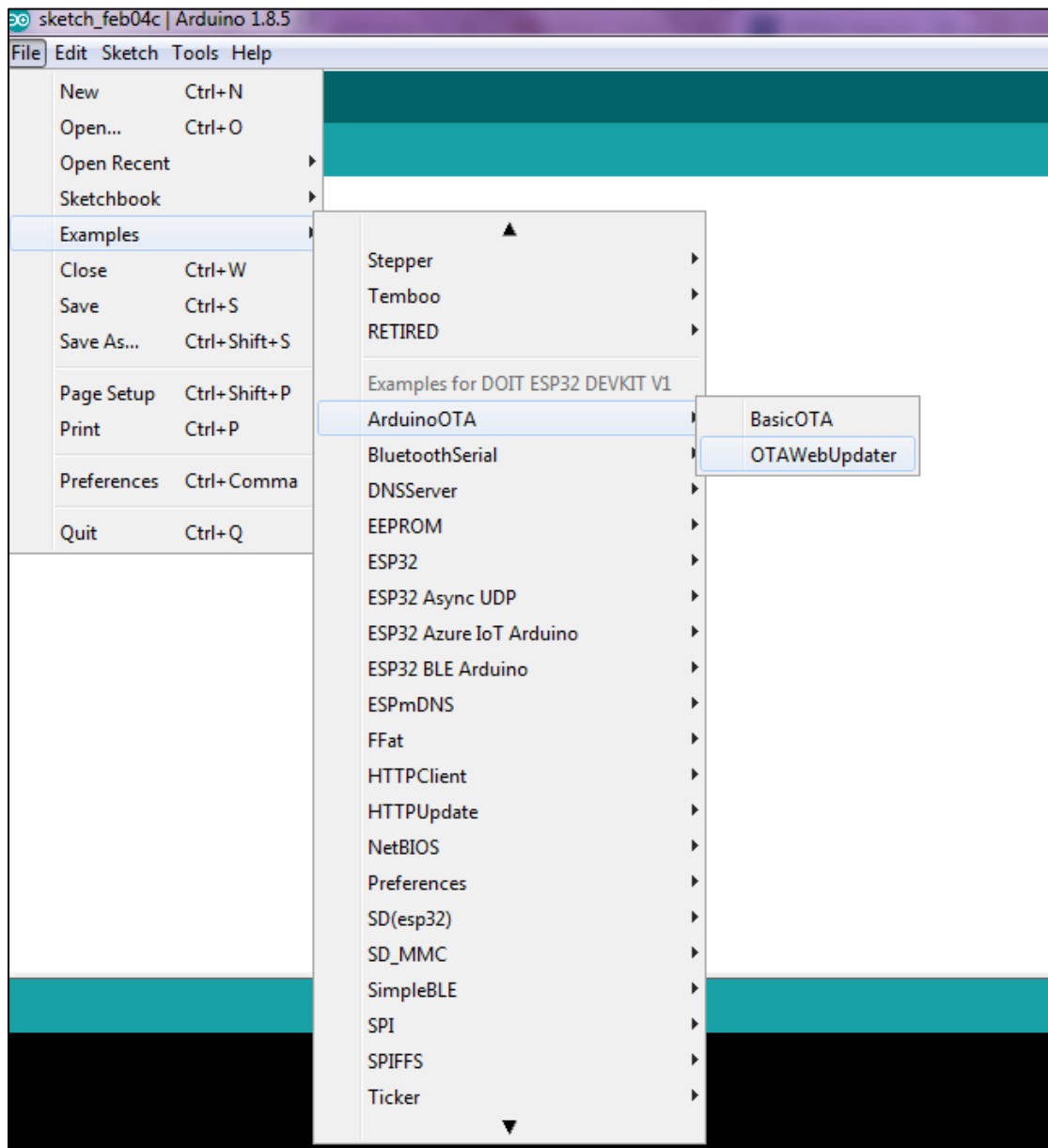
ESP32-Microcontroller development board.

