ESP32 IoT DEVELOPMENT KIT

Introduction:

ESP32 IoT Development Trainer Kit is a single board IoT edge computing node equipped with on board IO's, communication interfaces & peripherals. It is really easy to design, experiment with, and test circuits without soldering. Students, hobbyists, enthusiasts, and professionals can explore a wide variety of IoT concepts simply by connecting FRC cable & integrating with the cloud platform.

Features:

MCU

- ESP32-D0WD-V3 embedded, Xtensa® dual-core 32-bit LX6 microprocessor, up to 240 MHz
- 448 KB ROM for booting and core functions
- 520 KB SRAM for data and instructions
- 16 KB SRAM in RTC
- 4 MB SPI flash

Wi-Fi

- 802.11b/g/n
- Bit rate: 802.11n up to 150 Mbps
- A-MPDU and A-MSDU aggregation
- 0.4 µs guard interval support
- Center frequency range of operating channel: 2412 ~ 2484 MH

Bluetooth® / BLE

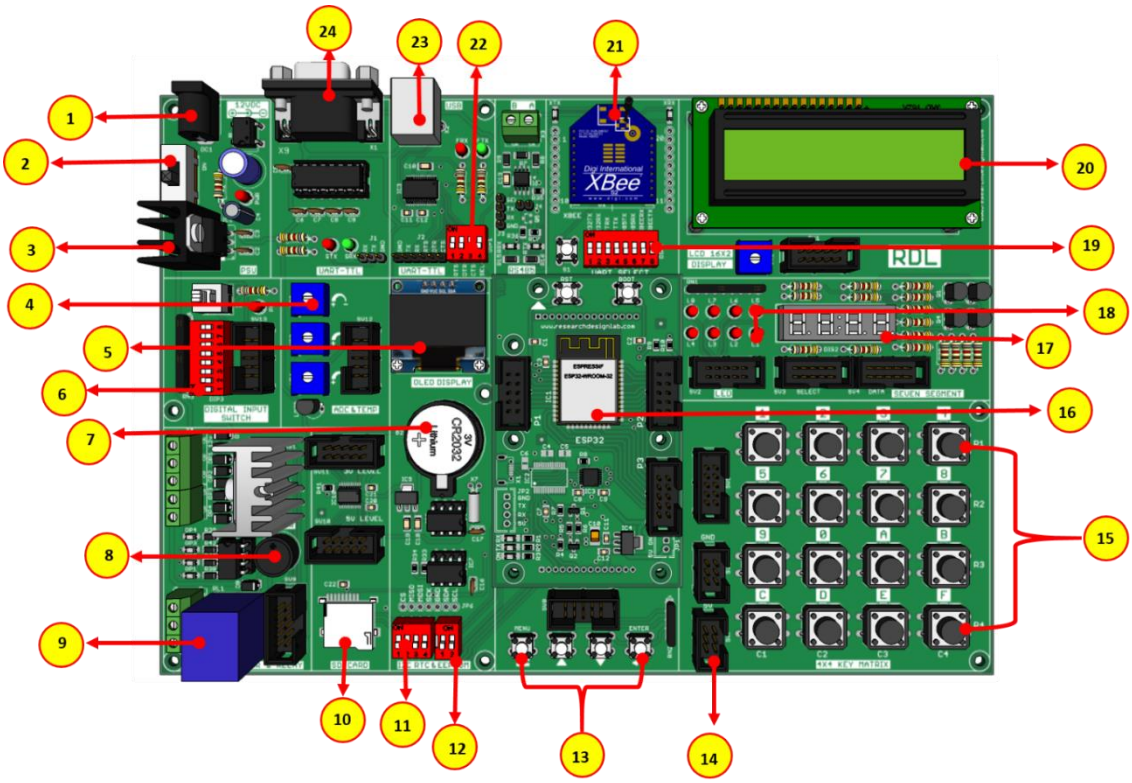
- Bluetooth V4.2 BR/EDR and Bluetooth LE specification
- Class-1, class-2 and class-3 transmitter
- AFH
- CVSD and SBC

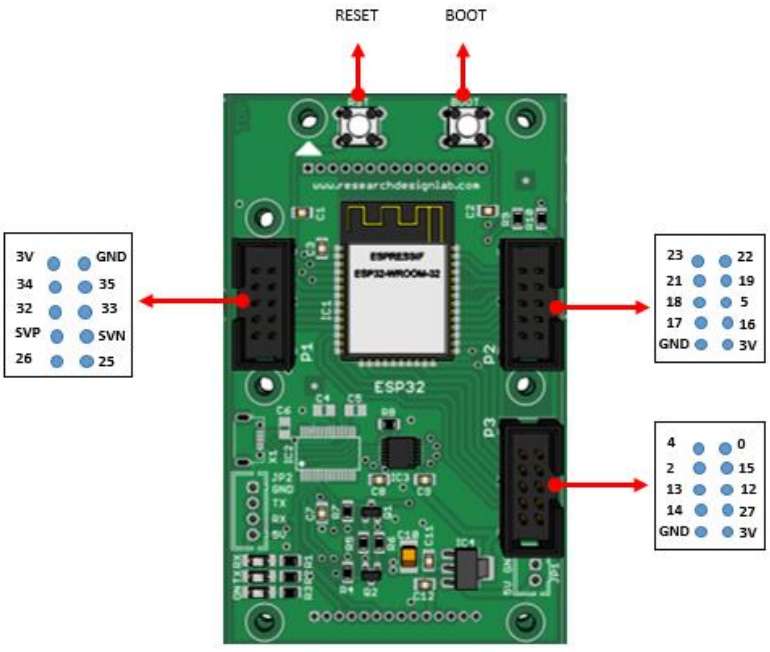
Hardware

- **Interfaces**: SD card, UART, SPI, SDIO, I2C, LED PWM, Motor PWM, I2S, IR, pulse counter, GPIO, capacitive touch sensor, ADC, DAC, Two-Wire Automotive Interface
- **Communication Interface**: RS232, RS485 (Modbus RTU) , USB , SPI , I2C .

- **On Board Peripheral** : OLED Display , 16x2 LCD Display, Seven Segment Display ,8x LED , 4x4 Hex Keypad , 1x4 Menu Keypad , Xbee Adapter, 3.3 to 5v Level Converter, SD CARD Interface, RTC & EEPROM, DC Motor / Stepper Motor Driver, Relay, Buzzer,1xTemperature Sensor,3x Analog Test POT ,8x Selection DIP Switch

ESP-32 Board Narration





1. Power Supply
2. Power ON Switch
3. Heat Sink
4. ADC (Variable Resistor POT)
5. OLED Display
6. Digital Input Switch
7. RTC Battery
8. Buzzer
9. Relay
10. SD Card Holder
11. Jumper Settings for I2C RTC
12. Jumper Settings for EEPROM
13. 1*4 Keypad Switches
14. RDL Bus FRC Connector
15. 4*4 Keypad Matrix
16. ESP32 Controller
17. 7 Segment Display
18. 2*4 LED's
19. Jumper Settings for UART Selection Pin
20. 16*2 LCD Display
21. XBee Adapter
22. Jumper Settings for UART TTL
23. USB Port
24. DB-9 Serial Female Connector

Applications

- Generic Low-power IoT Sensor Hub
- Generic Low-power IoT Data Loggers
- Cameras for Video Streaming
- Over-the-top (OTT) Devices
- Speech Recognition
- Image Recognition
- Mesh Network
- Home Automation
- Smart Building
- Industrial Automation
- Smart Agriculture
- Audio Applications
- Health Care Applications
- Wi-Fi-enabled Toys
- Wearable Electronics
- Retail & Catering Applications

Package Includes

- Development Board with Wooden Enclosure
- USB Cable
- 12V 2A Adapter
- FRC Cable

NOTE: XBee module is not included in the package

Contents

EXPERIMENT NO 1
BLINKING AN LED

Aim:

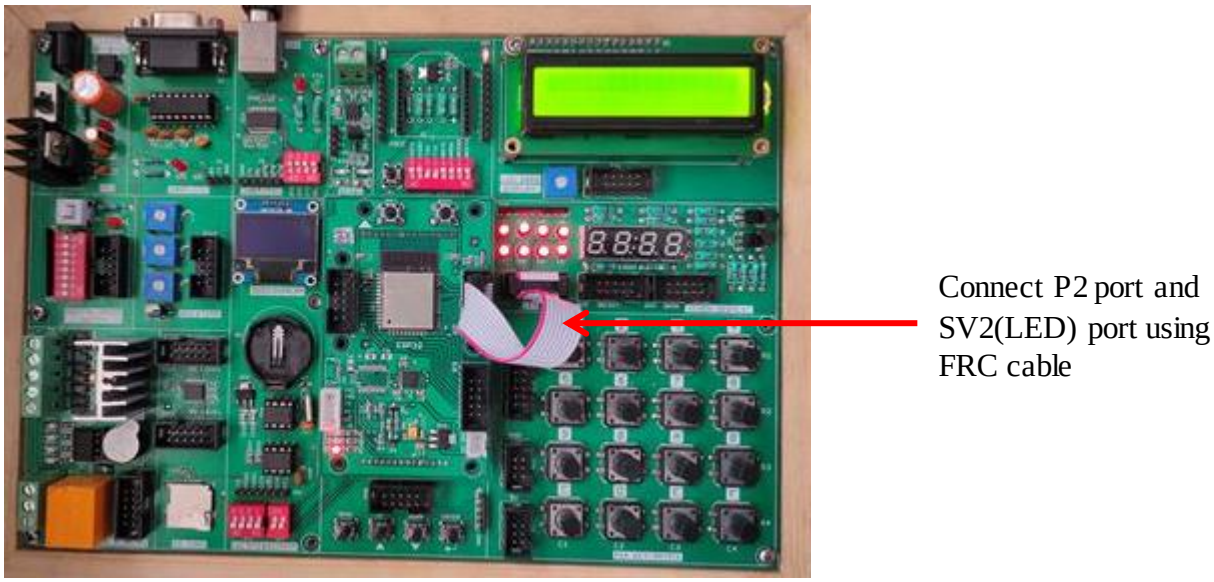
Turn ON and OFF an LED after Particular delay

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.



Procedure:

1. Connect P2 port and SV2(LED) port using FRC cable as shown above.
2. Connect the USB cable to the board.
3. Open Arduino IDE. Select DOIT ESP32 DEVKIT V1 in boards and select COM port.
4. Now verify the program and Upload it.
5. Now you can see the LED blink on the ESP32 development board.

Program:

```
const int L1=16, L2=17, L3=5, L4=18, L5=19, L6=21, L7=22, L8=23;  //initializing LED pins
void setup()
{
pinMode(L1, OUTPUT);      // Set all Port P2 pins as output
pinMode(L2, OUTPUT);
pinMode(L3, OUTPUT);
pinMode(L4, OUTPUT);
pinMode(L5, OUTPUT);
pinMode(L6, OUTPUT);
pinMode(L7, OUTPUT);
pinMode(L8, OUTPUT);
}
void loop()
{
digitalWrite(L1, HIGH);
digitalWrite(L2, HIGH);
digitalWrite(L3, HIGH);
digitalWrite(L4, HIGH);
digitalWrite(L5, HIGH);
digitalWrite(L6, HIGH);
digitalWrite(L7, HIGH);
digitalWrite(L8, HIGH);
delay(2000);
digitalWrite(L1, LOW);
digitalWrite(L2, LOW);
digitalWrite(L3, LOW);
digitalWrite(L4, LOW);
digitalWrite(L5, LOW);
digitalWrite(L6, LOW);
digitalWrite(L7, LOW);
digitalWrite(L8, LOW);
delay(2000);
}
```

EXPERIMENT NO 2

CONTROLLING LED USING SWITCH

Aim:

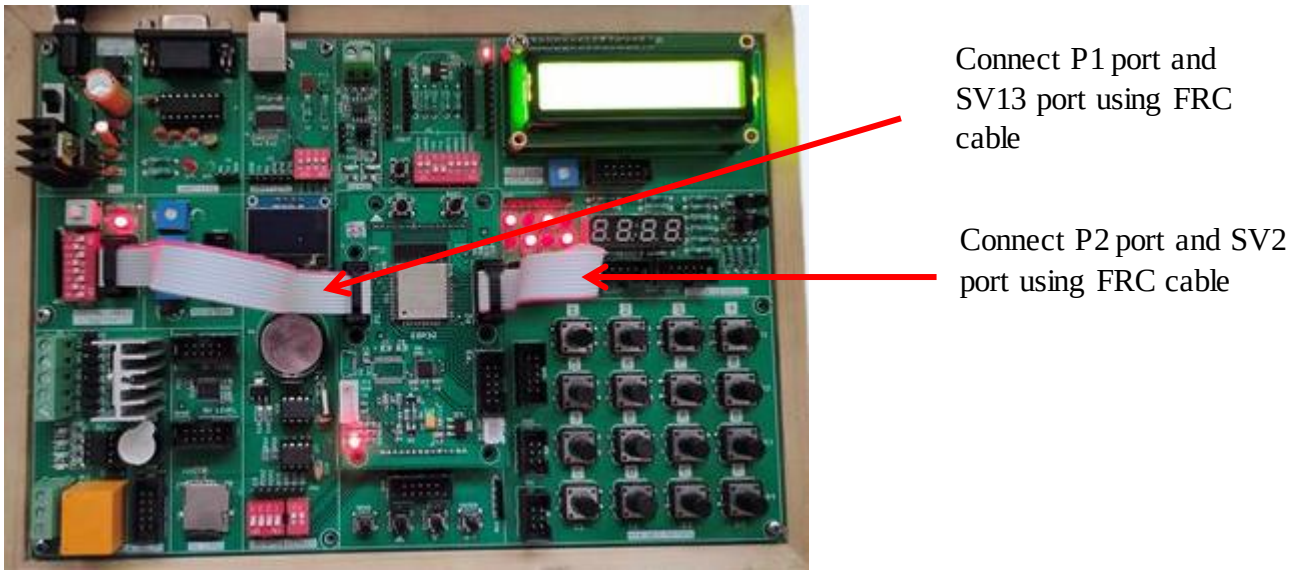
Controlling LED using a DIP Switch.

Description:

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

Hardware Requirement:

ESP32 Microcontroller Development board.



Procedure:

1. Connect P1 port and SV13(Digital Input Switch) port and connect P2 port and SV2(LED) port using FRC cable as shown above.
2. Connect the USB cable to the board.
3. Open Arduino IDE .Select DOIT ESP32 DEVKIT V1in boards and select COM port.
4. Now verify the program and Upload it.
5. Now you can see that when switch is ON LED starts to glow on the ESP32 development board.

Program:

```
const int Switch[8]={34, 35, 32, 33, 36, 39, 25, 26};  //declaring DIP Switches(Port P1)
const int Led[8]={16, 17, 5, 18, 19, 21, 22, 23};      //declaring LEDs (Port P2)

void setup()
{
  for(int i=0;i<8;i++)
  {
    pinMode(Switch[i],INPUT);
    pinMode(Led[i],OUTPUT);
    delay(20);
  }
}

void loop()
{
 for(int i=0; i<8;i++)
 {
  digitalWrite(Led[i],digitalRead(Switch[i]));    // Reads the state of each switches and replicate
it on LEDs
 }
delay(1000);
}
```

EXPERIMENT NO 3
SEVEN SEGMENT DISPLAYS

Aim:

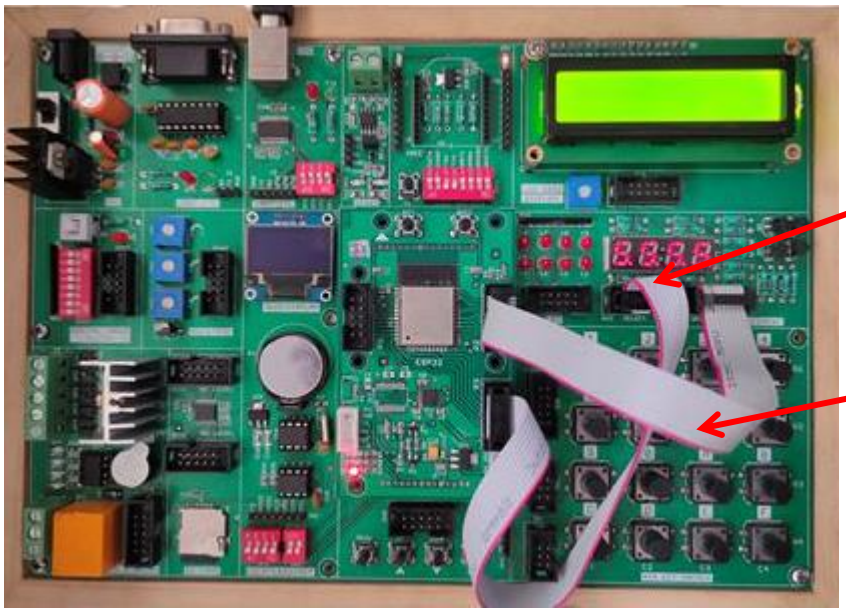
Interfacing ESP32-Microcontroller with seven segment display and to display the numbers.

Description:

To display numbers in the seven segment.

Hardware Required:

ESP32-Microcontroller Development board.



Procedure:

1. Connect P2 port and SV4(Data) port and connect P3 port and SV3(Select) port using FRC cable as shown above
2. Connect the USB cable to the board.
3. Open Arduino IDE .Select DOIT ESP32 DEVKIT V1 in boards and select COM port.
4. Now verify the program and Upload it.
5. Now you can see that number starts displaying on the seven segments on the ESP32 development board.

Program:

```
const int sel1=27, sel2=14, sel3=12, sel4=13;                    //initializing selection pins -Port P3
const int a=23 ,b=22, c=21, d=19, e=18, f=5, g=17, dp=16;  //initializing data pins -Port P2

void setup()
{
pinMode(sel1,OUTPUT);              //declaring Selection Pins as output
pinMode(sel2,OUTPUT);
pinMode(sel3,OUTPUT);
pinMode(sel4,OUTPUT);

 digitalWrite(sel1,LOW);           //selecting all 4 digits of 7-Segment display by making it LOW
 digitalWrite(sel2,LOW);
 digitalWrite(sel3,LOW);
 digitalWrite(sel4,LOW);

pinMode(a,OUTPUT);                 //declaring data pins as output
pinMode(b,OUTPUT);
pinMode(c,OUTPUT);
pinMode(d,OUTPUT);
pinMode(e,OUTPUT);
pinMode(f,OUTPUT);
pinMode(g,OUTPUT);
pinMode(dp,OUTPUT);
delay(100);
}

void loop()
{
  // print 0
  digitalWrite(a,LOW);
  digitalWrite(b,LOW);
  digitalWrite(c,LOW);
  digitalWrite(d,LOW);
  digitalWrite(e,LOW);
  digitalWrite(f,LOW);
  digitalWrite(g,HIGH);
  digitalWrite(dp,LOW);
  delay(2000);
 // print 1
  digitalWrite(a,HIGH);
  digitalWrite(b,LOW);
  digitalWrite(c,LOW);
  digitalWrite(d,HIGH);
  digitalWrite(e,HIGH);
  digitalWrite(f,HIGH);
  digitalWrite(g,HIGH);
  digitalWrite(dp,HIGH);
  delay(2000);
```

```
// print 2
 digitalWrite(a,LOW);
 digitalWrite(b,LOW);
 digitalWrite(c,HIGH);
 digitalWrite(d,LOW);
 digitalWrite(e,LOW);
 digitalWrite(f,HIGH);
 digitalWrite(g,LOW);
 digitalWrite(dp,LOW);
 delay(2000);

// print 3
 digitalWrite(a,LOW);
 digitalWrite(b,LOW);
 digitalWrite(c,LOW);
 digitalWrite(d,LOW);
 digitalWrite(e,HIGH);
 digitalWrite(f,HIGH);
 digitalWrite(g,LOW);
 digitalWrite(dp,LOW);
 delay(2000);

// print 4
 digitalWrite(a,HIGH);
 digitalWrite(b,LOW);
 digitalWrite(c,LOW);
 digitalWrite(d,HIGH);
 digitalWrite(e,HIGH);
 digitalWrite(f,LOW);
 digitalWrite(g,LOW);
 digitalWrite(dp,LOW);
 delay(2000);

// print 5
 digitalWrite(a,LOW);
 digitalWrite(b,HIGH);
 digitalWrite(c,LOW);
 digitalWrite(d,LOW);
 digitalWrite(e,HIGH);
 digitalWrite(f,LOW);
 digitalWrite(g,LOW);
 digitalWrite(dp,LOW);
 delay(2000);

// print 6
 digitalWrite(a,LOW);
 digitalWrite(b,HIGH);
 digitalWrite(c,LOW);
 digitalWrite(d,LOW);
```

```
 digitalWrite(e,LOW);
 digitalWrite(f,LOW);
 digitalWrite(g,LOW);
 digitalWrite(dp,LOW);
 delay(2000);

// print 7
 digitalWrite(a,LOW);
 digitalWrite(b,LOW);
 digitalWrite(c,LOW);
 digitalWrite(d,HIGH);
 digitalWrite(e,HIGH);
 digitalWrite(f,HIGH);
 digitalWrite(g,HIGH);
 digitalWrite(dp,HIGH);
 delay(2000);

// print 8
 digitalWrite(a,LOW);
 digitalWrite(b,LOW);
 digitalWrite(c,LOW);
 digitalWrite(d,LOW);
 digitalWrite(e,LOW);
 digitalWrite(f,LOW);
 digitalWrite(g,LOW);
 digitalWrite(dp,LOW);
 delay(2000);

 // print 9
 digitalWrite(a,LOW);
 digitalWrite(b,LOW);
 digitalWrite(c,LOW);
 digitalWrite(d,LOW);
 digitalWrite(e,HIGH);
 digitalWrite(f,LOW);
 digitalWrite(g,LOW);
 digitalWrite(dp,LOW);
 delay(2000);

}
```

EXPERIMENT NO 4
HEXKEYPAD

Aim:

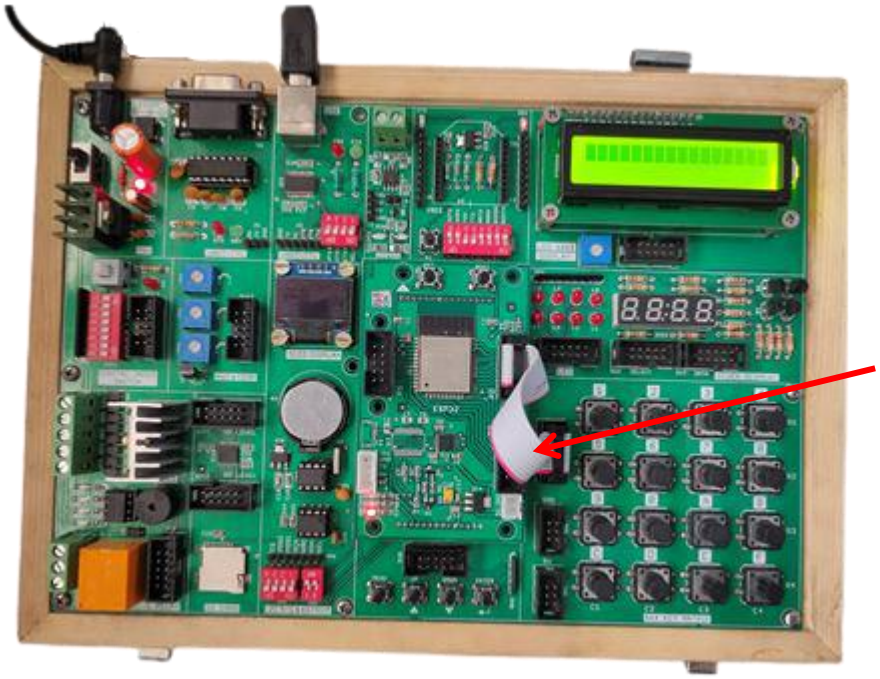
To interface 4x4 Hex Keypad with ESP32-Microcontroller module and display the pressed numbers on the Serial monitor.

Description:

To display the pressed key on the serial monitor.

Hardware Required:

ESP32-Microcontroller Development board



Connect P2 port and SV5 port using FRC cable

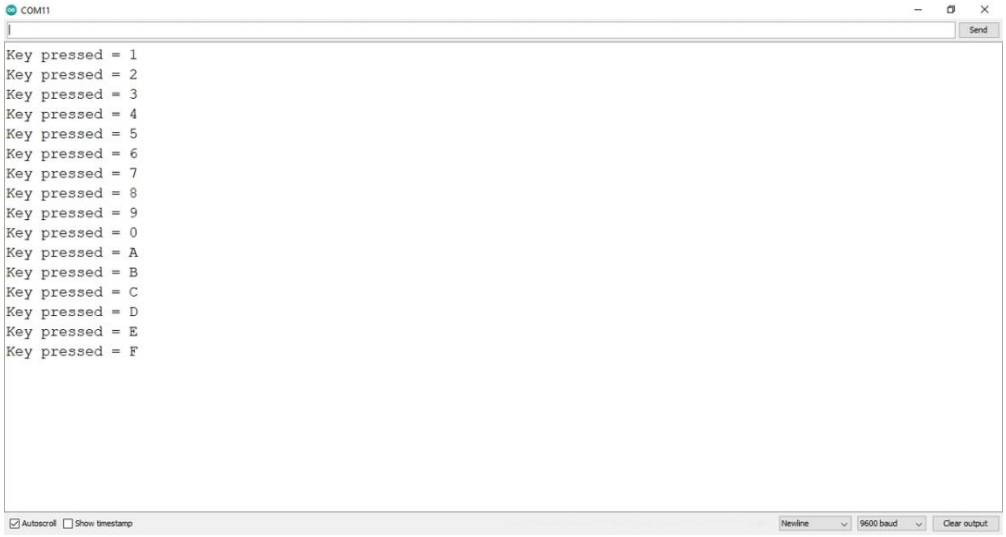
Procedure:

1. Connect P2 port and SV5(4*4 Key Matrix) port using FRC cable as shown above.
2. Connect the USB cable to the board.
3. Open Arduino IDE .Select DOIT ESP32 DEVKIT V1in boards and select COM port.
4. Now verify the program and Upload it.
5. After uploading is done open serial monitor to observe the output.
6. For a particular switch the same number displays on our serial monitor.

Program:

```
#include <Keypad.h>
char keys[4][4]={                //defining characters of 4X4 Key Matrix
 {'1','2','3','4'},
 {'5','6','7','8'},
 {'9','0','A','B'},
 {'C','D','E','F'}};

byte rowPin[4]={16,17,5,18};        //declaring the rows and column pins (Port P2)
byte colPin[4]={19,21,22,23};

Keypad keypad=Keypad(makeKeymap(keys),rowPin,colPin,4,4);   // Creating 4X4 Keypad
Matrix

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

void loop()
{
 char pressed=keypad.getKey();          //This function will fetch the character being pressed
 if(pressed)
 {
 Serial.print("Key pressed = ");
 Serial.println(pressed);
 }
 delay(500);
}
```

Output:


COM11
Send
Key pressed = 1
Key pressed = 2
Key pressed = 3
Key pressed = 4
Key pressed = 5
Key pressed = 6
Key pressed = 7
Key pressed = 8
Key pressed = 9
Key pressed = 0
Key pressed = A
Key pressed = B
Key pressed = C
Key pressed = D
Key pressed = E
Key pressed = F
Autoscroll
Show timestamp
Newline
9600 baud
Clear output

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.

Connect P2 port and SV1 port using FRC cable

Procedure:

1. Connect P2 port and SV1(LCD 16*2 Display) port using FRC cable as shown above
2. Connect the USB cable to the board.
3. Open Arduino IDE .Select DOIT ESP32 DEVKIT V1in boards and select COM port.
4. Verify the program and Upload it.
5. Now you can see the output displaying the message on LCD of ESP32 microcontroller board.

Program:

```
#include<LiquidCrystal.h>                    // Include the LCD library
LiquidCrystal lcd(21,19,18,5,17,16);         //Port P2 Mapping the pins with library
void setup()
{
Serial.begin(9600);                          //Baud Rate
lcd.begin(16,2);                             //initializing 16X2 LCD display
}
void loop()
{
lcd.setCursor(0,0);                          //first line in display
lcd.print("*WELCOME  TO RDL*");
delay(3000);
//lcd.clear();
lcd.setCursor(0,1);                          //second line in display
lcd.print("LEARNING  IS FUN");
delay(3000);
lcd.clear();
}
```

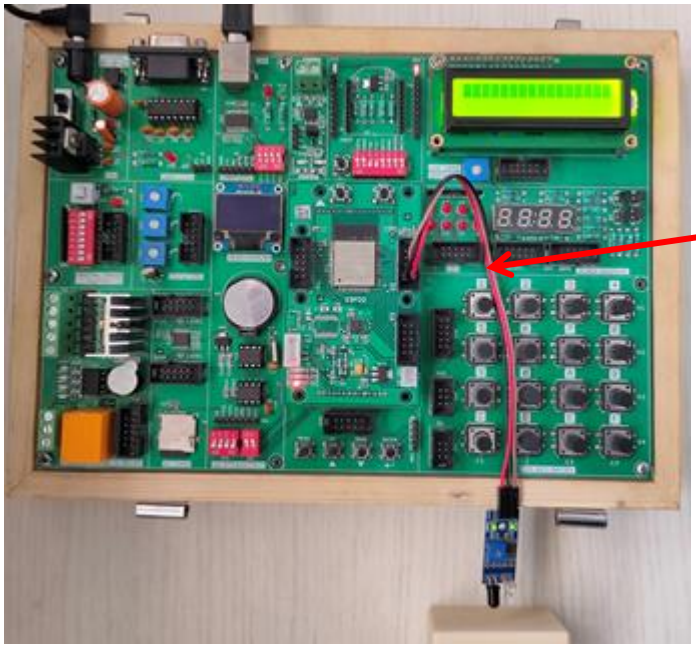
EXPERIMENT NO 6

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



Connect pin (P2 port) (IR Sensor)

5	-	OUT
GND	-	GND
3V	-	5V

Procedure:

1. Connect P2 port pins (5, GND, 3V) to IR Sensor pins (OUT, GND, 5V) using patch chords as shown above.
2. Connect the USB cable to the board.
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 the output on the serial monitor.

Program:

```
const int proximity=5;     //pin5 of port P1 connected to IR sensor
int value=0;

void setup() {
  Serial.begin(9600);
  pinMode(proximity, INPUT); //declared as input
  delay(100);
}

void loop() {
  value=digitalRead(proximity);  // storing sensor data in a variable.
  delay(1000);
  if(!value)                  //check for an obstacle if present.
  {
    Serial.println("obstacle detected.."); //display this message when obstacle detects
  }
}
```

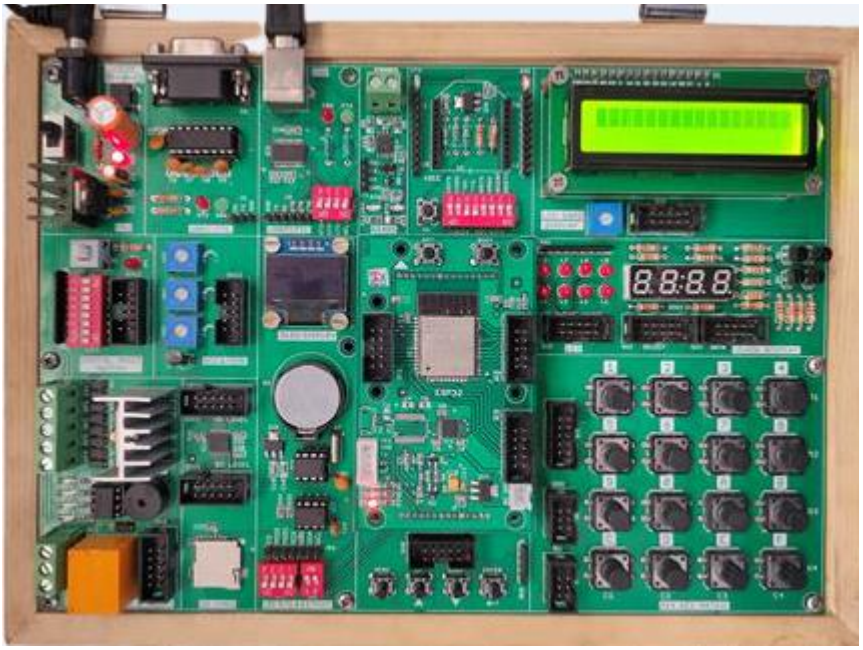
Output:

EXPERIMENT NO 6
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



Procedure:

1. Connect the USB cable to the board.
2. Open Arduino IDE .Select DOIT ESP32 DEVKIT V1in boards and select COM port.
3. Verify the program and Upload it.
4. Open the serial monitor to observe the output.