Pin connection:

Pin Mapping:

LED	ESP32
LED	I04

Procedure:

1. When you install the ESP32 add-on for the Arduino IDE, it will automatically install the Arduino OTA library.
2. Go to **File > Examples > ArduinoOTA > OTAWebUpdater**.



3. The following code should load

```
#include <WiFi.h>
#include <WiFiClient.h>
#include <WebServer.h>
#include <ESPmDNS.h>
#include <Update.h>
```

```
const char* host = "esp32";
const char* ssid = "ssid";
const char* password = "password";
```

```
WebServer server(80);
```

```
/*
```

```
 * Login page
```

```
*/
```

```
const char* loginIndex =
"<form name='loginForm'>"
  "<table width='20%' bgcolor='A09F9F' align='center'>"
    "<tr>"
      "<td colspan=2>"
        "<center><font size=4><b>ESP32 Login Page</b></font></center>"
        "<br>"
      "</td>"
      "<br>"
      "<br>"
    "</tr>"
    "<td>Username:</td>"
    "<td><input type='text' size=25 name='userid'><br></td>"
  "</tr>"
  "<br>"
  "<br>"
  "<tr>"
    "<td>Password:</td>"
    "<td><input type='Password' size=25 name='pwd'><br></td>"
  "<br>"
  "<br>"
  "</tr>"
  "<tr>"
    "<td><input type='submit' onclick='check(this.form)' value='Login'></td>"
  "</tr>"
"</table>"
"</form>"
"<script>"
  "function check(form)"
  "{"
  "if(form.userid.value=='admin' && form.pwd.value=='admin')"
```

```
"{"
```

```
"window.open('/serverIndex')"
```

```
"}"
```

```
"else"
```

```
"{"
```

```
" alert('Error Password or Username')/*displays error message*/"
```

```
"}"
```

```
"}"
```

```

"</script>";

/*
 * Server Index Page
 */

const char* serverIndex =
"<script src='https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js'></script>"
"<form method='POST' action='#' enctype='multipart/form-data' id='upload_form'>"
  "<input type='file' name='update'>"
  "<input type='submit' value='Update'>"
"</form>"
"<div id='prg'>progress: 0%</div>"
"<script>"
"$('form').submit(function(e){ "
  "e.preventDefault();"
  "var form = $('#upload_form')[0];"
  "var data = new FormData(form);"
  "$.ajax({ "
    "url: '/update',"
    "type: 'POST',"
    "data: data,"
    "contentType: false,"
    "processData:false,"
    "xhr: function() { "
      "var xhr = new window.XMLHttpRequest();"
      "xhr.upload.addEventListener('progress', function(evt) { "
        "if (evt.lengthComputable) { "
          "var per = evt.loaded / evt.total;"
          "$('#prg').html('progress: ' + Math.round(per*100) + '%');"
        } "
      }, false);"
    }, "return xhr;"
  }, " "
  "success:function(d, s) { "
    "console.log('success!)"
  }, " "
  "error: function (a, b, c) { "
    " "
  });"
  });"
"</script>";

/*
 * setup function
 */
void setup(void) {
  Serial.begin(115200);

  // Connect to WiFi network