Program:

```
#include "Wire.h"
#define DS1307_I2C_ADDRESS 0x68

  // Convert normal decimal numbers to binary coded decimal
  byte decToBcd(byte val){
  return( (val/10*16)  + (val%10) );
}
  // Convert binary coded decimal to normal decimal numbers
  byte bcdToDec(byte val){
  return( (val/16*10)  + (val%16) );
}

void setup(){
  Wire.begin();
  Serial.begin(9600);
  delay(1000);
                              // set the initial  time here:
  setDS1307time(00,50,12,2,22,3,21);   // DS1307 seconds, minutes, hours, day, date, month,
year
    delay(1000);
}
void setDS1307time(byte  second, byte minute, byte hour, byte dayOfWeek, byte
dayOfMonth, byte month, byte year){
  // sets time and date data to DS1307
  Wire.beginTransmission(DS1307_I2C_ADDRESS);
  Wire.write(0); // set next input to start at the seconds register
  Wire.write(decToBcd(second)); // set seconds
  Wire.write(decToBcd(minute)); // set minutes
  Wire.write(decToBcd(hour)); // set hours
  Wire.write(decToBcd(dayOfWeek)); // set day of week (1=Sunday, 7=Saturday)
  Wire.write(decToBcd(dayOfMonth)); // set date (1 to 31)
  Wire.write(decToBcd(month)); // set month
  Wire.write(decToBcd(year)); // set year (0 to 99)
  Wire.endTransmission();
}
void readDS1307time(byte  *second, byte *minute, byte *hour,
byte *dayOfWeek, byte *dayOfMonth, byte *month, byte *year)
{
  Wire.beginTransmission(DS1307_I2C_ADDRESS);
  Wire.write(0); // set DS1307 register pointer to 00h
  Wire.endTransmission();
  delay(100);
```

```
 Wire.requestFrom(DS1307_I2C_ADDRESS, 7);
 // request seven bytes of data from DS1307 starting from register 00h
 *second = bcdToDec(Wire.read() & 0x7f);
 *minute = bcdToDec(Wire.read());
 *hour = bcdToDec(Wire.read() & 0x3f);
 *dayOfWeek = bcdToDec(Wire.read());
 *dayOfMonth = bcdToDec(Wire.read());
 *month = bcdToDec(Wire.read());
 *year = bcdToDec(Wire.read());
  Wire.endTransmission();
}
void displayTime(){
 byte second, minute, hour, dayOfWeek, dayOfMonth, month, year;
 // retrieve data from DS1307
 readDS1307time(&second, &minute, &hour, &dayOfWeek, &dayOfMonth, &month,
 &year);
 // send it to the serial monitor
 Serial.print(hour, DEC);
 // convert the byte variable to a decimal number when displayed
 Serial.print(":");
 if (minute<10){
  Serial.print("0");
 }
 Serial.print(minute, DEC);
 Serial.print(":");
 if (second<10){
  Serial.print("0");
 }
 Serial.print(second, DEC);
 Serial.print(" ");
 Serial.print(dayOfMonth, DEC);
 Serial.print("/");
 Serial.print(month, DEC);
 Serial.print("/");
 Serial.print(year, DEC);
 Serial.print(" Day of week: ");
 switch(dayOfWeek){
 case 1:
  Serial.println("Sunday");
  break;
 case 2:
  Serial.println("Monday");
  break;
 case 3:
```

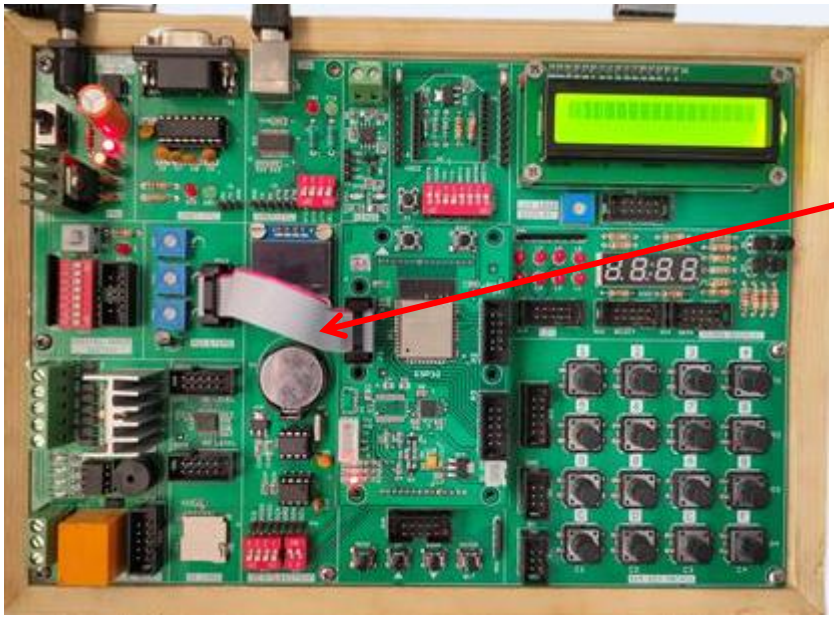
```
    Serial.println("Tuesday");
    break;
  case 4:
    Serial.println("Wednesday");
    break;
  case 5:
    Serial.println("Thursday");
    break;
  case 6:
    Serial.println("Friday");
    break;
  case 7:
    Serial.println("Saturday");
    break;
  }
}
void loop(){

  displayTime();  // display  the real-time  clock data on the Serial Monitor,
  delay(1000);  // every second
}
```

EXPERIMENT NO 7
ADC

Aim:
To display ADC value on the serial monitor using ESP 32 microcontroller development board.

Description:
To learn how to read ADC values using ESP32-Microcontroller.

Hardware required:
ESP32-Microcontroller Development board



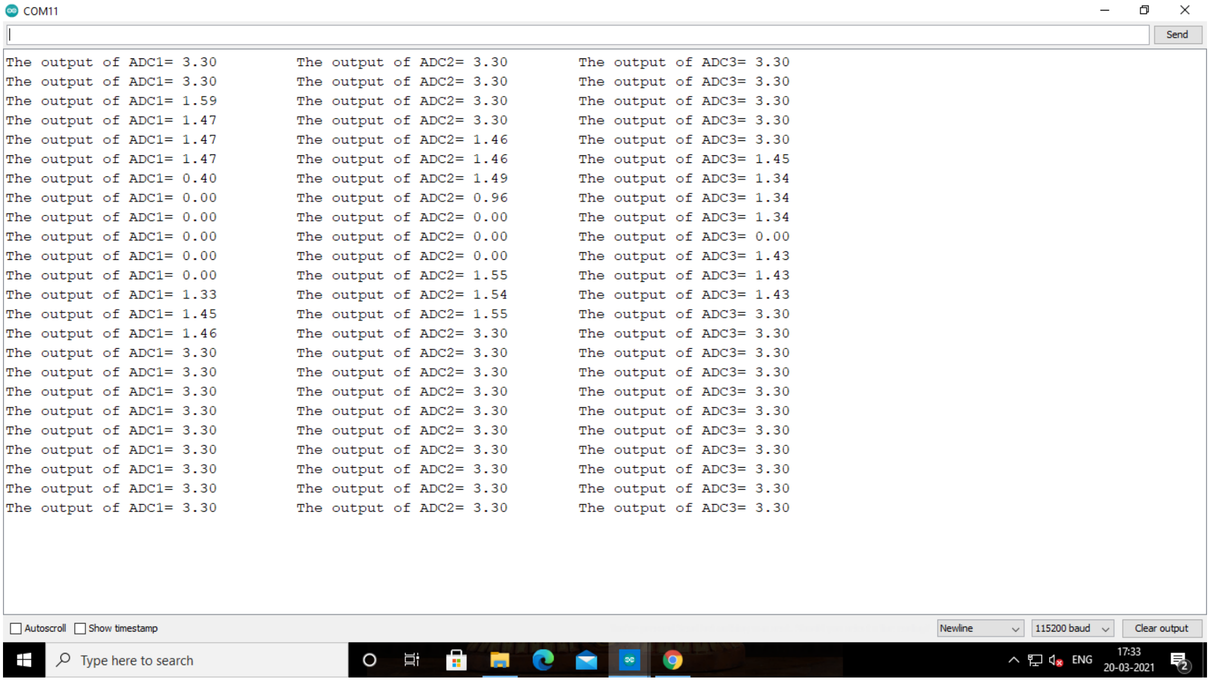
Procedure:

1. Connect P1 port and SV12(ADC & Temp) port using FRC cable as shown above
2. Connect the USB cable to the board.
3. Open Arduino IDE .Select DOIT ESP32 DEVKIT V1in boards and select COM port.
4. Verify the program and Upload it.

Program:

```
const int adc1=34;
const int adc2=35;
const int adc3=32;
int value1=0, value2=0, value3=0;
float res1=0, res2=0, res3=0;

void setup() {
  Serial.begin(115200);
  pinMode(adc1, INPUT);    // Pins Port1 Connected to ADC Knobs
  pinMode(adc2, INPUT);
  pinMode(adc3, INPUT);
  delay(500);

}

void loop() {
  delay(1000);
  value1=analogRead(adc1);
  value2=analogRead(adc2);
  value3=analogRead(adc3);

  res1=float((value1*3.3)/4095);    //3.3v is maximum voltage applied as a input
  res2=float((value2*3.3)/4095);    //it is 12bit ADC hence dividing by 4095 gives actual
voltage
  res3=float((value3*3.3)/4095);

 Serial.print("The output of ADC1= ");
 Serial.print(res1);
 delay(500);

 Serial.print("\t The output of ADC2= ");
 Serial.print(res2);
 delay(500);

 Serial.print("\t The output of ADC3= ");
 Serial.println(res3);
 delay(500);
}
```

Output:



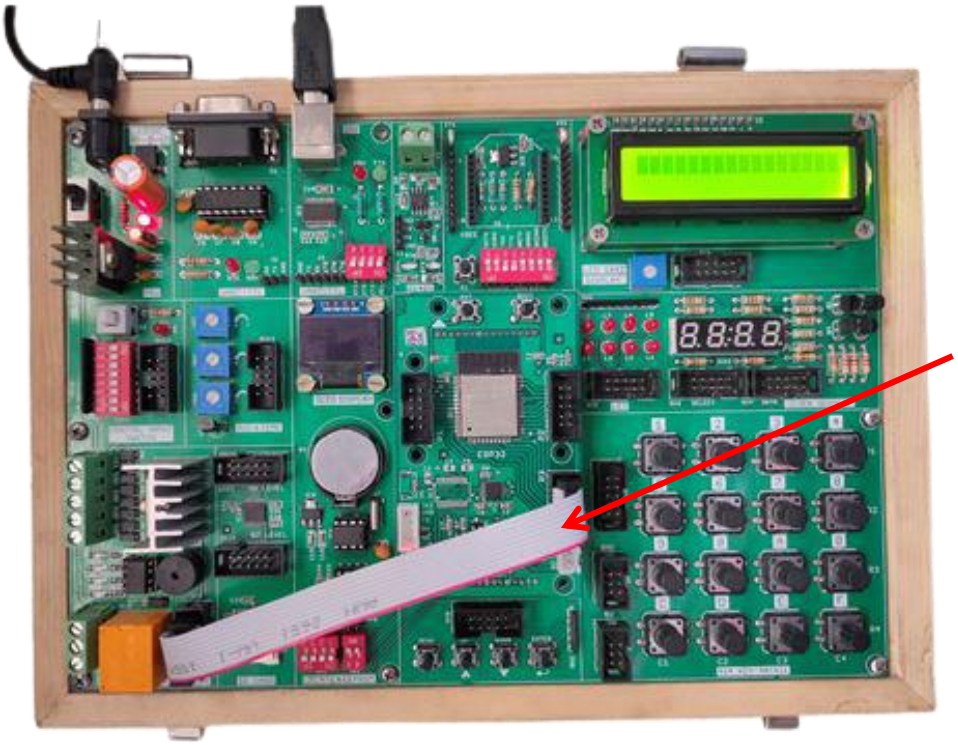

```
The output of ADC1= 3.30        The output of ADC2= 3.30        The output of ADC3= 3.30
The output of ADC1= 3.30        The output of ADC2= 3.30        The output of ADC3= 3.30
The output of ADC1= 1.59        The output of ADC2= 3.30        The output of ADC3= 3.30
The output of ADC1= 1.47        The output of ADC2= 3.30        The output of ADC3= 3.30
The output of ADC1= 1.47        The output of ADC2= 1.46        The output of ADC3= 3.30
The output of ADC1= 1.47        The output of ADC2= 1.46        The output of ADC3= 1.45
The output of ADC1= 0.40        The output of ADC2= 1.49        The output of ADC3= 1.34
The output of ADC1= 0.00        The output of ADC2= 0.96        The output of ADC3= 1.34
The output of ADC1= 0.00        The output of ADC2= 0.00        The output of ADC3= 1.34
The output of ADC1= 0.00        The output of ADC2= 0.00        The output of ADC3= 0.00
The output of ADC1= 0.00        The output of ADC2= 0.00        The output of ADC3= 1.43
The output of ADC1= 0.00        The output of ADC2= 1.55        The output of ADC3= 1.43
The output of ADC1= 1.33        The output of ADC2= 1.54        The output of ADC3= 1.43
The output of ADC1= 1.45        The output of ADC2= 1.55        The output of ADC3= 3.30
The output of ADC1= 1.46        The output of ADC2= 3.30        The output of ADC3= 3.30
The output of ADC1= 3.30        The output of ADC2= 3.30        The output of ADC3= 3.30
The output of ADC1= 3.30        The output of ADC2= 3.30        The output of ADC3= 3.30
The output of ADC1= 3.30        The output of ADC2= 3.30        The output of ADC3= 3.30
The output of ADC1= 3.30        The output of ADC2= 3.30        The output of ADC3= 3.30
The output of ADC1= 3.30        The output of ADC2= 3.30        The output of ADC3= 3.30
The output of ADC1= 3.30        The output of ADC2= 3.30        The output of ADC3= 3.30
The output of ADC1= 3.30        The output of ADC2= 3.30        The output of ADC3= 3.30
The output of ADC1= 3.30        The output of ADC2= 3.30        The output of ADC3= 3.30
The output of ADC1= 3.30        The output of ADC2= 3.30        The output of ADC3= 3.30
```

EXPERIMENT NO 8
L298 Motor

Aim:
This experiment shows how to rotate the L298 Motor clockwise and anticlockwise using ESP32-Microcontroller.

Description:
To learn how to rotate the L298 Motor using ESP32-Microcontroller.

Hardware required:
ESP32-Microcontroller Development board



Procedure:

1. Connect P3 port and SV9 port using FRC cable.
2. Connect the USB cable to the board.
3. Open Arduino IDE .Select DOIT ESP32 DEVKIT V1in boards and select COM port.
4. Verify the program and Upload it.