```

```
WiFi.begin(ssid, password);
Serial.println("");

// Wait for connection
while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
}
Serial.println("");
Serial.print("Connected to ");
Serial.println(ssid);
Serial.print("IP address: ");
Serial.println(WiFi.localIP());

/*use mdns for host name resolution*/
if (!MDNS.begin(host)) { //http://esp32.local
    Serial.println("Error setting up MDNS responder!");
    while (1) {
        delay(1000);
    }
}
Serial.println("mDNS responder started");
/*return index page which is stored in serverIndex */
server.on("/", HTTP_GET, []() {
    server.sendHeader("Connection", "close");
    server.send(200, "text/html", loginIndex);
});
server.on("/serverIndex", HTTP_GET, []() {
    server.sendHeader("Connection", "close");
    server.send(200, "text/html", serverIndex);
});
/*handling uploading firmware file */
server.on("/update", HTTP_POST, []() {
    server.sendHeader("Connection", "close");
    server.send(200, "text/plain", (Update.hasError()) ? "FAIL" : "OK");
    ESP.restart();
}, []() {
    HTTPUpload& upload = server.upload();
    if (upload.status == UPLOAD_FILE_START) {
        Serial.printf("Update: %s\n", upload.filename.c_str());
        if (!Update.begin(UPDATE_SIZE_UNKNOWN)) { //start with max available size
            Update.printError(Serial);
        }
    } else if (upload.status == UPLOAD_FILE_WRITE) {
        /* flashing firmware to ESP*/
        if (Update.write(upload.buf, upload.currentSize) != upload.currentSize) {
            Update.printError(Serial);
        }
    } else if (upload.status == UPLOAD_FILE_END) {
        if (Update.end(true)) { //true to set the size to the current progress
```

```

    Serial.printf("Update Success: %u\nRebooting...\n", upload.totalSize);
  } else {
    Update.printError(Serial);
  }
}
});
server.begin();
}

void loop(void) {
  server.handleClient();
  delay(1);
}

```

4. You should change the following lines on the code to include your own network credentials
 - `const char* ssid = " ";`
 - `const char* password = " ";`
5. Upload the above code to ESP32 board. Enter proper network credentials.
6. Now select proper board and serial port.
7. Once the code is uploaded , open serial monitor select baud rate 115200 and press enable button and then you will get ESP32 IP address.

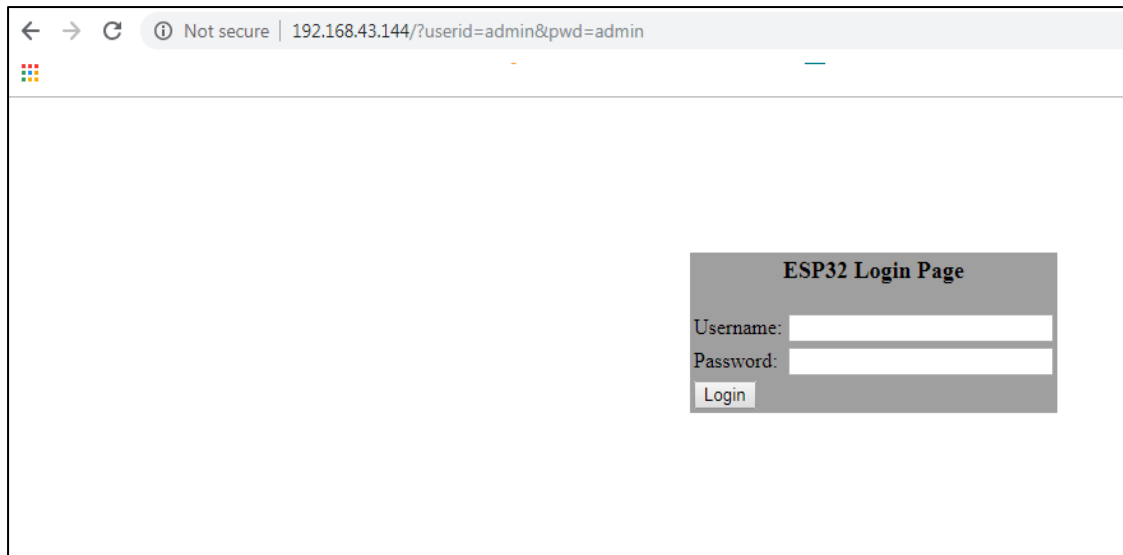
```

.....
Connected to J7 pro
IP address: 192.168.43.144
mDNS responder started
Update: LED_WEB_UPDATER.ino.doitESP32devkitV1.bin
Update Success: 782624
Rebooting...
ets Jun  8 2016 00:22:57

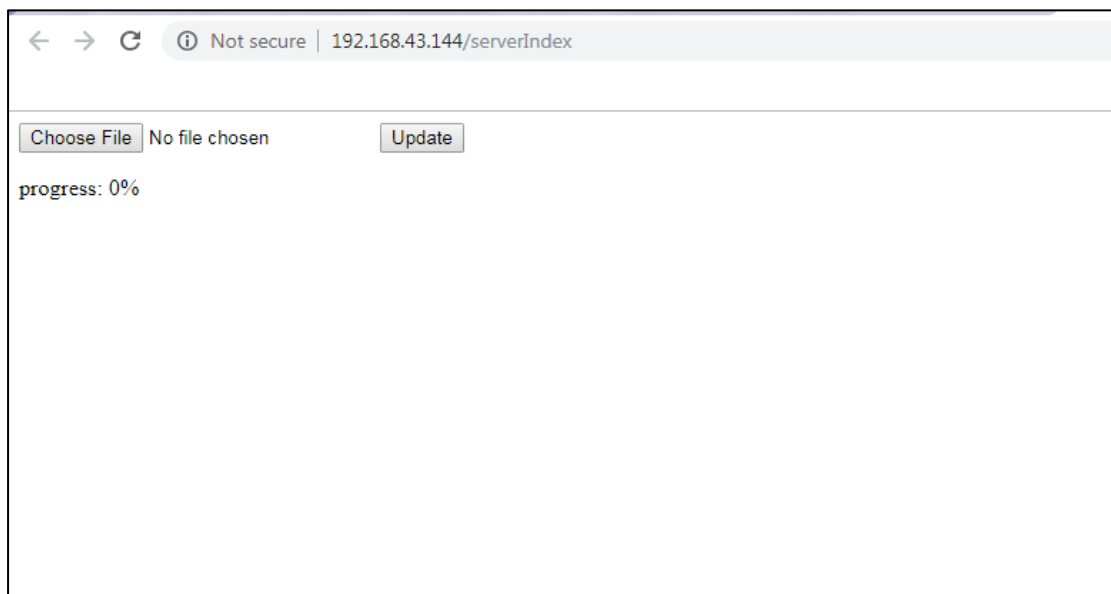
rst:0xc (SW_CPU_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:1
load:0x3fff0018,len:4
load:0x3fff001c,len:928
ho 0 tail 12 room 4
load:0x40078000,len:9280
load:0x40080400,len:5848
entry 0x40080698

```

8. Copy the IP address which you obtain from the serial monitor and paste it in your browser.

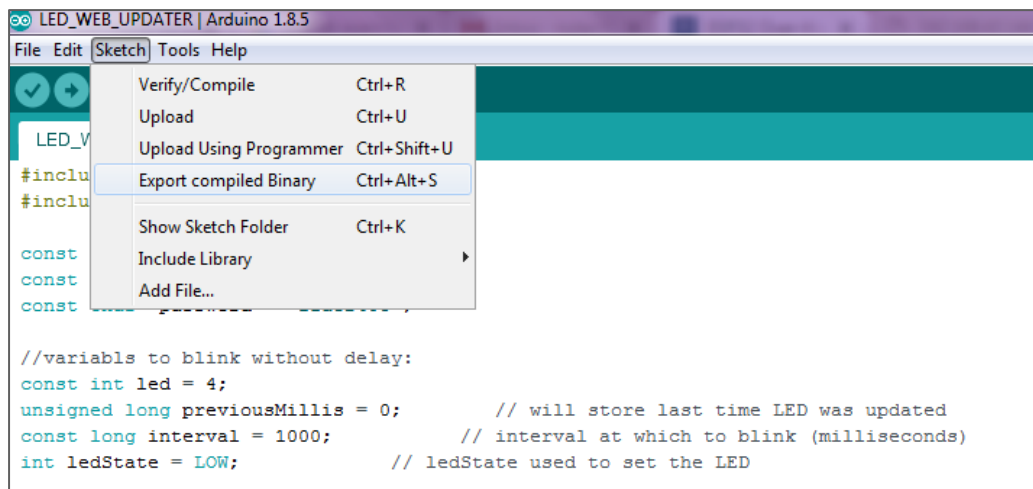


9. Username: admin
Password: admin
10. After entering the username and password a new tab should open on the `/serverIndex` URL. This page allows you to upload a new code to your ESP32. You should upload `.bin` files .