Program:

```
const int En1=12,En2=13;                //initializing enable pins
const int in1=15, in2=2, in3=0, in4=4;  //initializing input pins
int count=0;

void setup()
{
  // channel A
  pinMode(En1,OUTPUT);
  pinMode(in1,OUTPUT);
  pinMode(in2,OUTPUT);

  // channel B
  pinMode(En2,OUTPUT);
  pinMode(in3,OUTPUT);
  pinMode(in4,OUTPUT);
}

void loop() {

   digitalWrite(En1,HIGH);  //enabling motor1
   digitalWrite(En2,LOW);

  // Motor 1 clockwise rotation
  while(count!=100)            //motor1 keep rotating for 1 minute in clockwise direction
  {
    digitalWrite(in1,HIGH);
    digitalWrite(in2,LOW);
    delay(600);
    count++;
  }
count=0;

  //Motor1 anticlockwise rotation
  while(count!=100)            //motor1 keep rotating for 1 minute in anti-clockwise direction
  {
    digitalWrite(in1,LOW);
    digitalWrite(in2,HIGH);
    delay(600);
    count++;
  }
count=0;
    digitalWrite(in1,LOW);   //to stop motor1
```

```
    digitalWrite(in2,LOW);
    delay(100);

   digitalWrite(En1,LOW);    //enabling  Motor 2
   digitalWrite(En2,HIGH);
// Motor 2 clockwise rotation
  while(count!=100)          //motor2  keep rotating  for 1 minute  in clockwise direction
  {
    digitalWrite(in3,HIGH);
    digitalWrite(in4,LOW);
    delay(600);
    count++;
  }
  count=0;

  //Motor2 anticlockwise  rotation

  while(count!=100)         //motor2 keep rotating  for 1 minute  in anti-clockwise  direction
  {
    digitalWrite(in3,LOW);
    digitalWrite(in4,HIGH);
    delay(600);
    count++;
  }
  //to stop motor2
    digitalWrite(in3,LOW);   //to stop motor2
    digitalWrite(in4,LOW);
    delay(100);
}
```

EXPERIMENT NO 9
SD CARD

Aim:

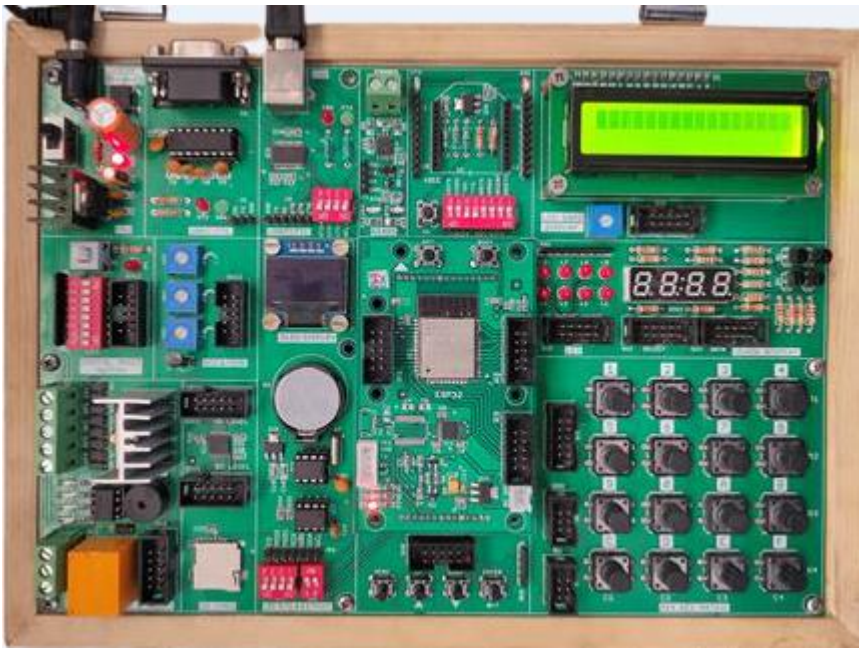
To read the stored directories in SD card using ESP 32 microcontroller development board.

Description:

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

Hardware required:

ESP32-Microcontroller Development board



Procedure:

1. Insert the SD Card in the slot given in the board.
2. Connect the USB cable to the board.
3. Open Arduino IDE .Select DOIT ESP32 DEVKIT V1in boards and select COM port.
4. Verify the program and Upload it.
5. Open the serial monitor to observe the output.

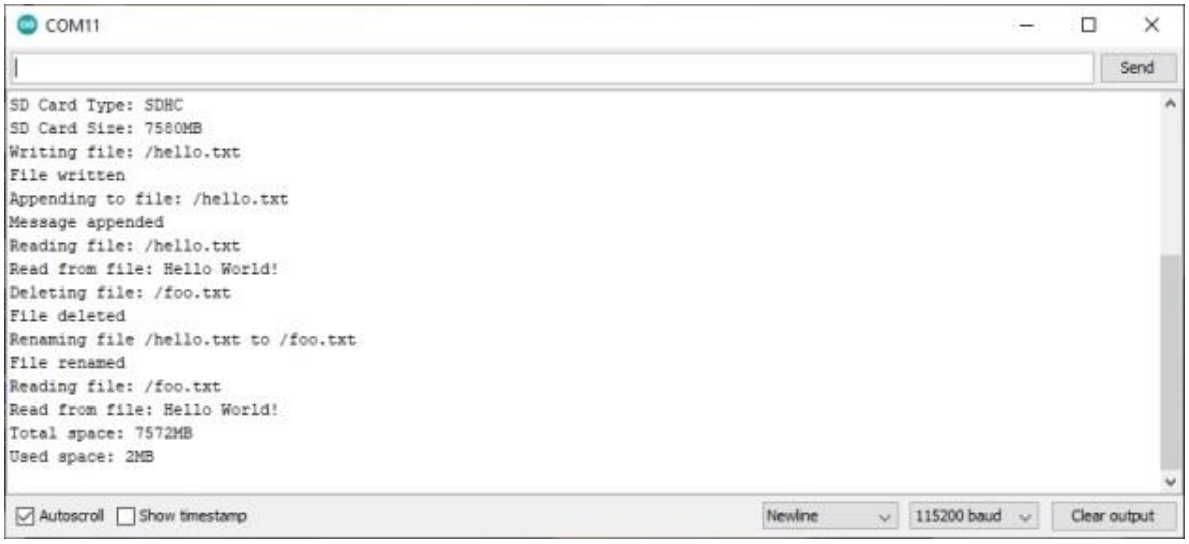
Program:

```
#include "FS.h"
#include "SD.h"
#include "SPI.h"

void writeFile(fs::FS &fs, const char * path, const char * message){
    Serial.printf("Writing file: %s\n", path);

    File file = fs.open(path, FILE_WRITE);
    if(!file){
        Serial.println("Failed to open file for writing");
        return;
    }
    if(file.print(message)){
        Serial.println("File written");
    } else {
        Serial.println("Write failed");
    }
    file.close();
}

void appendFile(fs::FS &fs, const char * path, const char * message){
    Serial.printf("Appending to file: %s\n", path);

    File file = fs.open(path, FILE_APPEND);
    if(!file){
        Serial.println("Failed to open file for appending");
        return;
    }
    if(file.print(message)){
        Serial.println("Message appended");
    } else {
        Serial.println("Append failed");
    }
    file.close();
}

void readFile(fs::FS &fs, const char * path){
    Serial.printf("Reading file: %s\n", path);

    File file = fs.open(path);
    if(!file){
        Serial.println("Failed to open file for reading");
        return;
    }

    Serial.print("Read from file: ");
    while(file.available()){
        Serial.write(file.read());
```

```
    }
    file.close();
}

void renameFile(fs::FS &fs, const char * path1, const char * path2)
{
    Serial.printf("Renaming file %s to %s\n", path1, path2);
    if (fs.rename(path1, path2))
    {
        Serial.println("File renamed");
    }
    else
    {
        Serial.println("Rename failed");
    }
}

void deleteFile(fs::FS &fs, const char * path)
{
    Serial.printf("Deleting file: %s\n", path);
    if(fs.remove(path)){
        Serial.println("File deleted");
    } else {
        Serial.println("Delete failed");
    }
}

  void setup(){
    Serial.begin(115200);
    if(!SD.begin())
    {
        Serial.println("Card Mount Failed");
        return;
    }
    uint8_t cardType = SD.cardType();

    if(cardType == CARD_NONE){
        Serial.println("No SD card attached");
        return;
    }

    Serial.print("SD Card Type: ");
    if(cardType == CARD_MMC)
    {
        Serial.println("MMC");
    } else if(cardType == CARD_SD){
        Serial.println("SDSC");
    } else if(cardType == CARD_SDHC){
        Serial.println("SDHC");
    } else {
```

```
        Serial.println("UNKNOWN");
    }

    uint64_t cardSize = SD.cardSize() / (1024 * 1024);
    Serial.printf("SD Card Size: %lluMB\n", cardSize);
    writeFile(SD, "/hello.txt", "Hello ");
    appendFile(SD, "/hello.txt", "World!\n");
    readFile(SD, "/hello.txt");
    deleteFile(SD, "/foo.txt");
    renameFile(SD, "/hello.txt", "/foo.txt");
    readFile(SD, "/foo.txt");
//    testFileIO(SD, "/test.txt");
    Serial.printf("Total space: %lluMB\n", SD.totalBytes() / (1024 * 1024));
    Serial.printf("Used space: %lluMB\n", SD.usedBytes() / (1024 * 1024));
}

void loop()
{

}
```

Output:



EXPERIMENT NO 10
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

Procedure:

1. Connect the USB cable to the board.
2. Open Arduino IDE .Select DOIT ESP32 DEVKIT V1 in boards and select COM port.
3. Verify the program and Upload it.
4. Now you can see the output displaying the message on OLED of ESP32 microcontroller board.

Program:

```
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

#define SCREEN_WIDTH 128 // OLED display width, in pixels
#define SCREEN_HEIGHT 32 // OLED display height, in pixels

#define OLED_RESET     4 // Reset pin
#define SCREEN_ADDRESS 0x3C //0x3C for 128x32pixels OLED
Adafruit_SSD1306 display (SCREEN_WIDTH, SCREEN_HEIGHT, &Wire,
OLED_RESET);

void setup() {
  Serial.begin(9600);
  if(!display.begin(SSD1306_SWITCHCAPVCC, SCREEN_ADDRESS)) //
SSD1306_SWITCHCAPVCC = generate display voltage from 3.3V internally
  {
    Serial.println(F("SSD1306 allocation failed"));
    for(;;); // Don't proceed, loop forever
  }

}

void loop() {
  text();
  display.invertDisplay(true);
  delay(2000);
  display.invertDisplay(false);
  delay(2000);

}

void text(void) {
  display.clearDisplay();
  display.setTextSize(2);
  display.setTextColor(SSD1306_WHITE);          // Draw white text
  display.setCursor(5,3);                       //setting cursor on X Y plane
  display.print(F("Welcome To ...RDL..."));

  display.setTextColor(SSD1306_BLACK, SSD1306_WHITE); // Draw 'inverse' text

  display.display();
  delay(2000);
}
```

EXPERIMENT NO 11
TEMPERATURE SENSOR

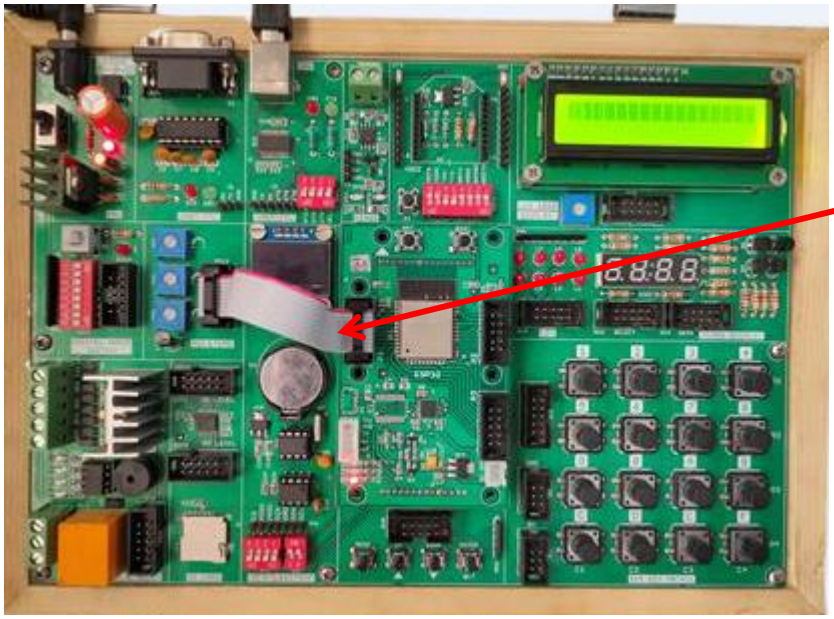
Aim:
To extract information from temperature sensor.

Description:

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

Hardware required:

ESP32-Microcontroller Development board



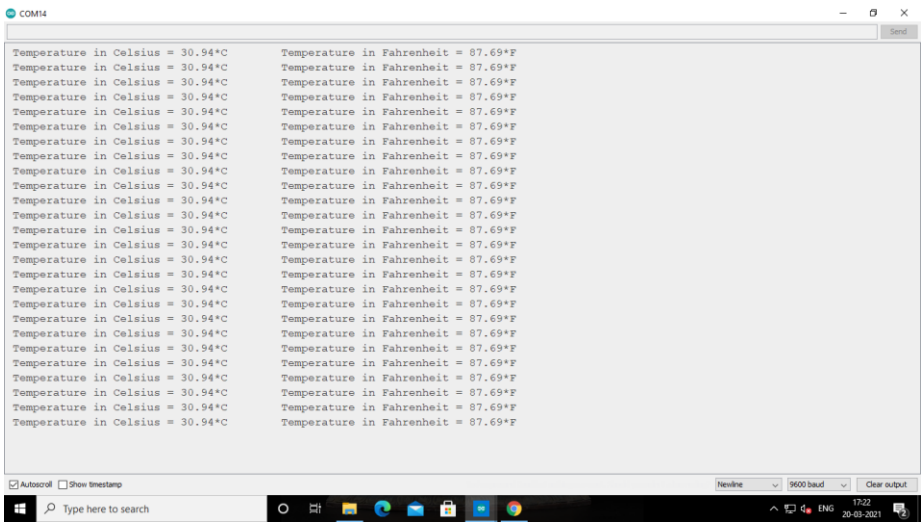
Procedure:

1. Connect P1 port and SV12 port using FRC cable as shown above
2. Connect the USB cable to the board.
3. Open Arduino IDE .Select DOIT ESP32 DEVKIT V1in boards and select COM port.
4. Verify the program and Upload it.
5. Now you can see the output on the serial monitor.

PROGRAM:

```
const int tempPin = 33;          // pin 33 of port P1 connected to LM35 output

int Value;
double milivlt,Cel,Far;

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

void loop()
{
  Value = analogRead(tempPin);          //read sensor output value
  milivlt  = (Value / 2048.0) * 3300;    // converting it to millivots.
  Cel = milivlt  * 0.1;                 // calculating temperature in Celsius
  Far = (Cel * 1.8) + 32;               // convert from C to Fahrenheit

  Serial.print(" Temperature in Celsius = ");
  Serial.print(Cel);
  Serial.print("*C");
  Serial.print("\t Temperature in Fahrenheit = ");
  Serial.print(Far);
  Serial.println("*F");
  delay(2000);                          //check the temperature every 2 second
```

OUTPUT:



EXPERIMENT NO 12
MQTT

Aim:

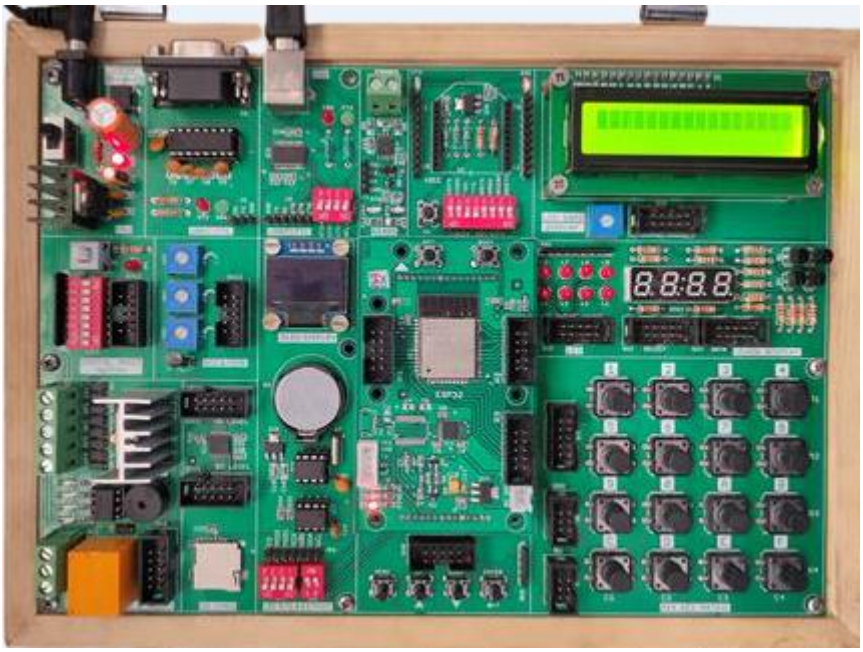
To interface Wi-Fi module with ESP32 to receive data from cloud using RDL ESP32 Development board.

Description:

Interfacing Wi-Fi module with the cloud MQTT to receive and send data using ESP32 microcontroller.

Hardware required:

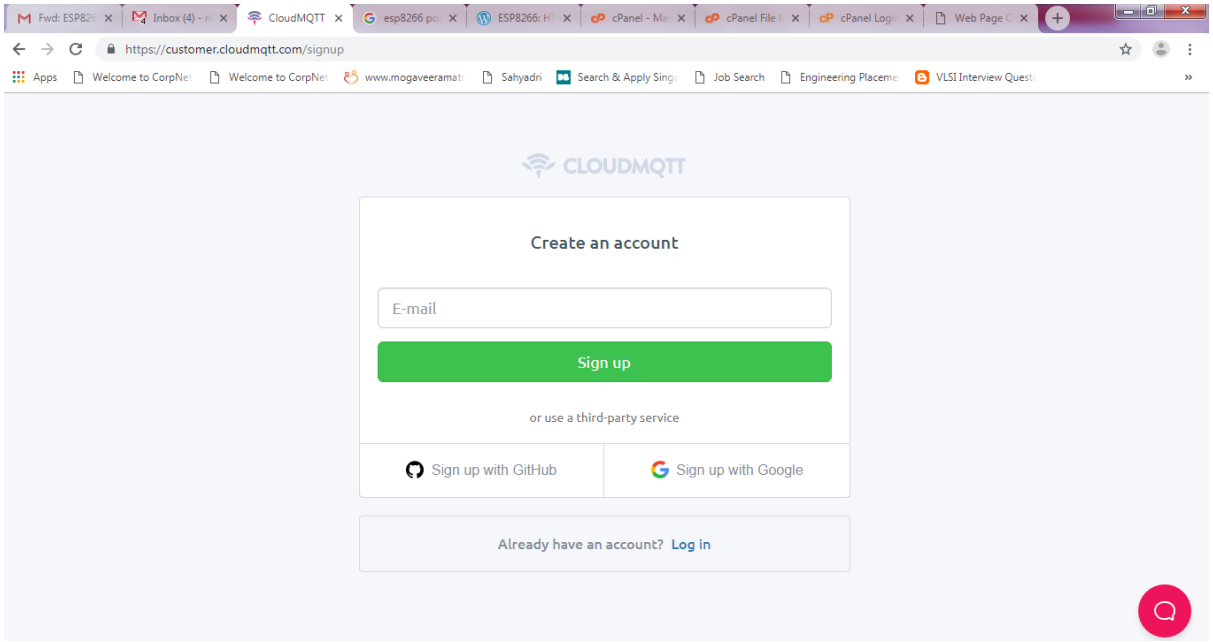
ESP32-Microcontroller development board.

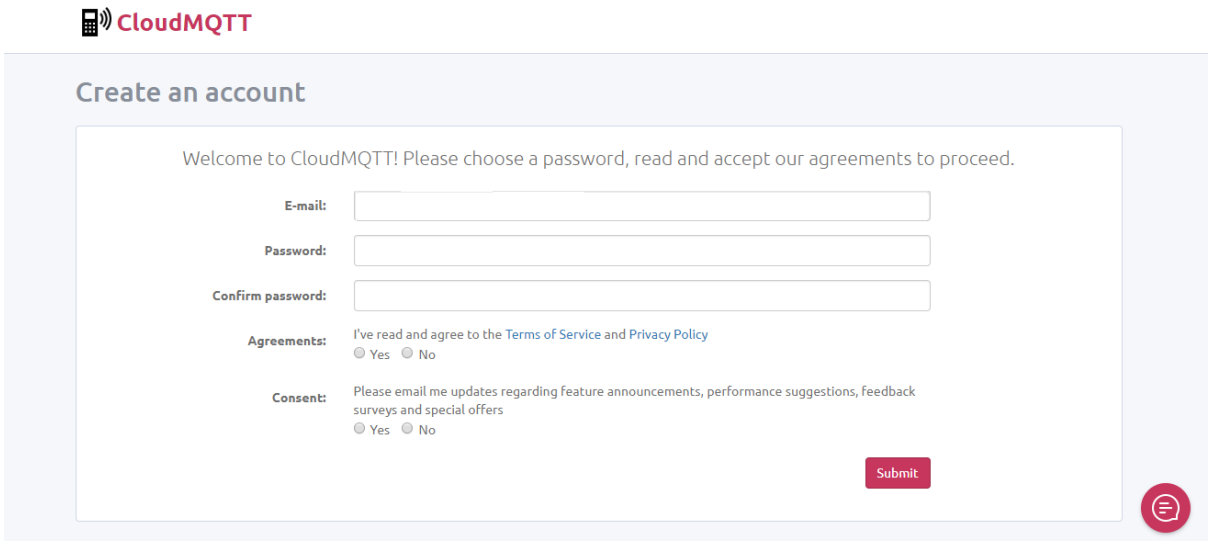


Procedure:

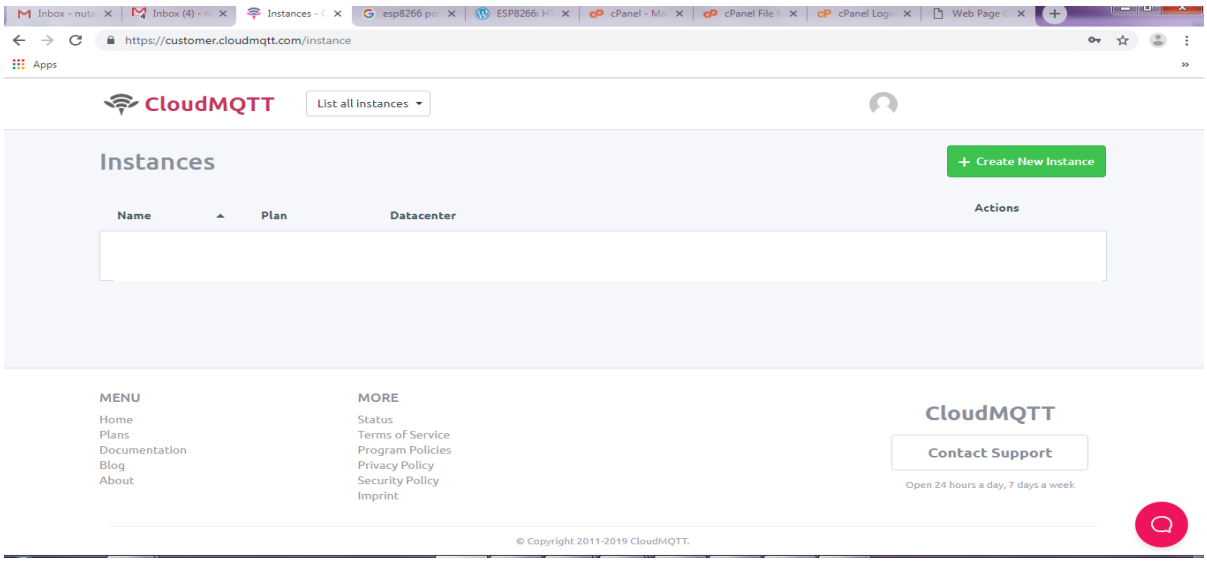
Uploading and Receiving Data from Cloud using ESP32

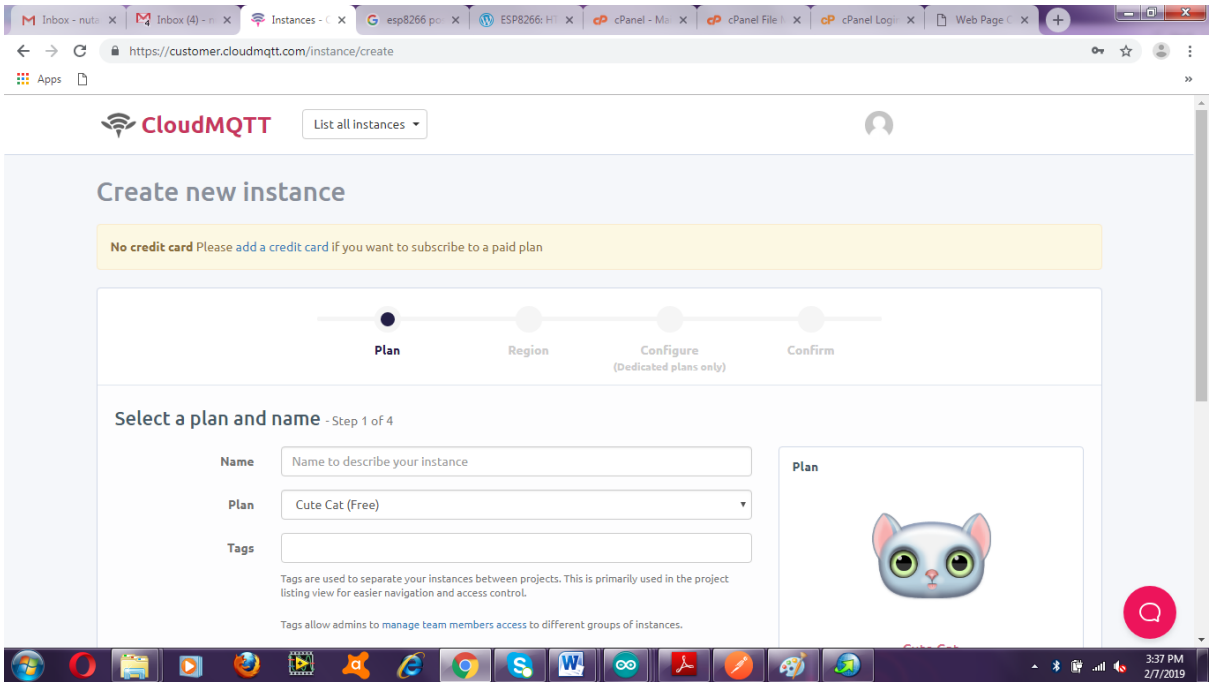
1. Create an Account in www.cloudmqtt.com

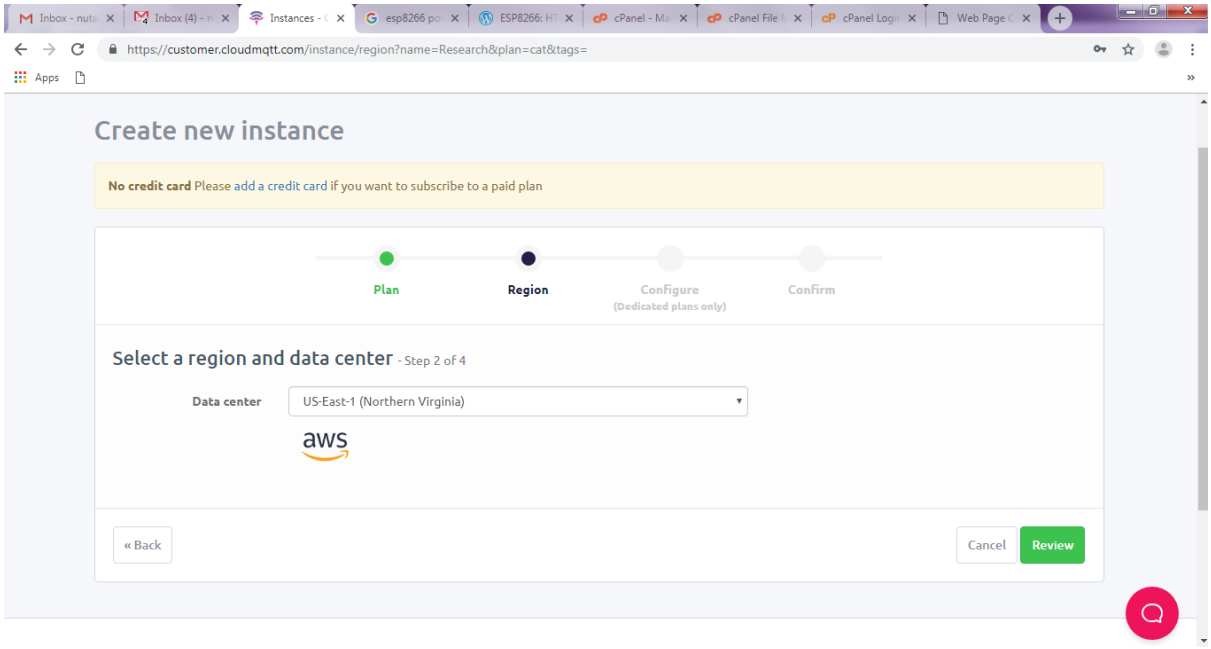




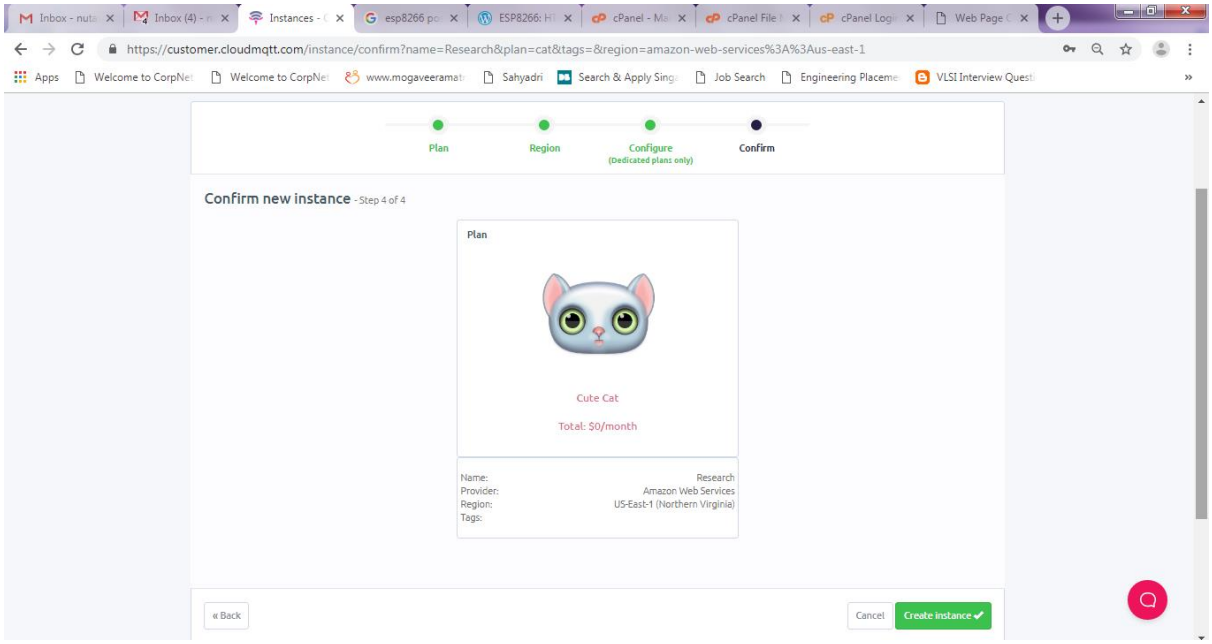
2. Create an instance which will be used in the program to publish data to the cloud.


https://customer.cloudmqtt.com/instance
CloudMQTT
List all instances
Instances
Create New Instance
Name
Plan
Datacenter
Actions
MENU
Home
Plans
Documentation
Blog
About
MORE
Status
Terms of Service
Program Policies
Privacy Policy
Security Policy
Imprint
CloudMQTT
Contact Support
Open 24 hours a day, 7 days a week
© Copyright 2011-2019 CloudMQTT.

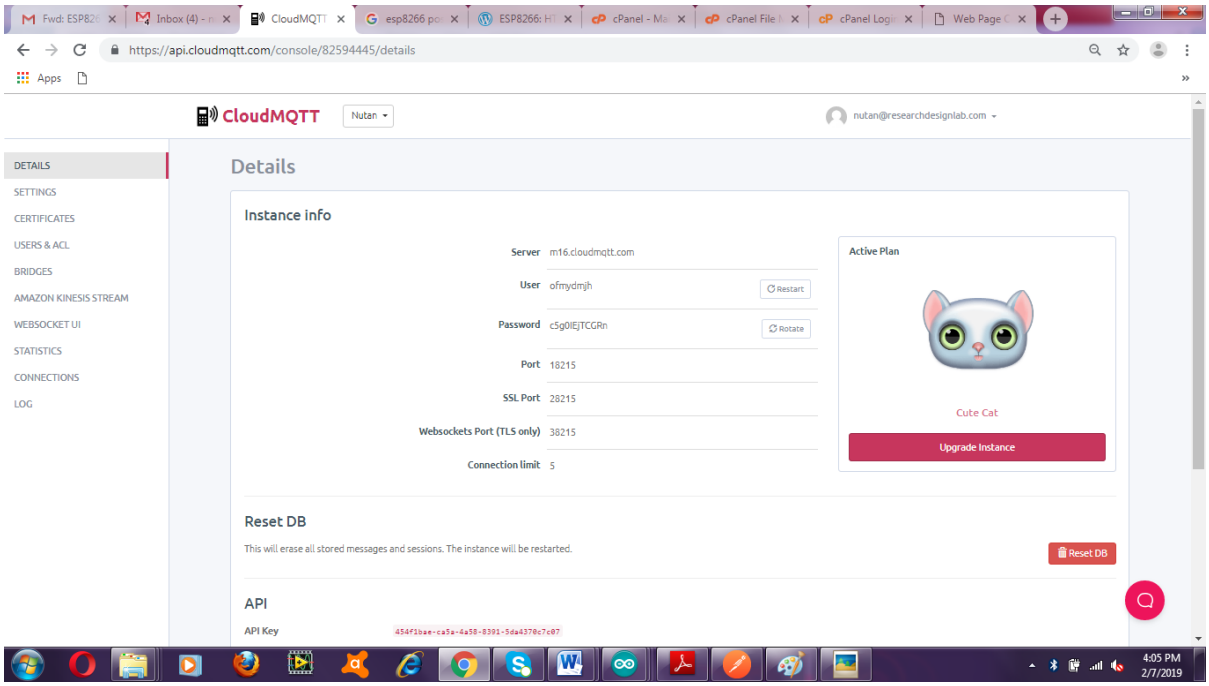

https://customer.cloudmqtt.com/instance/create
CloudMQTT
List all instances
Create new instance
No credit card Please add a credit card if you want to subscribe to a paid plan
Plan
Region
Configure
(Dedicated plans only)
Confirm
Select a plan and name - Step 1 of 4
Name
Name to describe your instance
Plan
Cute Cat (Free)
Tags
Tags are used to separate your instances between projects. This is primarily used in the project listing view for easier navigation and access control.
Tags allow admins to manage team members access to different groups of instances.
Plan
3:37 PM
2/7/2019



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



4.

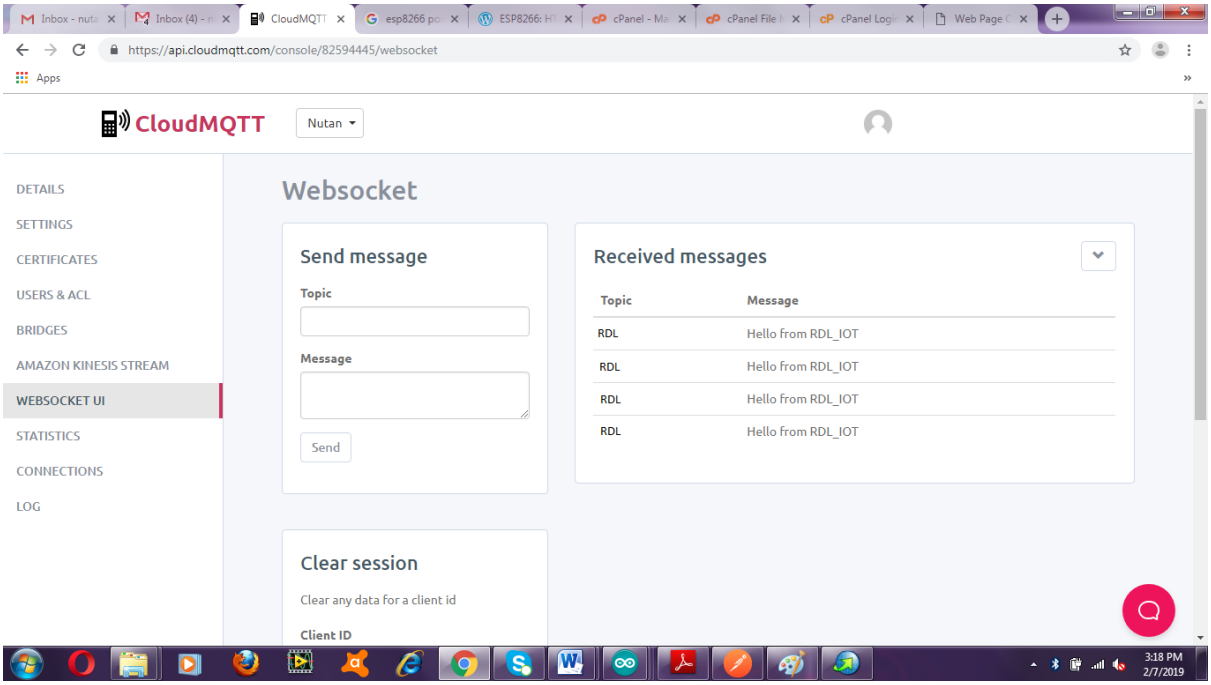


5. Enter the details given under server, port, user, ssl port in the given program.

```
const char* mqttServer =”m16.mqtt.com”;
const int mqttPort = 18215;
const char* mqttUser = “ofmydmjh”;;
const char* mqttPassword = “c5goIEjTCGRn”;
```

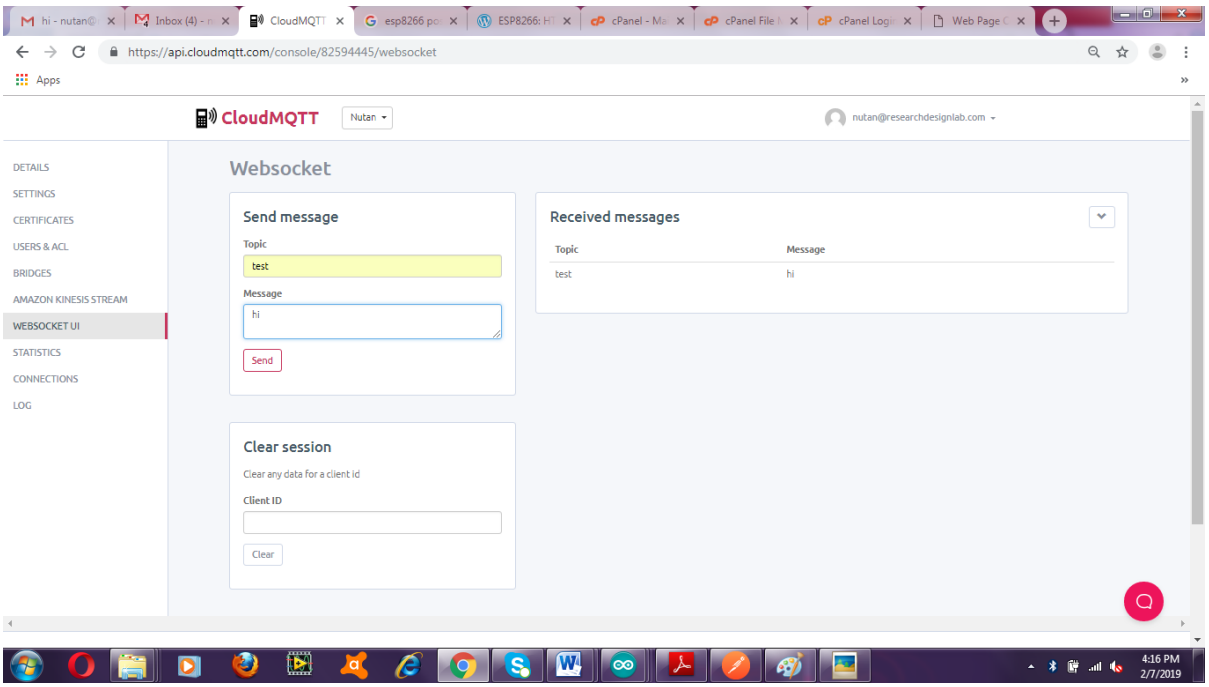
6. In order to send a message to the CloudMQTT, include the line

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



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

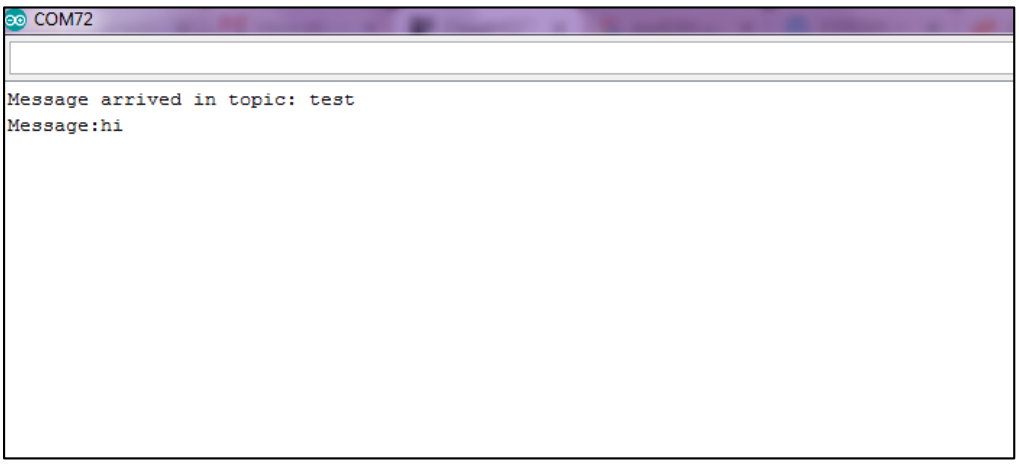
```
client.subscribe("test");
```



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);
}
}
```

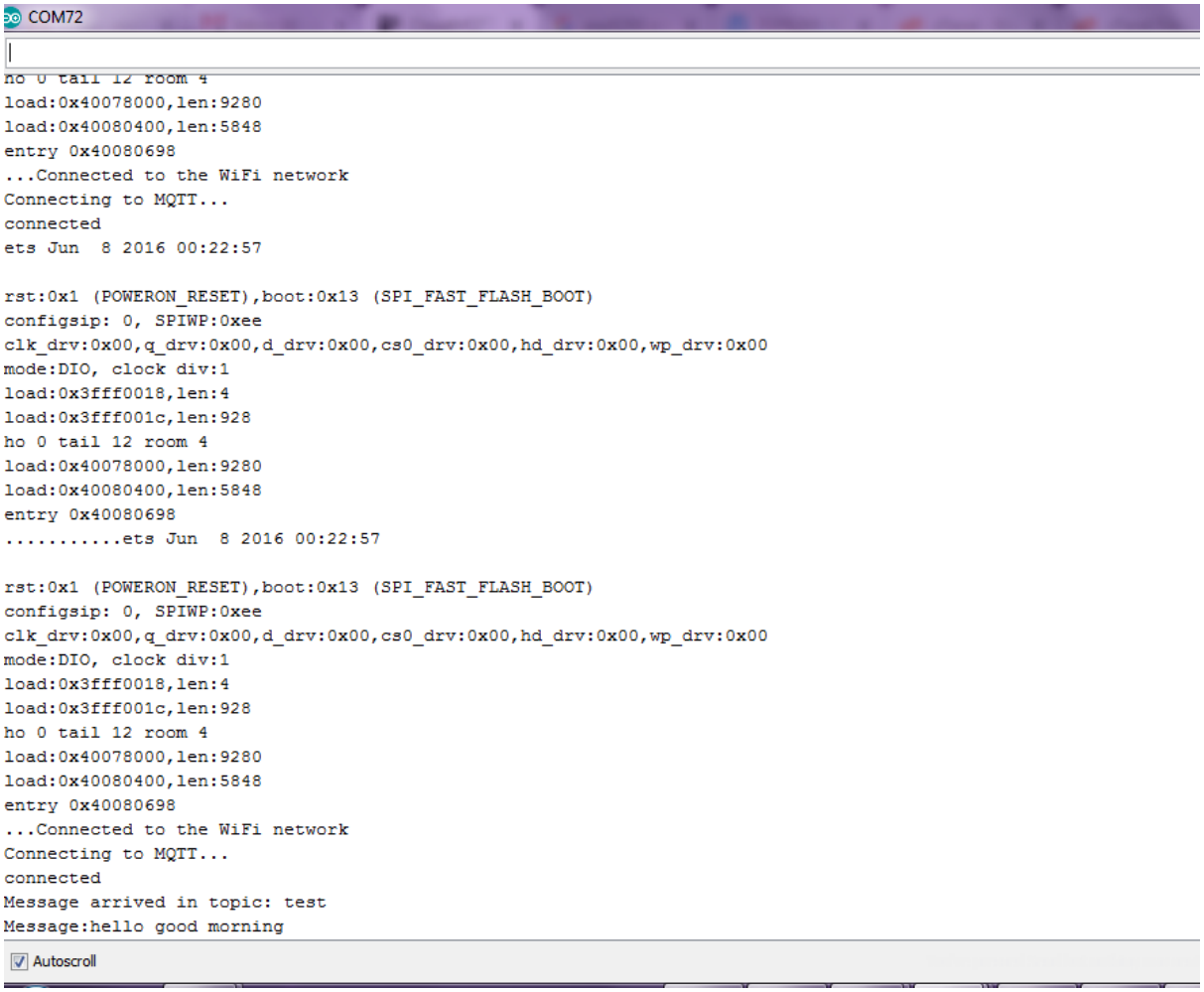


9. The final Program:

```
#include <WiFi.h>
#include <PubSubClient.h>
const char* ssid = "J7 pro";
const char* password = "fzao2408";
/* 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("Sub", "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:

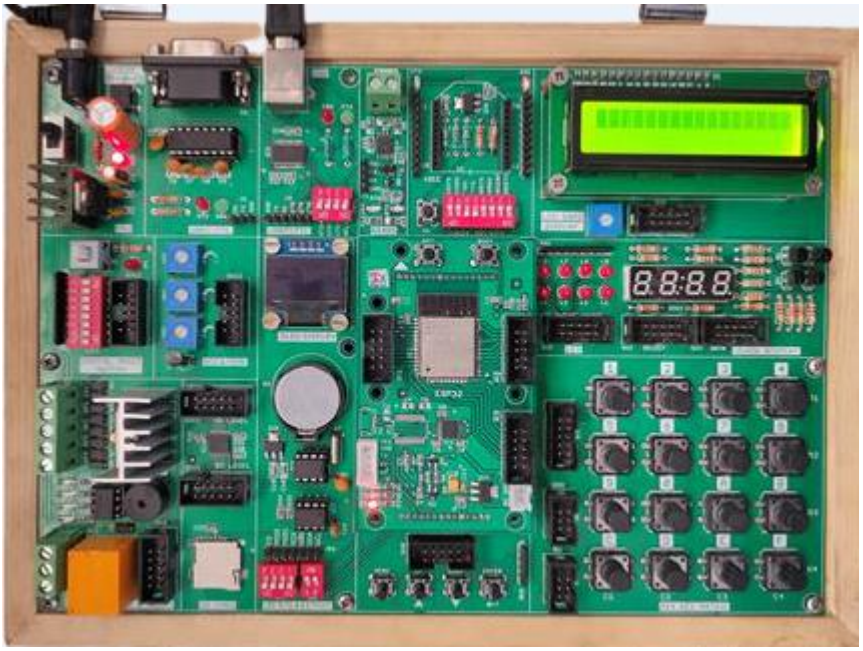

COM72
ho 0 tail 12 room 4
load:0x40078000,len:9280
load:0x40080400,len:5848
entry 0x40080698
...Connected to the WiFi network
Connecting to MQTT...
connected
ets Jun 8 2016 00:22:57
rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:1
load:0x3fff0018,len:4
load:0x3fff001c,len:928
ho 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
ho 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
Autoscroll

EXPERIMENT NO 13
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 ESP32microcontroller.

Hardware required:
ESP32-Microcontroller development board.



Procedure:

1. Connect the USB cable to the ESP32 development board.
2. Open Arduino IDE .Select DOIT ESP32 DEVKIT V1in boards and select COM port.
3. Verify the program and upload it.
4. Now you can see wifi connected along with the IP address on the serial monitor.
5. To check the inserted values of gas and temperature use the server address and check it on the browser.

Program:

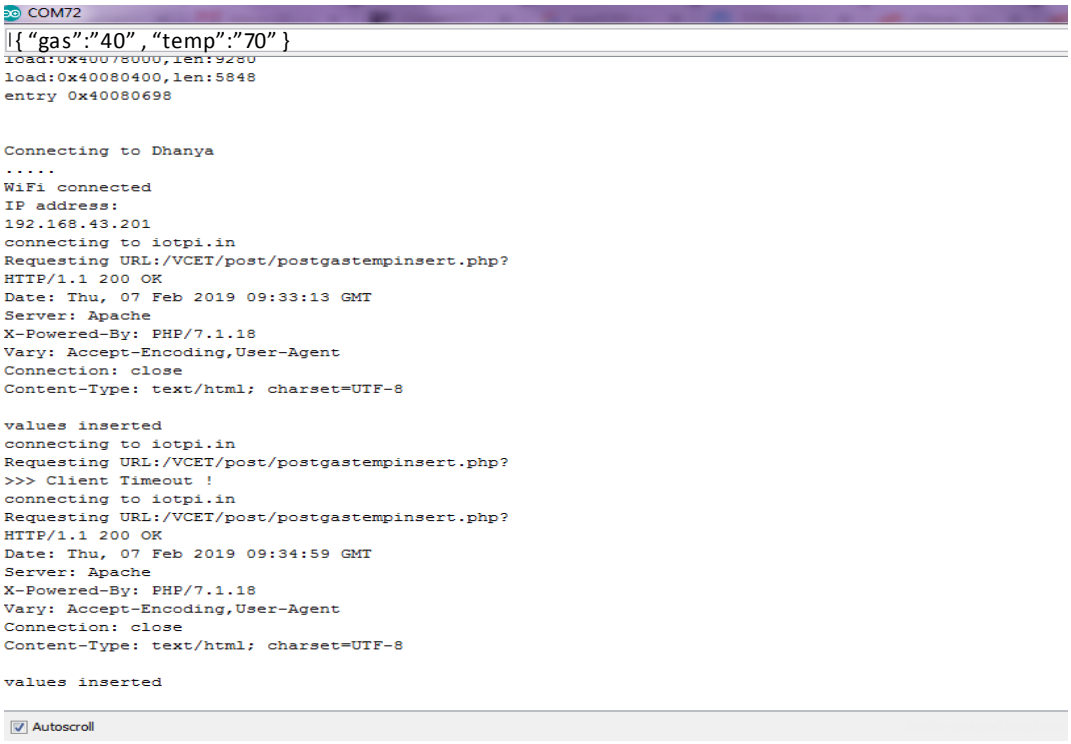
```
/*
 *  This sketch sends data via HTTP GET requests to data.sparkfun.com service.
 *  You need to get streamId and privateKey at data.sparkfun.com and paste them
 *  below. Or just customize this script to talk to other HTTP servers.
 *
 */
#include <WiFi.h>
/* type your username */
char ssid[]     = "your ssid";
/* type your password */
char password[] = "your password";
const char* host = "iotpi.in";
const char* streamId   = ".....";
const char* privateKey = ".....";

WiFiClient esp32client;
void setup(void)
{
  Serial.begin(115200);
  delay(10);
  /* We start by connecting to a WiFi network */
  Serial.println();
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  /* Explicitly set the ESP32 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.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(void)
{
String url="/VCET/post/postgastempinsert.php?";
String jason_string="";
char cmd=0;
while(cmd!=13)
 {
   if (Serial.available())
   {
     cmd=Serial.read();
     if(cmd!=13)
     jason_string+=cmd;
   }
 }
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 */
Serial.print("Requesting  URL:");
Serial.println(url);
esp32client.print(String("POST  ") + url + " HTTP/1.0\r\n" +
             "Host:  " + host + "\r\n" +
             "Accept: *" + "/" + "*\r\n" +
             "Content-Length:  " + jason_string.length()  + "\r\n" +
             "Content-Type:  application/json\r\n"  +
             "\r\n"  + jason_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()){
```

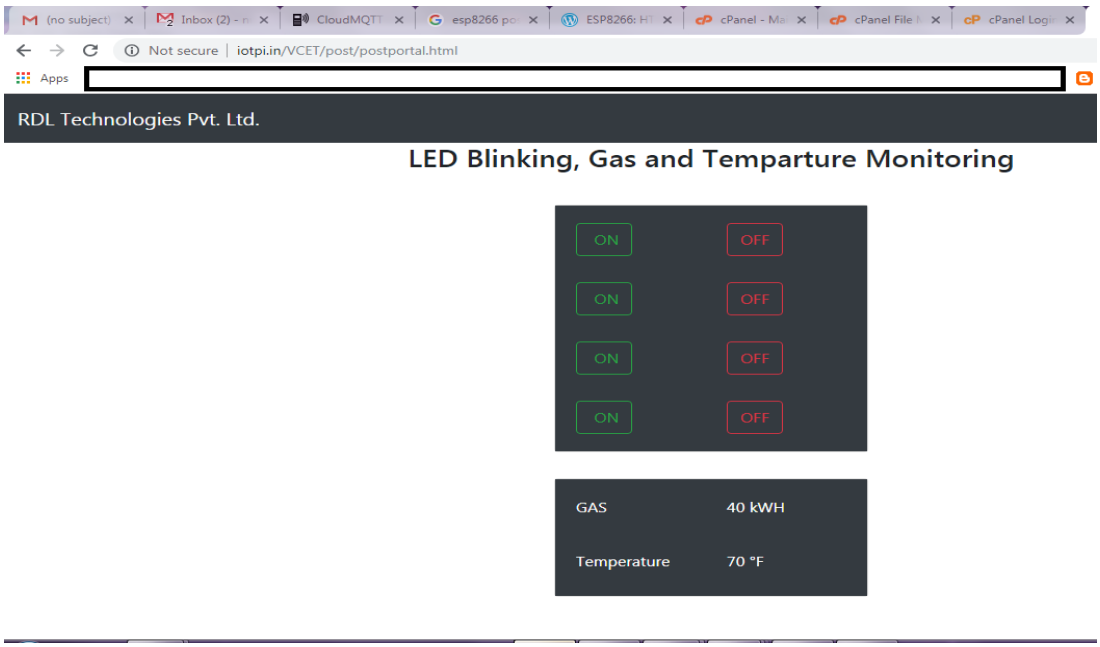
```
String line = esp32client.readStringUntil('\r');
Serial.print(line);
}
Serial.println();
}
```

Output:

1. Open serial monitor you can see Wi-Fi connected along with IP address
Insert { "gas":"40" , "temperature":"70" } in the serial monitor and press enter.



2. Enter iotpi.in/VCET/post/postportal.html in your browser.

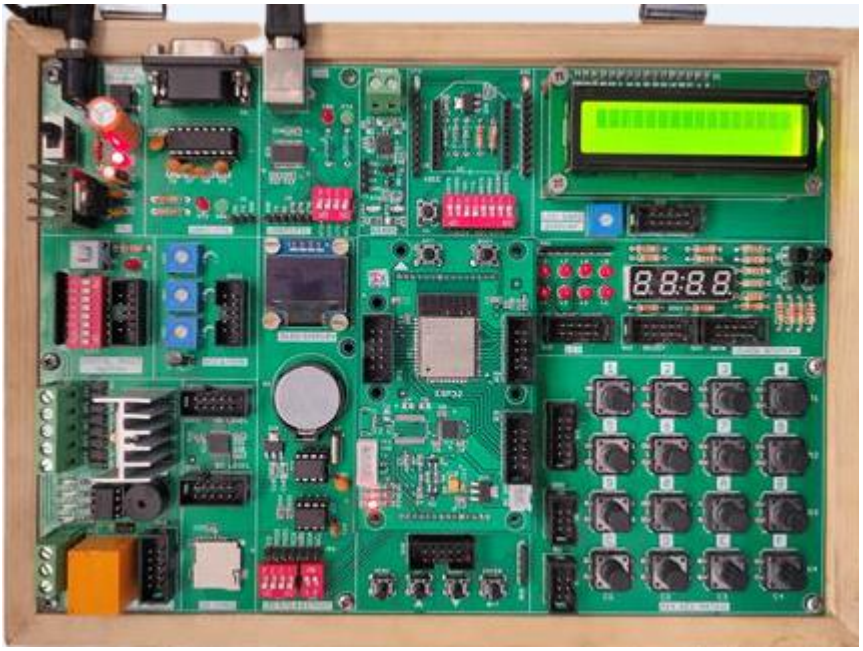

Not secure | iotpi.in/VCET/post/postportal.html
Apps
RDL Technologies Pvt. Ltd.
LED Blinking, Gas and Temparture Monitoring
ON
OFF
ON
OFF
ON
OFF
ON
OFF
GAS
40 kWH
Temperature
70 °F

EXPERIMENT NO 14
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.



Procedure:
1. Connect the USB cable to the ESP32 development board.
2. Open Arduino IDE .Select DOIT ESP32 DEVKIT V1in boards and select COM port.
3. Verify the program and upload it.

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 username";
const char* password = "your password";
/* ftp server */
char server_link[] = "abcd.xxxxx.com";
byte clientBuf[]="i love my INDIA";
/* size of data string */
int clientCount = sizeof(clientBuf);
/* 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(void)
{
  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(void)
{
  byte inChar;
  inChar = Serial.read();
  if(inChar == 'f')
  {
    if(doFTP()) Serial.println(F("FTP OK"));
    else Serial.println(F("FTP FAIL"));
  }
}
```

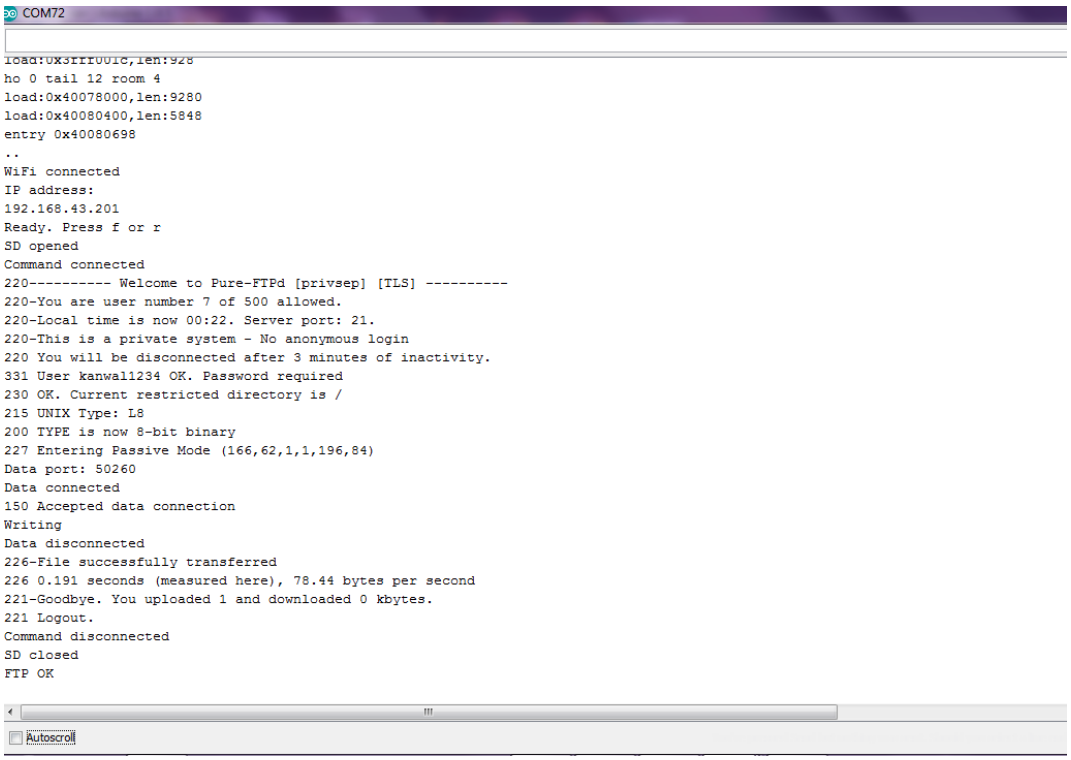
```
 byte doFTP()
{
  Serial.println(F("SD  opened"));
  if (client.connect(server_link,21))  {
  Serial.println(F("Command  connected"));
}
else
{
  Serial.println(F("Command  connection  failed"));
  return 0;
}
if(!eRcv()) return 0;
/* ftp username */
client.println(F("USER  xxxxxxx"));
if(!eRcv()) return 0;
/* ftp password */
client.println(F("PASS  xxxx@13xx5"));
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();
return 0;
}
#ifdef  FTPWRITE
client.print(F("STOR  "));
```

```
client.println(fileName);
#else
client.print(F("RETR "));
client.println(fileName);
#endif
if(!eRcv())
 {
  dclient.stop();
  return 0;
 }
#ifdef FTPWRITE
Serial.println(F("Writing"));
if(clientCount > 0) dclient.write(clientBuf,clientCount);
#else
 while(dclient.connected())
 {
  while(dclient.available())
  {
   char c = dclient.read();
   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"));
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;
  }
```

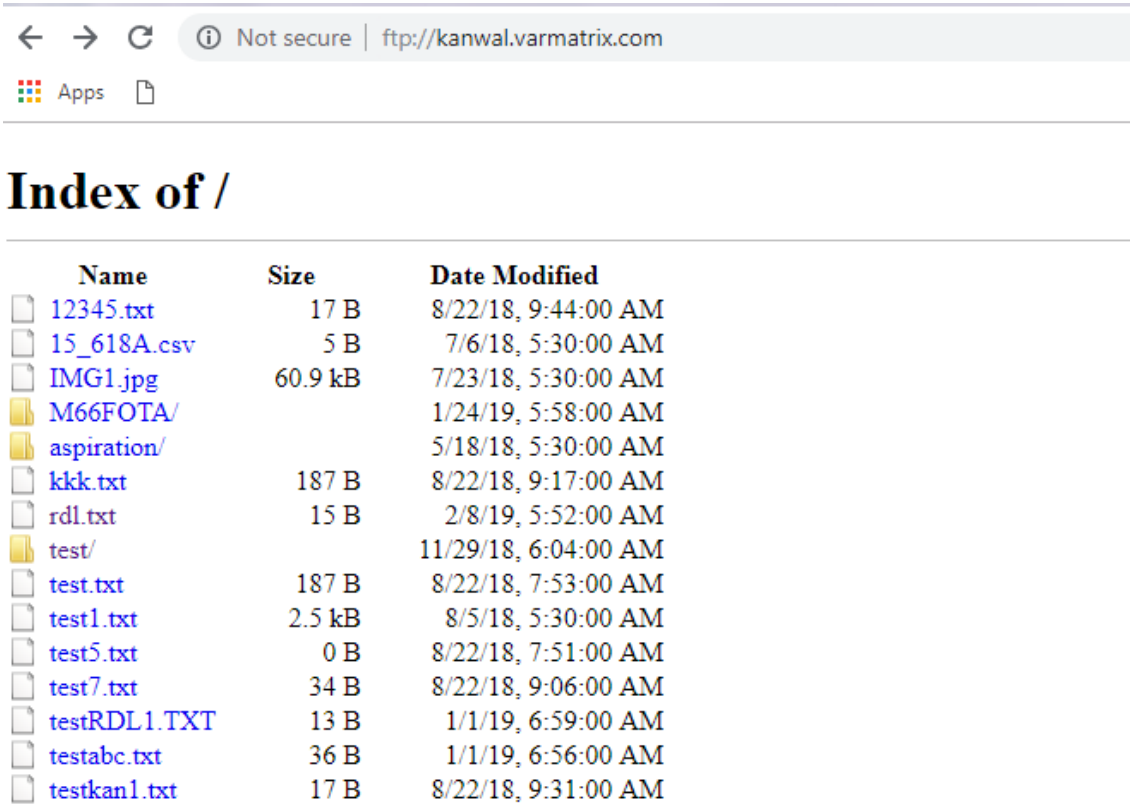
```
  }
if(respCode >= '4')
  {
    efail();
    return 0;
  }
  return 1;
}
void efail()
{
  byte thisByte = 0;
  client.println(F("QUIT"));
  while(!client.available()) delay(1);
  while(client.available())
  {
    thisByte = client.read();
    Serial.write(thisByte);
  }
client.stop();
Serial.println(F("Command disconnected"));
Serial.println(F("SD closed"));
}
```

Output:

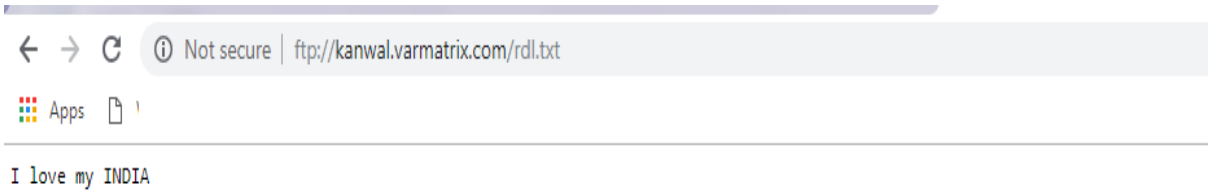
1. Gives information whether the WiFi is connected to the server.



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.



3. The contents in the folder are given below.

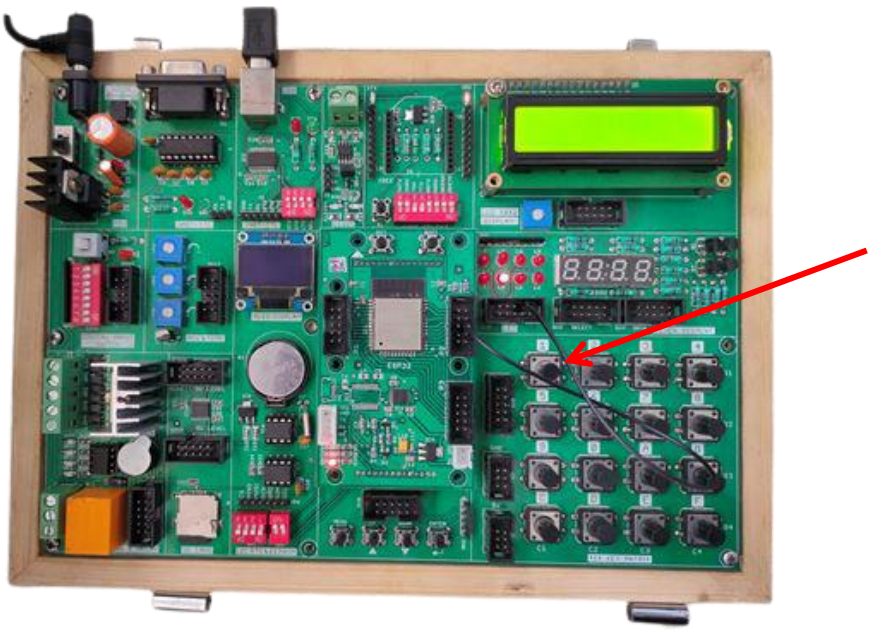


EXPERIMENT NO 15
OTA (Over the air) programming

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.

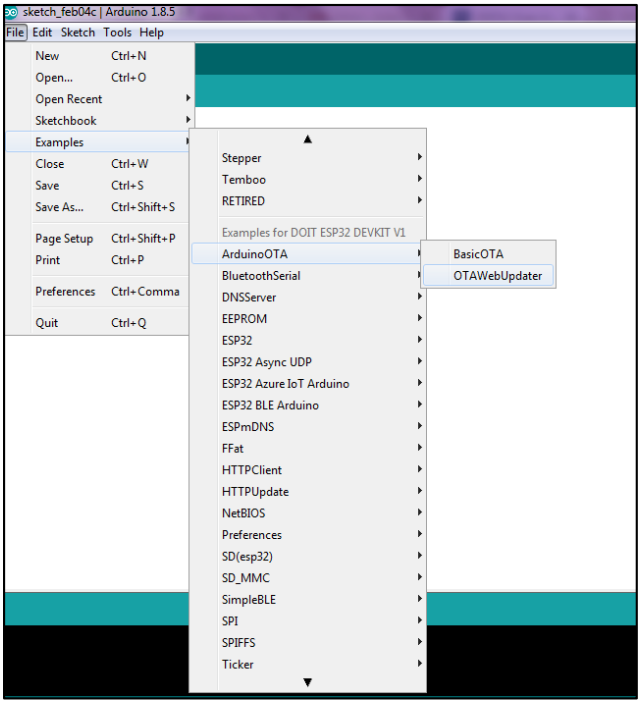
Hardware required:
ESP32-Microcontroller development board.



Connect pin 5(P1 port) and any pin of SV2 port using patch chords.

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

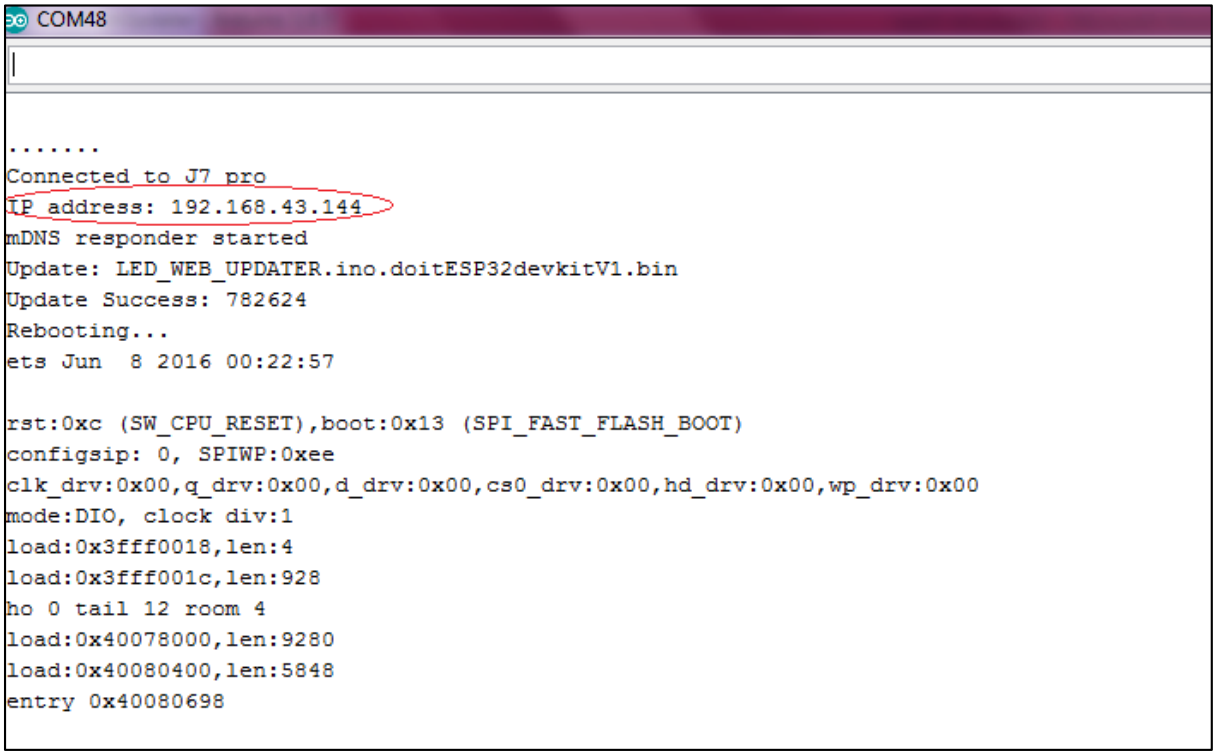
```
#include <WiFi.h>
#include <WiFiClient.h>
#include <WebServer.h>
#include <ESPmDNS.h>
#include <Update.h>

const char* host = "esp32";
const char* ssid = "****";              // enter your WiFi SSID
const char* password = "*****";         // enter your WiFi Password

WebServer server(80);                   //WebServer initiated with port 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>"
 "<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')"              //You can
assign your user ID Password here
   "{"
   "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) {"
```

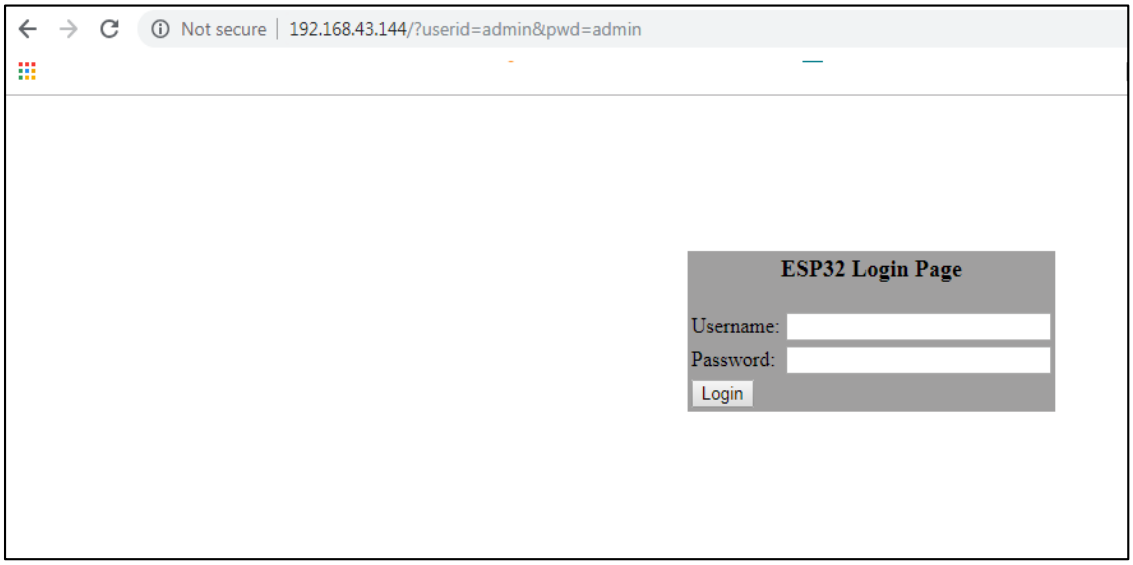
```
 "var per = evt.loaded / evt.total;"
 "$('#prg').html('progress: ' + Math.round(per*100) + '%');"
 "}"
 "}, false);"
 "return xhr;"
 "},"
 "success:function(d, s) {"
 "console.log('success!')"
"},"
"error: function (a, b, c) {"
"}"
"});"
"});"
"</script>";

/*
 * setup function
 */
void setup(void) {
/*******************/
// use this section to write your new code which needs to run once for every Reset.
pinMode(5,OUTPUT);
/***************/

  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();
  /**********************/
 // use this section to write your new code which needs to run continuously.
  digitalWrite(5,HIGH);
  delay(1000);
  digitalWrite(5,LOW);
  delay(1000);
  /**********************/

  delay(10);
}
```

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.

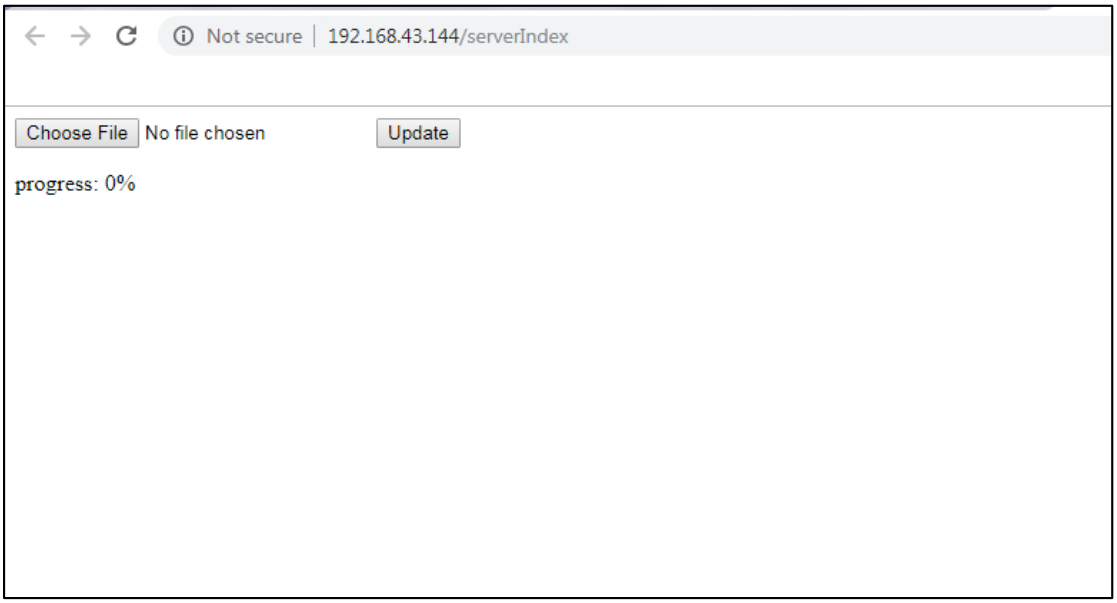


8. Copy the IP address which you obtain from the serial monitor and paste it in your browser.



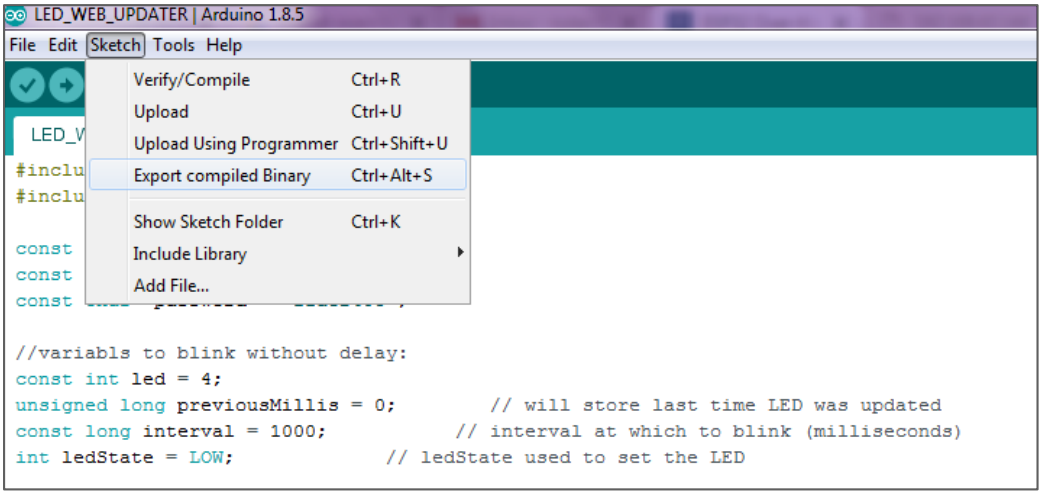
9. Username: admin
   Password: admin

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 .

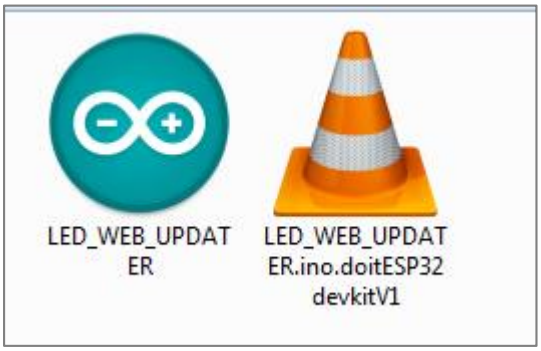


Preparing the sketch:

1. Write a simple program (Blinking of LED) using OTA web server and save it with the name LED_blink and complile 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.