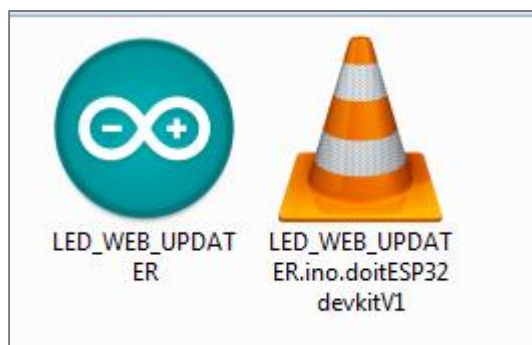


Preparing the sketch:

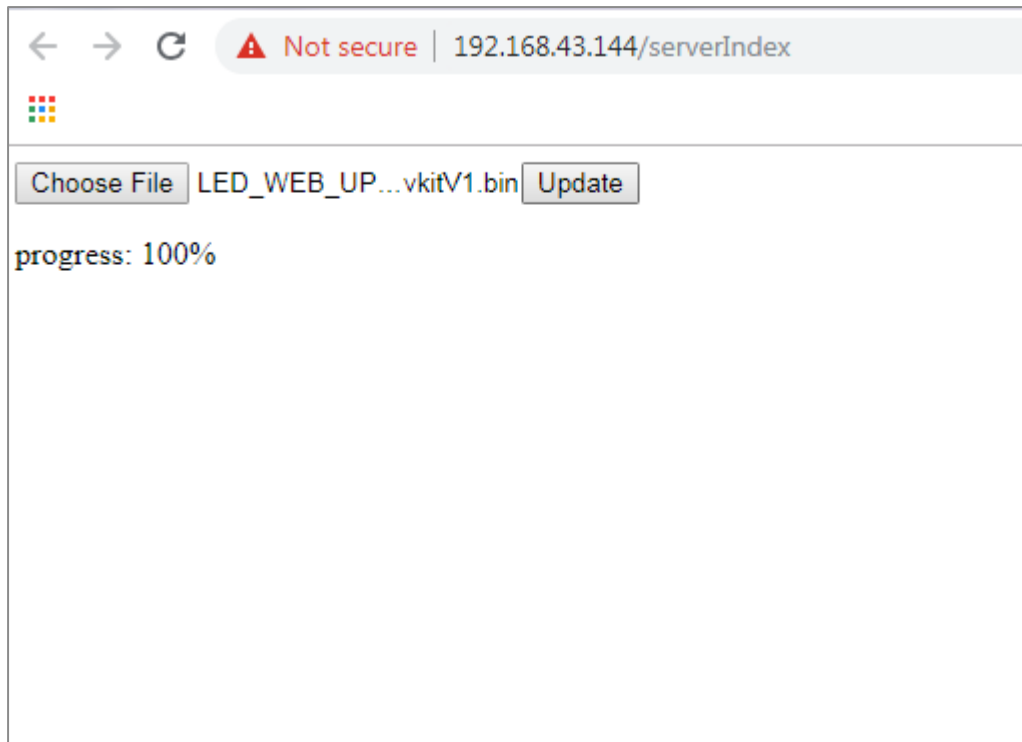
1. Write a simple program (Blinking of LED) using OTA web server and save it with the name `LED_blink` and compile it.
2. Once the uploading is done go to Sketch<Export compiled binary.



3. Then select Sketch<Show sketch folder.
4. In this folder two files will be generated .ino and .bin.
5. You should upload the .bin file using OTA Web Server.



6. In browser on ESP32 OTA Web Updater page , click on **choose file** button
7. Select .bin file and click **Update**.
8. Code will be successfully uploaded.



OUTPUT:

