# Task number: 01

## Task name: blinking internal led of ESP32.

**Reference:**https://iotdesignpro.com/projects/getting-started-with-esp32-program-it-using-arduino-ide-blinking-led

**Components used:** ESP32 dev module board, USB cable, Arduino IDE.

**Connections:** connected the ESP32 to the computer using an USB cable.

## Code

```
Int led_btn=2;
void setup(){
   pinMode(led_btn, OUTPUT);
}
void loop(){
  digitalWrite(led_btn, HIGH);
  delay(1000);
  digitalWrite(led_btn, LOW);
  delay(1000);
}
```

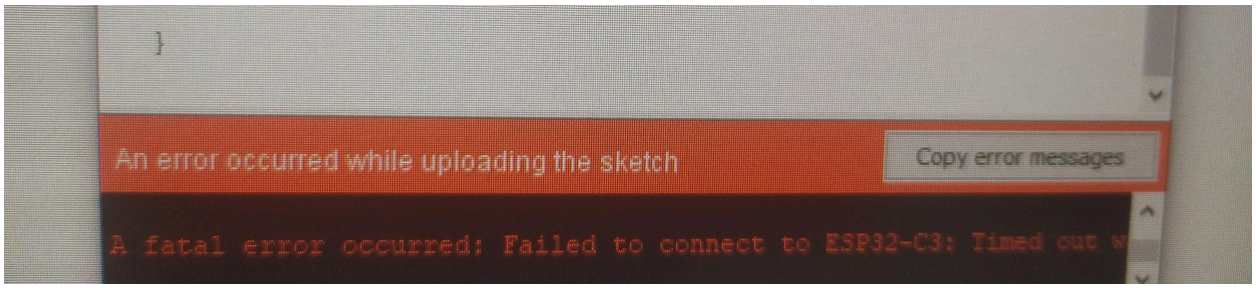
Here, we first declare a variable led_btn as 2, which is the pin number of internal led of ESP32. Next we set up pinmode for the same, as we are using it to observe output, we set it as output. Next step, we write for how long the delay should be after ON(HIGH) or OFF(LOW) of led, this loop continues to run till we disconnect the USB cable or upload a new program.

## Procedure

1. Create a sketch on Arduino IDE for this project.
2. Write the above code, save and verify.
3. Connect the ESP32 to the computer using USB cable.
4. Go to tools, select appropriate board name and COM port.
5. Once verified, upload this code to the ESP32.
6. Now observe the internal LED blink.

## Problem faced

There was no syntax or compilation error, but when I tried to upload the code to ESP32, it failed and threw me an error, “fatal error-package not found.” Another time it showed different error, which was:

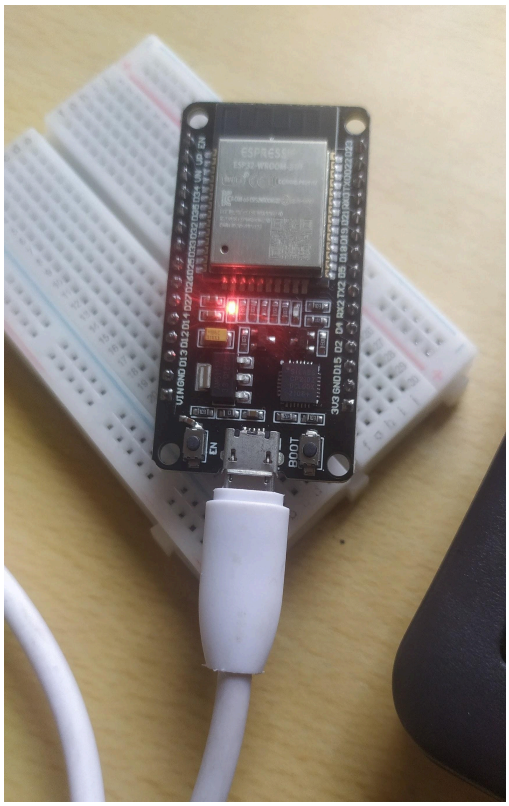


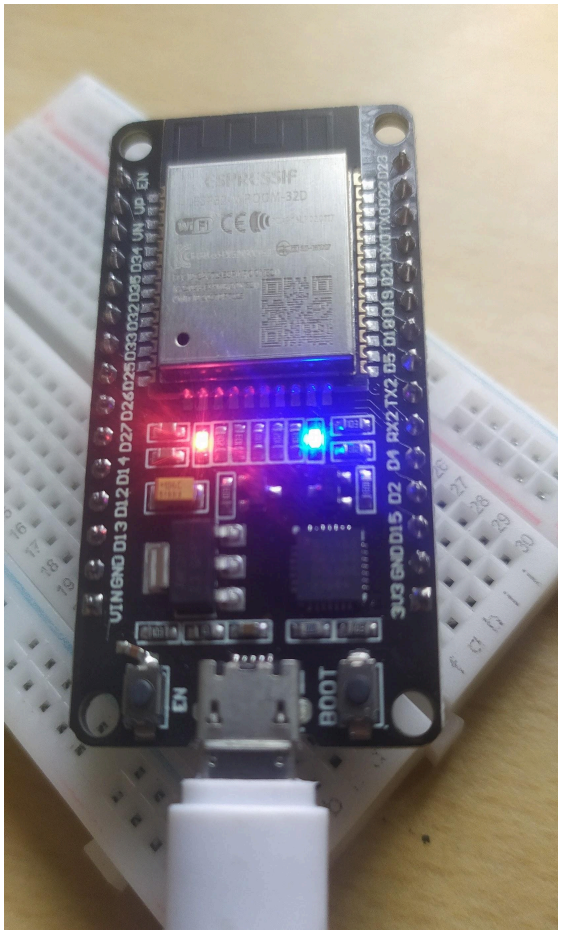

This could be due to 2 problems:

1. Wrong COM port selection: go to tools and change the port, try to upload again.
2. Due to some problem in uploading a new program: press the boot button on ESP32 while uploading the program.

## Output

Internal led switched between on and off with a delay of 1 second.






## Task number: 02

### Task name: Internet clock using ESP32.

**Reference:** https://circuitdigest.com/microcontroller-projects/esp32-internet-clock
For I2C OLED: https://arduinogetstarted.com/tutorials/arduino-oled

**Components used:** ESP32, I2C OLED, breadboard, jumper wires, USB cable.

**Connections:** Connect SDA of oled to D21 of ESP32, SCK to D18. Connect VCC and ground of the OLED to that of ESP32. Connect ESP32 to computer.

## Theory

Connect ESP32 to NTP(network time protocol) and UDP(user datagram protocol) to fetch time from the internet. NTP is used for synchronization of time between systems and data networks. Here we are getting time from the internet and displaying it on OLED.

## Code

Here offset is 19800 sec, because Indian Standard Time is 5hr 30 mins ahead of GMT. And, interval is for how often it should receive a new value, here it is 60 seconds.

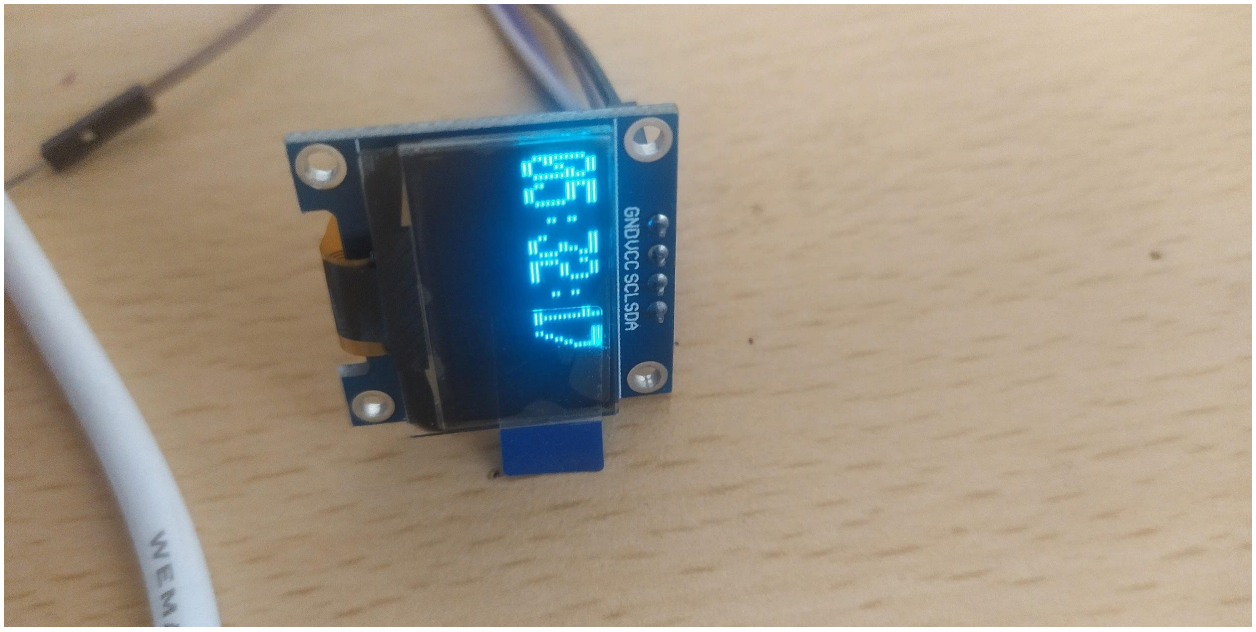
```
#include <WiFi.h>
#include <SPI.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <NTPClient.h>
#include <WiFiUdp.h>

const char* ssid = "ssid";
const char* password = "password";

#define NTP_OFFSET  19800   //  In seconds
#define NTP_INTERVAL 60*1000    // In milliseconds
#define NTP_ADDRESS  "in.pool.ntp.org"  //NTP server for India

Adafruit_SSD1306 display(-1);
WiFiUDP ntpUDP;
NTPClient timeClient(ntpUDP, NTP_ADDRESS, NTP_OFFSET, NTP_INTERVAL);

void initWiFi() {
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);
  Serial.print("Connecting to WiFi...");
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(1000);
  }
  Serial.println(WiFi.localIP());
}
void setup() {
  display.begin();
  Serial.begin(9600);
  Serial.println();
  Serial.println();
  initWiFi();
```

```
  Serial.print("connected");
  timeClient.begin();

  display.begin(SSD1306_SWITCHCAPVCC);

  display.clearDisplay();
  display.setTextColor(WHITE);
  display.setTextSize(2);
  display.setCursor(0,0);
  display.print("  Internet ");
  display.println("  Clock ");
  display.display();
  delay(3000);
}
void loop() {
  timeClient.update();
  String formattedTime = timeClient.getFormattedTime();

  Serial.println(formattedTime);

  display.clearDisplay();
  display.setTextSize(2);
  display.setCursor(0, 0);
  display.println(formattedTime);

  display.display();   // write the buffer to the display
  delay(10);
  delay(100);

}
```

## Problems faced

The clock is printing the wrong time. Every time it starts from around 5:30 only.  We are not sure about the error, but I think it is receiving wrong input from the server, or worse it isn't receiving any input, hence every time it starts from 5:30, by logic of 19800 sec we provided while defining NTP_OFFSET in the code. I'm still working on this.

### Output




The clock first displays “Internet clock”, and then the current time. The same values will be printed on the Serial monitor also.

## Task number: 03

### Task name: controlling GPIO pins using telegram bot.

### Reference:

https://iotdesignpro.com/projects/telegram-bot-with-esp32-control-gpio-pins-through-telegram-chat

**Components used:** ESP32, USB cable. Along with Arduino IDE and Telegram app.

**Connections:** connect ESP32 to computer using USB cable. Create a telegram bot to monitor the GPIO pins of ESP32.

### Theory

Here in this task, we are controlling an led connected to ESP32 by sending commands to telegram bot.

**Creating a bot:** go to your telegram account, search for **botfather** and click on it. This helps us to create, delete and manage our bots. Click on **start**, type **/newbot**, give name and username to the bot. Once a bot is successfully created, we will receive a message with a **link** to access it and a **token**(unique ID) which helps us to communicate with the bot.
**Getting Telegram chat ID:** we use this to avoid unauthorized access to the bot. This helps to uniquely identify a use or chat or group. Go to **IDBot**, type **/getid**, it sends an ID, save this for later use.
Also install the necessary bot library for telegram bot.

## Code

```
#include <WiFi.h>
#include <WiFiClientSecure.h>
#include <UniversalTelegramBot.h>
#include <ArduinoJson.h>

const char* ssid = "Your_SSID";
const char* password = "Your_pass";

// Initialize Telegram BOT
#define BOTtoken "Your_Bot_Token"
#define CHAT_ID "Your_chat_id"

WiFiClientSecure client;
UniversalTelegramBot bot(BOTtoken, client);
// Checks for new messages every 1 second.
int botRequestDelay = 1000;
unsigned long lastTimeBotRan;
const int ledPin = 2;
bool ledState = LOW;

// Handle new receive messages
void handleNewMessages(int numNewMessages) {
  Serial.println("handleNewMessages");
  Serial.println(String(numNewMessages));
  for (int i=0; i<numNewMessages; i++) {
    // Chat id of the requester
    String chat_id = String(bot.messages[i].chat_id);
    if (chat_id != CHAT_ID){
      bot.sendMessage(chat_id, "Unknown user", "");
      continue;
    }

    // Print the received message
```

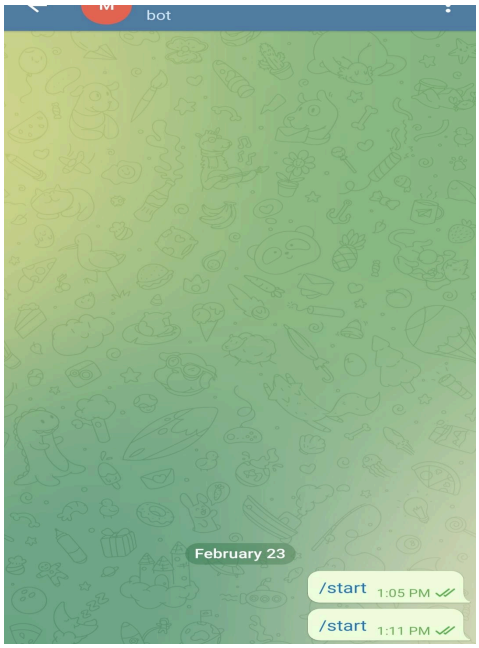
```
    String text = bot.messages[i].text;
    Serial.println(text);
    String from_name = bot.messages[i].from_name;

// commands
    if (text == "/start") {
      String welcome = "Welcome, " + from_name + ".\n";
      welcome += "Use the following commands to control your outputs.\n\n";
      welcome += "/led_on to turn GPIO ON \n";
      welcome += "/led_off to turn GPIO OFF \n";
      welcome += "/state to request current GPIO state \n";
      bot.sendMessage(chat_id, welcome, "");
    }
    if (text == "/led_on") {
      bot.sendMessage(chat_id, "LED state set to ON", "");
      ledState = HIGH;
      digitalWrite(ledPin, ledState);
    }
    if (text == "/led_off") {
      bot.sendMessage(chat_id, "LED state set to OFF", "");
      ledState = LOW;
      digitalWrite(ledPin, ledState);
    }
    if (text == "/state") {
      if (digitalRead(ledPin)){
        bot.sendMessage(chat_id, "LED is ON", "");
      }
      else{
        bot.sendMessage(chat_id, "LED is OFF", "");
      }
    }
}
}
void setup() {
  Serial.begin(115200);
  client.setInsecure();
  pinMode(ledPin, OUTPUT);
  digitalWrite(ledPin, ledState);

  // Connect to Wi-Fi
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);
  WiFi.setSleep(false);
  while (WiFi.status() != WL_CONNECTED) {
```

```
    delay(1000);
    Serial.println("Connecting to WiFi..");
  }
  // Print ESP32 Local IP Address
  Serial.println(WiFi.localIP());
}
void loop() {
  if (millis() > lastTimeBotRan + botRequestDelay)  {
    int numNewMessages = bot.getUpdates(bot.last_message_received + 1);
    while(numNewMessages) {
      Serial.println("got response");
      handleNewMessages(numNewMessages);
      numNewMessages = bot.getUpdates(bot.last_message_received + 1);
    }
    lastTimeBotRan = millis();
  }
}
```

## Problems faced

The bot isn’t responding to any of the commands, I'm still working on this error.

## Output




There was no change or response visible on LED while a command was given. No response from bot also.

# Task number: 04

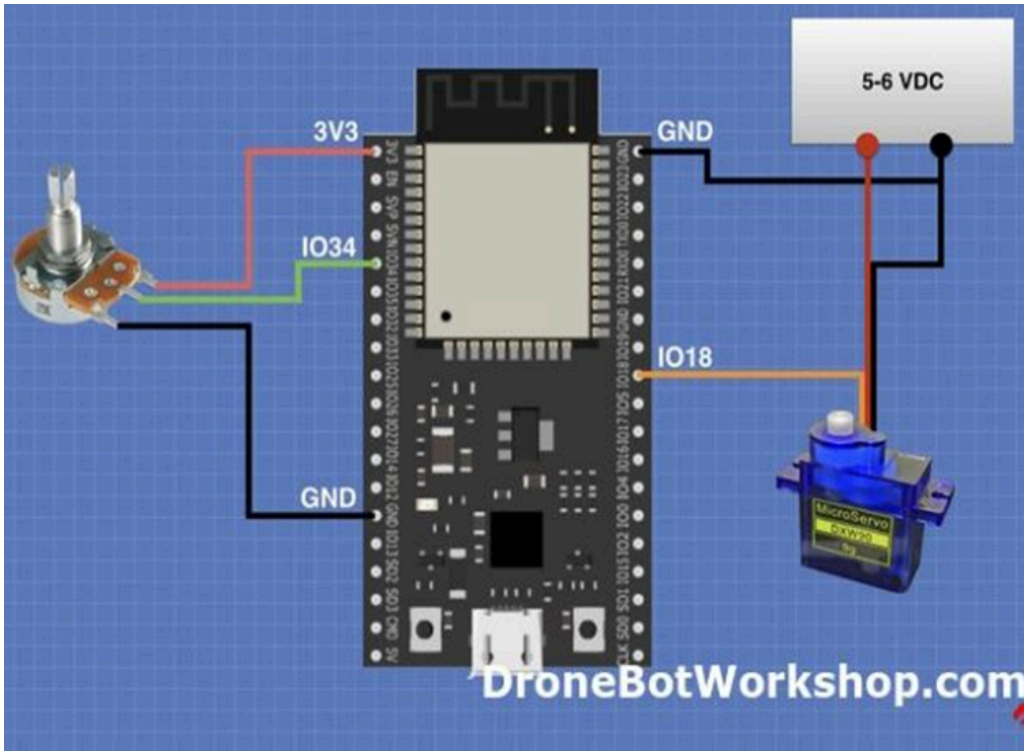
## Task name: Controlling servo motor with ESP32.

**Reference:** https://dronebotworkshop.com/esp32-servo/

**Components used:** micro servo motor, ESP32, potentiometer, servo power supply, DC jack, jumper wires, breadboard.

## Connections

1. **Sweep:** connect ground and VCC wires of servo motor to ground and VCC of ESP32 respectively. Next, connect the control input pin of the servo to GPIO18 of the ESP32.
2. **Knob:** connect middle wire of potentiometer to GPIO34 of ESP32, and connect one of the other two to ground and 3V pins of ESP32. Connect the ground of the servo motor and ground of the ESP32 to the ground pins of the DC power supply. Connect control pin of servo to GPIO18 of ESP32.




3. **Web remote controlled:** connect ground and VCC wires of servo motor to ground and VCC of ESP32 respectively. Next, connect the control input pin of the servo to GPIO18 of the ESP32. Once the code is uploaded, it prints an IP address on the serial monitor. Copy this and open it on any device connected to the internet, now monitor the movement of the servo motor shaft based on the slider.

## Theory

Servo motors have an inbuilt servomechanism with a feedback loop to allow precise positioning of the motor shaft. It has 3 pins, brown for ground, red for VCC and orange for PWM input, we use this pin to provide signal hence control the movement of the motor shaft. We are using ESP32 to provide PWM input to control input of the servo.
In the knob sketch, we use a potentiometer to position the motor shaft. It moves as the position of the potentiometer is changed.
A web page was created with a slider on it, and is managed using the IP address.

## Code

### 1. Sweep:

```
#include <Servo.h>
Servo myservo;
Int pos=0;
Void setup(){
        for(pos=0; pos<=180; pos+=1){
                myservo.write(pos);
                delay(15);
        }
        for(pos=180; pos>=0; pos-=1){
                myservo.write(pos);
                delay(15);
        }
}
```

### 2. Knob:

```
#include <ESP32Servo.h>

Servo myservo;

int servoPin = 18;      //this is pin we give as control to servo
int potPin = 34;
int ADC_Max = 4096;

int val;    // variable to read the value from the analog pin

void setup()
```

```
{
        ESP32PWM::allocateTimer(0);
        ESP32PWM::allocateTimer(1);
        ESP32PWM::allocateTimer(2);
        ESP32PWM::allocateTimer(3);
  myservo.setPeriodHertz(50);// Standard 50hz servo
  myservo.attach(servoPin, 500, 2400);   // attaches the servo on pin 18 to the servo object
                                  }

void loop() {
  val = analogRead(potPin);
  val = map(val, 0, ADC_Max, 0, 180);
  myservo.write(val);                  // set the servo position according to the scaled value
  delay(200);
}
```

### 3. Web controlled:

```
#include <WiFi.h>
#include <ESP32Servo.h>

Servo myservo;

static const int servoPin = 18;

const char* ssid     = "Marvel-Guest";
const char* password = "";

// Web server on port 80 (http)
WiFiServer server(80);

String header;  //stores http request

String valueString = String(5);
int pos1 = 0;
int pos2 = 0;

unsigned long currentTime = millis(); //time right now
unsigned long previousTime = 0;       //time before
const long timeoutTime = 2000;        //time in milliseconds

void setup() {
```

```
  // Allow allocation of all timers for servo library
  ESP32PWM::allocateTimer(0);
  ESP32PWM::allocateTimer(1);
  ESP32PWM::allocateTimer(2);
  ESP32PWM::allocateTimer(3);

  myservo.setPeriodHertz(50);  //PWM frequency

  myservo.attach(servoPin,500, 2400);    //500 is the minimum position and 2400 is the
//maximum position

  Serial.begin(115200);

  Serial.print("Connecting to ");
  Serial.println(ssid);
  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());
  server.begin();
}

void loop(){

  // Listen for incoming clients
  WiFiClient client = server.available();

  // Client Connected
  if (client) {
    // Set timer references
    currentTime = millis();
    previousTime = currentTime;

    Serial.println("New Client.");

    // String to hold data from client
    String currentLine = "";
```

```
   while (client.connected() && currentTime - previousTime <= timeoutTime) {
     currentTime = millis();
     if (client.available()) {             // if there's bytes to read from the client,
       char c = client.read();             // read a byte, next
       Serial.write(c);                    // print it on the serial monitor
       header += c;
       if (c == '\n') {                    // if the byte is a newline character
         // if the current line is blank, you get two newline characters in a row.
         // that's the end of the client HTTP request, so send a response:

         if (currentLine.length() == 0) {

           client.println("HTTP/1.1 200 OK");
           client.println("Content-type:text/html");
           client.println("Connection: close");
           client.println();

           // Display the HTML web page

           // HTML Header
           client.println("<!DOCTYPE html><html>");
           client.println("<head><meta name=\"viewport\" content=\"width=device-width,
initial-scale=1\">");
           client.println("<link rel=\"icon\" href=\"data:,\">");

           client.println("<style>body { text-align: center; font-family: \"Trebuchet MS\", Arial;
margin-left:auto; margin-right:auto; }");
           client.println(".slider { -webkit-appearance: none; width: 300px; height: 25px;
border-radius: 10px; background: #ffffff; outline: none;  opacity: 0.7;-webkit-transition: .2s;
transition: opacity .2s;}");
           client.println(".slider::-webkit-slider-thumb {-webkit-appearance: none; appearance:
none; width: 35px; height: 35px; border-radius: 50%; background: #ff3410; cursor: pointer;
}</style>");

           // Get JQuery
           client.println("<script
src=\"https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js\"></script>");

           // Page title
           client.println("</head><body style=\"background-color:#70cfff;\"><h1
style=\"color:#ff3410;\">Servo Control</h1>");

           // Position display
```

```
          client.println("<h2 style=\"color:#ffffff;\">Position: <span
id=\"servoPos\"></span>&#176;</h2>");

          // Slider control
          client.println("<input type=\"range\" min=\"0\" max=\"180\" class=\"slider\"
id=\"servoSlider\" onchange=\"servo(this.value)\" value=\""+valueString+"\"/>");

          // Javascript
          client.println("<script>var slider = document.getElementById(\"servoSlider\");");
          client.println("var servoP = document.getElementById(\"servoPos\"); servoP.innerHTML
= slider.value;");
          client.println("slider.oninput = function() { slider.value = this.value; servoP.innerHTML =
this.value; }");
          client.println("$.ajaxSetup({timeout:1000}); function servo(pos) { ");
          client.println("$.get(\"/?value=\" + pos + \"&\"); {Connection: close};}</script>");

          // End page
          client.println("</body></html>");

          // GET data
          if(header.indexOf("GET /?value=")>=0) {
            pos1 = header.indexOf('=');
            pos2 = header.indexOf('&');

            // String with motor position
            valueString = header.substring(pos1+1, pos2);

            // Move servo into this new position
            myservo.write(valueString.toInt());

            // Print value to serial monitor
            Serial.print("Val =");
            Serial.println(valueString);
          }
          // The HTTP response ends with another blank line
          client.println();

          // Break out of the while loop
          break;

        } else {
          // New line is received, clear currentLine
          currentLine = "";
        }
```

```
        } else if (c != '\r') {  // if you got anything else but a carriage return character,
          currentLine += c;      // add it to the end of the currentLine
        }
      }
    }
    // Clear the header variable
    header = "";
    client.stop();
    Serial.println("Client disconnected.");
    Serial.println("");
  }
}
```

## Problems faced

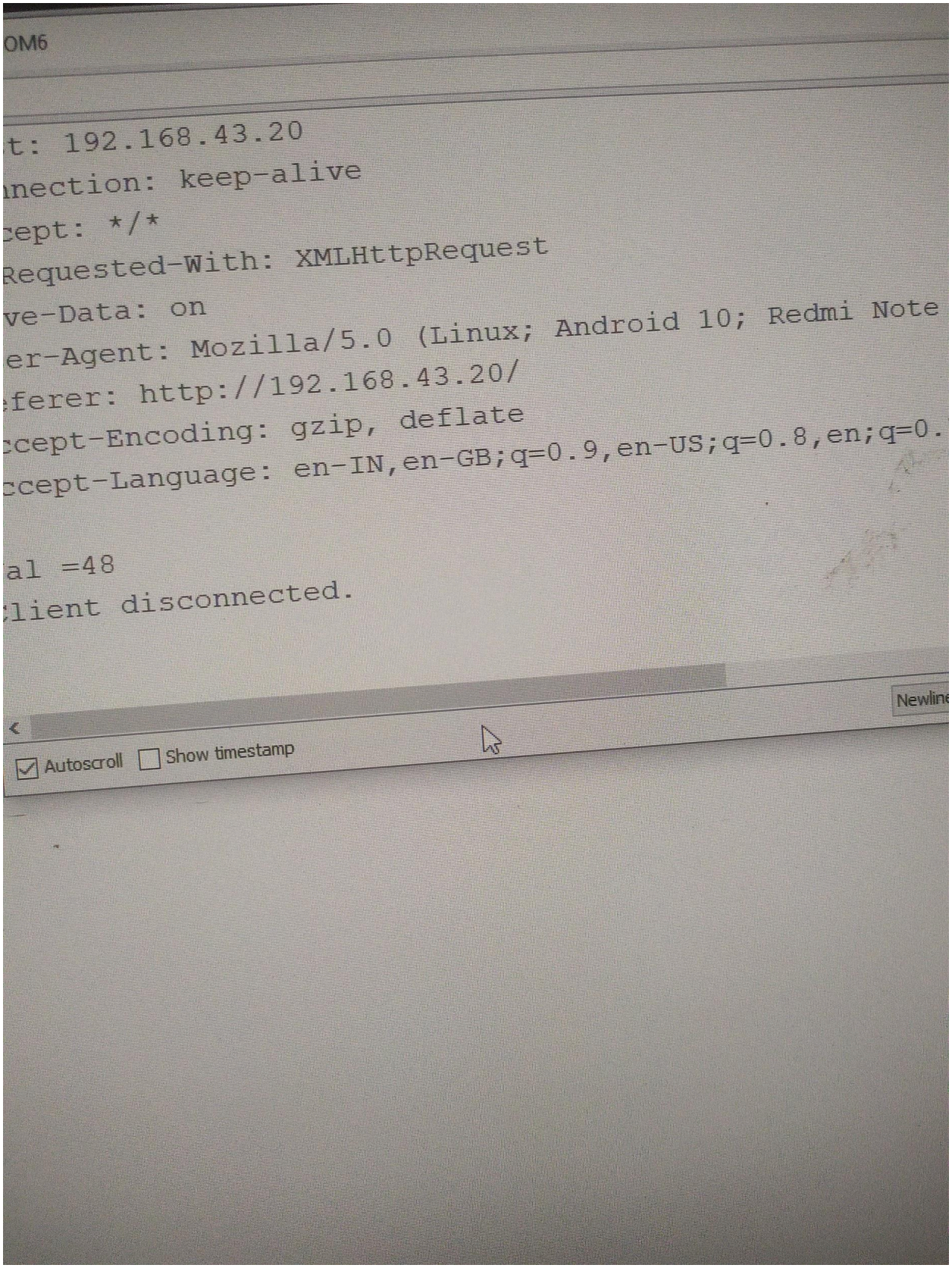
Servo was rotating full 360 degrees on its own, it was because of some loose connection, when connections were rechecked it worked as directed in the code.
When the server page was opened on the computer, there was no movement on the servo. Opened the same on mobile phone, it worked as the slider was slid.

## Output

Servo made a proper 180 degree rotation and then did the same in reverse direction.
https://drive.google.com/open?id=1DGTOdFvdRKjr1z-95rhBGvuoUigH5_NF&authuser=0

Servo motor shaft changed its position as the potentiometer value was changed.
https://drive.google.com/open?id=1-WGQ87_-YVjjq_2hq1i2zRRYELsAY4VW&authuser=0

Motor shaft changed its position as directed on the web page.



WebControl.mp4

# Task number: 05

## Task name: sending and receiving data from the cloud using ESP32.

### Reference:
https://iotdesignpro.com/projects/how-to-send-data-to-thingspeak-cloud-using-esp32

**Components used:** ESP32, USB cable.

**Connections:** connect ESP32 to computer. Create an account on https://thingspeak.com/, create a new channel.
Go to API keys, copy the write API key, and store it.

### Theory

In this task we are going to measure temperature and hall voltage using ESP32, and observe its plot versus time on our thingspeak account.
**Hall effect sensor:** detects the presence and measures the magnitude of magnetic effect using Hall effect. Hall effect principle is, “output voltage is directly proportional to strength of the field.” Principle of working of this sensor: current is applied to a thin strip of metal. In the presence of magnetic field, perpendicular to the direction of current, charge carriers are deflected by Lorentz force, producing a difference in voltage between 2 sides of strips.”

### Code

In the first section of the code, we are measuring temperature by including temperature_sens_read() library. Next, we have added the necessary code to connect to Wi-Fi.

```
#ifdef __cplusplus
extern "C" {
#endif
uint8_t temprature_sens_read();
#ifdef __cplusplus
}
#endif
uint8_t temprature_sens_read();
#include <WiFi.h>
```

```
#include <HTTPClient.h>

// Set our wifi name and password
const char* ssid = "Marvel-Guest";
const char* password = "";

// Your thingspeak channel url with api_key query
String serverName = "https://api.thingspeak.com/update?api_key=your_api_key"; //here replace
your_api_key with ur actual API key

// Assign some variables to allow us read and send data every minute
unsigned long lastTime = 0;
unsigned long timerDelay = 60000;


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


  WiFi.begin(ssid, password);
  Serial.println("Connecting"); // Print our status to the serial monitor
  // Wait for wifi to connect
  while(WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.print("Connected to WiFi network with IP Address: ");
  Serial.println(WiFi.localIP());
}


void loop() {

  if ((millis() - lastTime) > timerDelay) { // Check if its been a minute
    if(WiFi.status()== WL_CONNECTED){ // Check to make sure wifi is still connected
      sendData(1, 2, 3, 4); // Call the sendData function defined below
    }
    else {
      Serial.println("WiFi Disconnected");
    }
    lastTime = millis();
  }
}
```

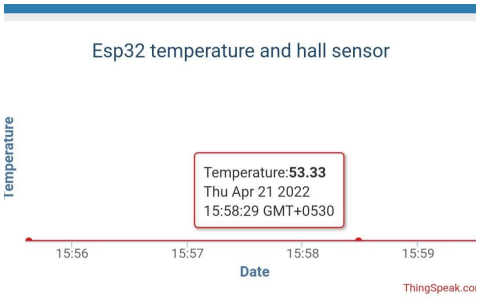
```
void sendData(double temp, double pres, double alt, double hum){
   int h = 0;
float t =0;

h = hallRead();
t = ((temprature_sens_read()-32)/1.8);
  HTTPClient http; // Initialize our HTTP client


  String url = serverName + "&field1=" + String(h) + "&field2=" + String(t) + "&field3=" + alt +
"&field4=" + hum; // Define our entire url

  http.begin(url.c_str()); // Initialize our HTTP request

  int httpResponseCode = http.GET(); // Send HTTP request

  if (httpResponseCode > 0){ // Check for good HTTP status code
    Serial.print("HTTP Response code: ");
    Serial.println(httpResponseCode);
  }else{
    Serial.print("Error code: ");
    Serial.println(httpResponseCode);
  }
  http.end();
}
```

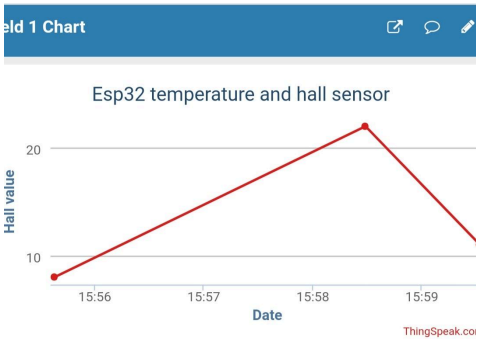
## Problems faced

The values are printed on the serial monitor, but there is change on graphs in my channel on Thingspeak. There is some problem, due to which either ESP32 is not sending or the api is not receiving the values. Made some changes in code, this plotted perfectly.

## Output

The temperature and hall values were successfully printed on the serial monitor.
The graph was plotted on the thingspeak account page.



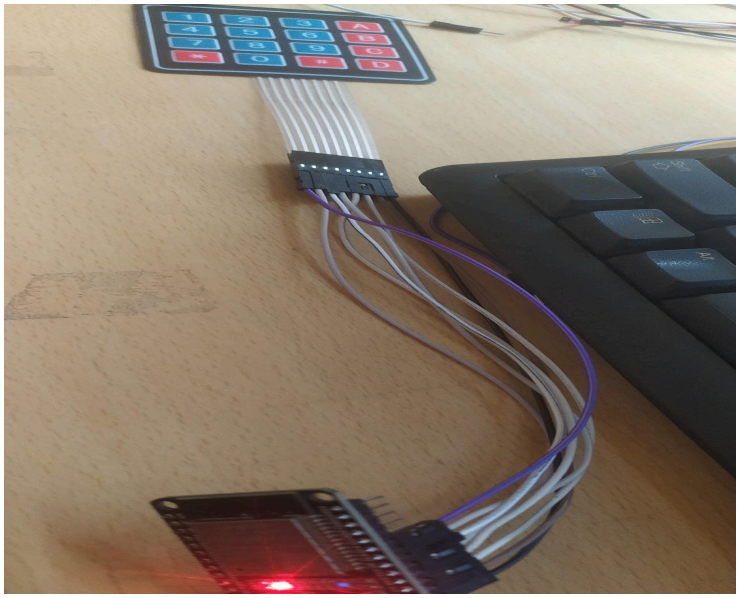

# Task number: 06

**Task name: Interfacing a keypad with ESP32.**

**Reference:** https://esp32io.com/tutorials/esp32-keypad

**Components used:** ESP32, 4x4 keypad, jumper wires.

**Connections:** connect ESP32 to computer using USB cable. Connect the pin r1 to GIOP19, r2 to GIOP18, r3 to GIOP05, r4 to GIOP17, c1 to GIOP16, c2 to GIOP04, c3 to GIOP0 and c4 to GIOP02.



## Theory

Keypad is a bunch of buttons arranged in a matrix, where each button represents a key. We access keys using pins provided.
We are using a keypad to authenticate. Here we have pre-defined a password in code, if a user types the same, access is granted else not.

## Code

1. **Printing key pressed on a serial monitor**

```
#include <Keypad.h>
#define ROW_NUM     4
#define COLUMN_NUM  4

char keys[ROW_NUM][COLUMN_NUM] = {
  {'1', '2', '3', 'A'},
  {'4', '5', '6', 'B'},
  {'7', '8', '9', 'C'},
  {'*', '0', '#', 'D'}
};

byte pin_rows[ROW_NUM]      = {19, 18, 5, 17}; // GIOP19, GIOP18, GIOP5, GIOP17 connect to the row pins
```

```
byte pin_column[COLUMN_NUM] = {16, 4, 0, 2};   //connect to the column pins

Keypad keypad = Keypad( makeKeymap(keys), pin_rows, pin_column, ROW_NUM,
COLUMN_NUM );

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

void loop() {
  char key = keypad.getKey();

  if (key) {
    Serial.println(key);
  }
}
```

2. **Interfacing password**

```
#include <Keypad.h>

#define ROW_NUM     4
#define COLUMN_NUM  4

char keys[ROW_NUM][COLUMN_NUM] = {
  {'1', '2', '3', ‘A’},
  {'4', '5', '6', ‘B’},
  {'7', '8', '9', ‘C’},
  {'*', '0', '#', ‘D’}
};

byte pin_rows[ROW_NUM] = {19, 18, 5, 17}; // connect to the row pins
byte pin_column[COLUMN_NUM] = {16,  4, 0, 2};  // connect to the column pins

Keypad keypad = Keypad( makeKeymap(keys), pin_rows, pin_column, ROW_NUM,
COLUMN_NUM );

const String password = "7890"; // change your password here
String input_password;
```

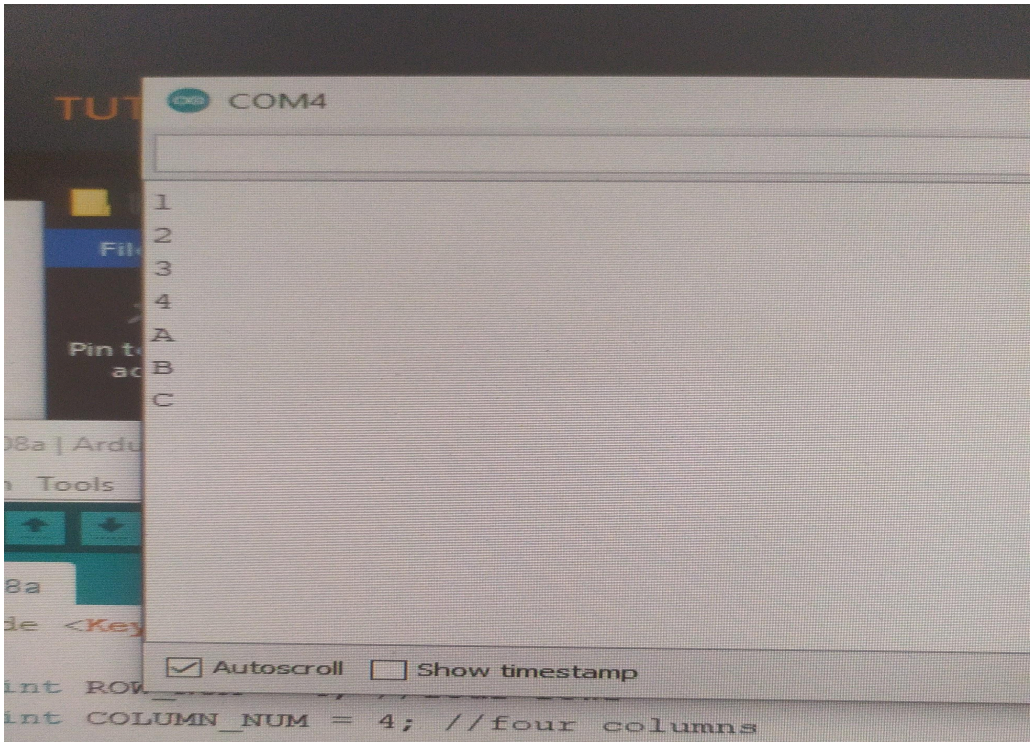
```
void setup() {
  Serial.begin(9600);
  input_password.reserve(32); // maximum input characters is 33, change if needed
}

void loop() {
  char key = keypad.getKey();

  if (key) {
    Serial.println(key);

    if (key == '*') {
      input_password = ""; // clear input password
    } else if (key == '#') {
      if (password == input_password) {
        Serial.println("The password is correct, ACCESS GRANTED!");


      } else {
        Serial.println("The password is incorrect, ACCESS DENIED!");
      }

      input_password = ""; // clear input password
    } else {
      input_password += key; // append new character to input password string
    }
  }
}
```

**Problems faced:** None for this task.

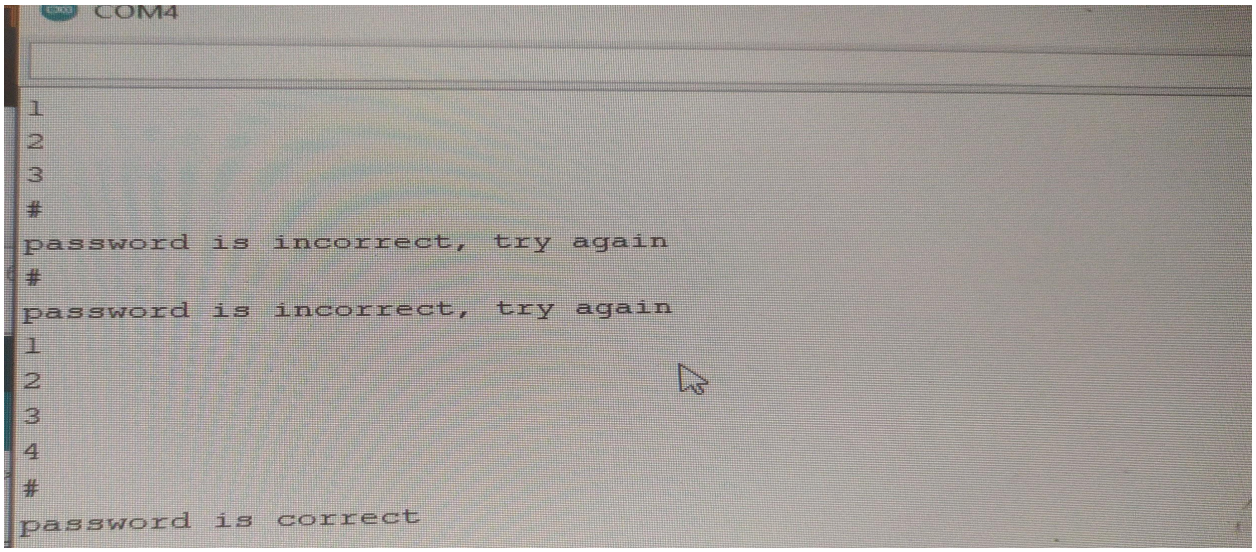
## Output

### Trying to print the key pressed

Key pressed on the keypad was accurately printed on the serial monitor.




**Verifying password: password was correctly verified.**



# Task number: 07

## Task name: Obtaining data from ultrasonic sensor using ESP32.

**Reference:** https://randomnerdtutorials.com/esp32-hc-sr04-ultrasonic-arduino/

**Components used:** ESP32, ultrasonic sensor,oled display, jumper wires and breadboard.

**Connections:** Here, we connect the trig pin of the sensor to GPIO-5 and echo pin to GPIO-18. Connect Vcc and ground of ESP32 to that of the sensor.
If we wish to see output on oled then, along with the above connections, connect the VCC and ground of it to that of ESP32. Connect SCK and SDA pins of oled to the same pins of ESP32 i.e. GPIO-22 and GPIO-21 respectively.

## Theory

Here we are printing and observing the distance measured by the HC-SR04 sensor connected to ESP32.
**HC-SR04 Ultrasonic sensor:** uses sonar to find distance to an object, can read from 2cm to 400cm. This has 4 pinouts, *VCC*, *ground*, *trigger* input and *echo* output.
The transmitter at trig pin, transmits high frequency sound waves,this sound traveling through air bounces back when it finds an object. The ultrasonic receiver at the echo pin, receives this bounced back signal.
Now, *distance=((velocity of sound in air)*duration)/2*
Here, *duration* is the interval between the sound wave emitted and detected.

## Code

### With OLED:

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

//declaring the dimensions of serial monitor
#define ScreenWidth 126
#define ScreenHeight  64

Adafruit_SSD1306 display(ScreenWidth, ScreenHeight, &Wire, -1);

const int trig_pin = 5;
const int echo_pin = 18;
```

```
#define Sound_speed 0.034
#define cm_to_inch 0.393701

long duration;
int distanceCm;
int distanceInch;

void setup() {
  Serial.begin(115200);
  pinMode(trig_pin, OUTPUT);
  pinMode(echo_pin, INPUT);

/*in the below line, 0x3C indicates address at which oled is found, if it is at different address,
check for it, and feed it in place of 0x3C */

  if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
    Serial.println(F("SSD1306 allocation failed"));
    for(;;); //when it is unable to connect to the serial monitor, this for loop keeps running.
  }
  delay(500);

  display.clearDisplay();
  display.setTextSize(2);
  display.setTextColor(WHITE);
}

void loop() {
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  duration = pulseIn(echoPin, HIGH);

  distanceCm = duration * Sound_Speed/2;

  distanceInch = distanceCm * cm_to_inch;

  // here we are printing the distance on the Serial Monitor
  Serial.print("Distance (cm): ");
  Serial.println(distanceCm);
  Serial.print("Distance (inch): ");
  Serial.println(distanceInch);
```

```
  display.clearDisplay();
  display.setCursor(0, 25);

  //Displaying distance in cm
  display.print(distanceCm);
  display.print(" cm");

  // Displaying distance in inches, use only if u want to observe this output
  /* display.print(distanceInch);
  display.print(" in");*/

  display.display();

  delay(500);
}
```

The above code is for connections with OLED, if not required just remove the unnecessary part of the code, and it works fine.

### With no OLED:

```
const int trigPin = 5;
const int echoPin = 18;

//define sound speed in cm/uS
#define SOUND_SPEED 0.034
#define CM_TO_INCH 0.393701

long duration;
float distanceCm;
float distanceInch;

void setup() {
  Serial.begin(115200);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
}

void loop() {
  digitalWrite(trigPin, LOW); //this clears the trigger pin
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
```

```
  digitalWrite(trigPin, LOW);

  // Reads the echoPin, returns the sound wave travel time in microseconds
  duration = pulseIn(echoPin, HIGH);

  distanceCm = duration * SOUND_SPEED/2;

  // Convert to inches
  distanceInch = distanceCm * CM_TO_INCH;

  // Prints the distance on the Serial Monitor
  Serial.print("Distance (cm): ");
  Serial.println(distanceCm);
  Serial.print("Distance (inch): ");
  Serial.println(distanceInch);

  delay(1000);
}
```

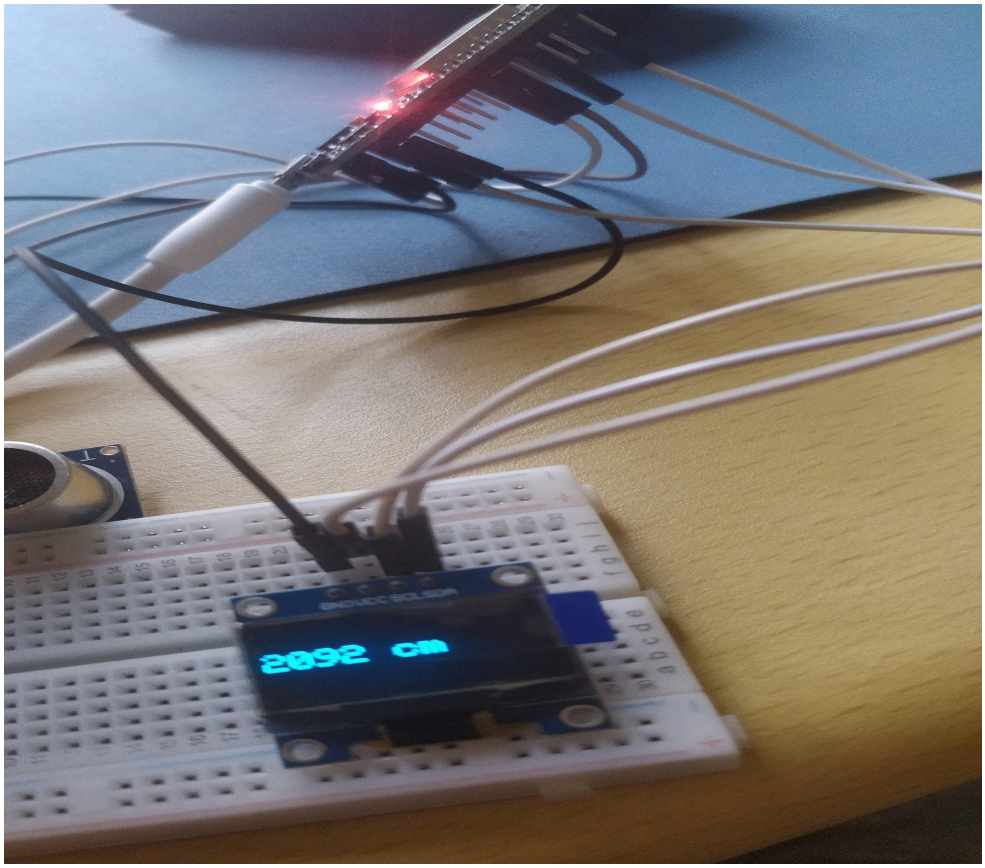
## Procedure

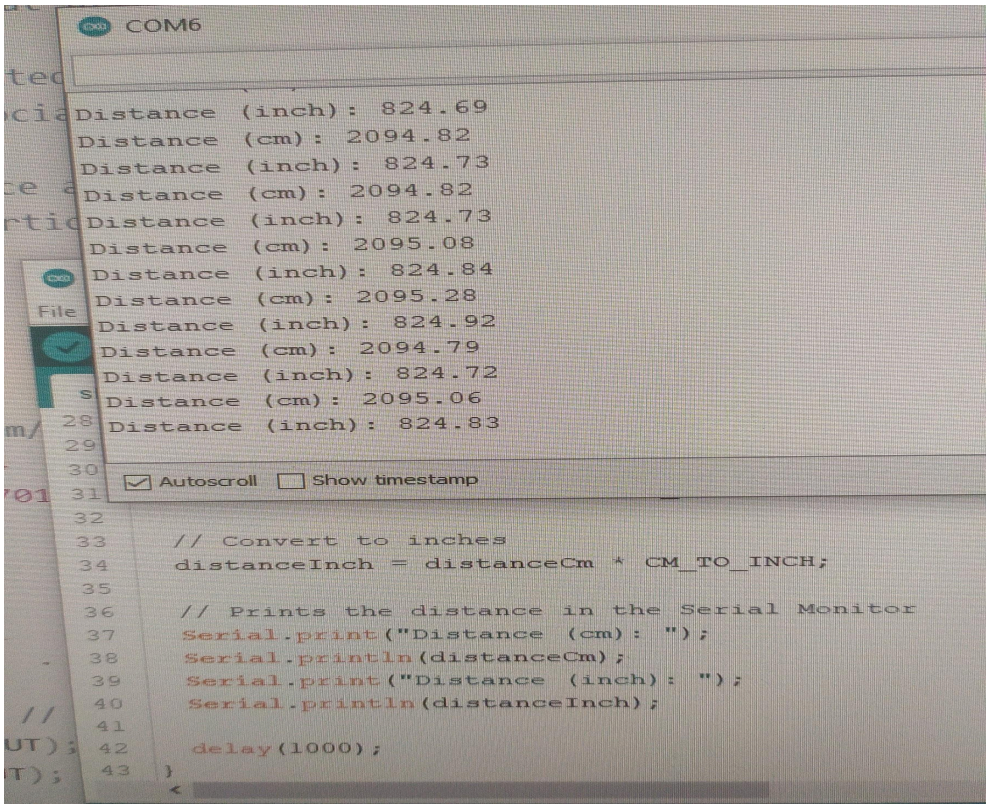
1. Create a sketch for this project on arduino, write and verify the above code.
2. Make the necessary connections between ESP32, OLED and Sensor.
3. Connect this circuit to the computer through USB cable.
4. Go to tools, select appropriate board names and com ports.
5. Upload the code to ESP32.
6. Open the serial monitor to see the output.
7. Also observe the output on OLED if connected.

## **Problems faced:** none for this task**.**

## Output

The ultrasonic measured the distance from the object, distance was successfully printed on serial monitor and also the OLED display. Here it has measured how far is the ceiling above the table.


2092 cm


COM6
Distance (inch): 824.69
Distance (cm): 2094.82
Distance (inch): 824.73
Distance (cm): 2094.82
Distance (inch): 824.73
Distance (cm): 2095.08
Distance (inch): 824.84
Distance (cm): 2095.28
Distance (inch): 824.92
Distance (cm): 2094.79
Distance (inch): 824.72
Distance (cm): 2095.06
Distance (inch): 824.83
Autoscroll
Show timestamp
// Convert to inches
distanceInch = distanceCm * CM_TO_INCH;
// Prints the distance in the Serial Monitor
Serial.print("Distance (cm): ");
Serial.println(distanceCm);
Serial.print("Distance (inch): ");
Serial.println(distanceInch);
delay(1000);
}

# Task number: 08

## Task name: Installing OS on Raspberry pi4

**Reference:** https://www.circuitbasics.com/how-to-install-the-raspberry-pi-operating-system/
For basics: https://www.circuitbasics.com/getting-started-with-the-raspberry-pi/

**Components used:** microSD card, Raspberry pi board, HDMI cable, USB keyboard, mouse, power supply for board.

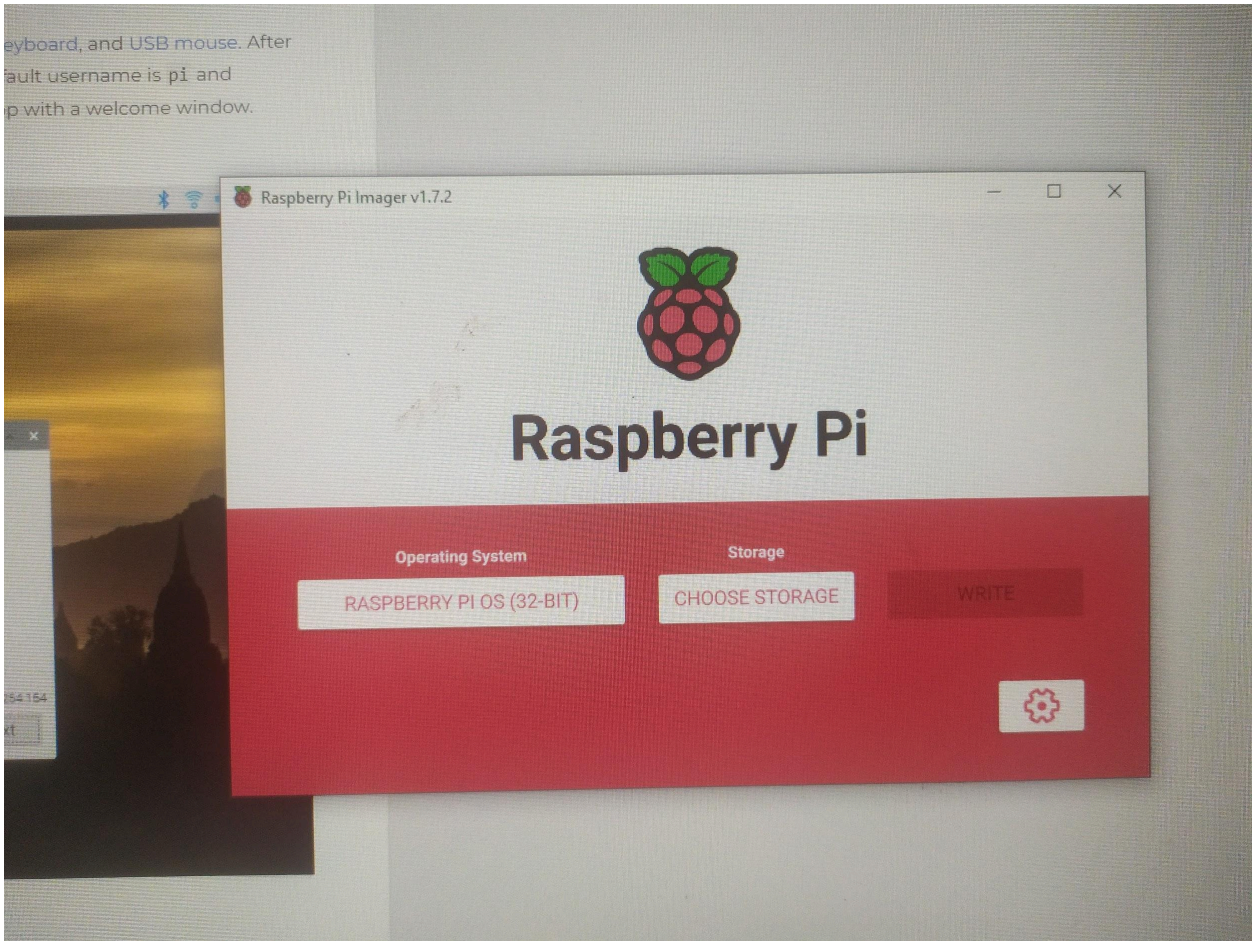
## Connections

1. Insert microSD card to CPU or laptop using a card reader.
2. Format the SD card, if necessary.
3. Download OS(Raspberry pi imager)  into the microSD from Raspberry Pi OS – Raspberry Pi
4. After completing download, run the executable file and install the software.
5. Choose OS- select the first option, 32 bit(recommended)
6. Select the appropriate microSD card.
7. Write OS image file to microSD.
8. Once writing and verifying completes, enter continue.
9. Go to file explorer, open the disk folder.
10. Search for config file, uncomment the *#hdmi_force_hotplug=1* line. Save the change, and close the file.
11. Insert this SD card into a Raspberry pi board, connect HDMI cable to desktop, connect keyboard and mouse.
12. Username is pi, and the password is raspberry.
13. Finish setup.

## Problems faced

- Some microSD cards were corrupted, so there was a problem with formatting them, so I just changed the SD card.
- Monitor going to sleep, because of no signal: make sure power is supplied for the board. Also, check if the hdmi line in config file (*#hdmi_force_hotplug=1* line) is uncommented.

### Output




## Task number: 09

### Task name: Control LED with Raspberry pi using python.

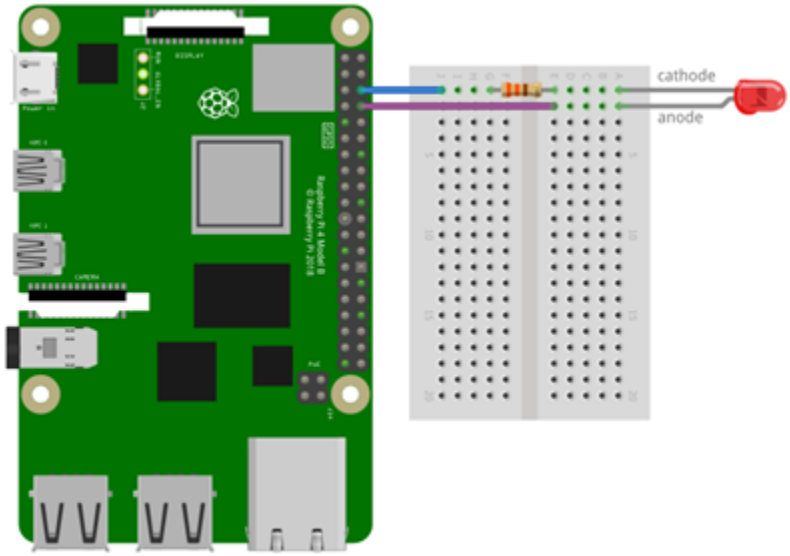
**Reference:** https://www.circuitbasics.com/how-to-control-led-using-raspberry-pi-and-python/
To set up: https://www.circuitbasics.com/how-to-install-the-raspberry-pi-operating-system/

**Components used:** microSD card, Raspberry pi board, HDMI cable, USB keyboard, mouse, power supply for board.

### Connections

Insert a microSD card to the Raspberry pi board, connect it to monitor using HDMI cable. Also connect the mouse and keyboard.

Connect the cathode(shorter leg) of LED to GPIO pin 14 of board through a current limiting resistor. Connect the anode(longer leg) of the LED to pin below that.




Next, once open the raspberry pi terminal and type nano led.py to create a python file. This creates and opens the file in nano text editor, next step write the python code on the file. Once done, save and close the file.
Open Raspberry pi terminal, to run the program type sudo python led.py, and press enter.

## Theory

To write python code to control an LED with Raspberry pi, first import *RPI.GPIO* library to manage GPIO pins. And also import *time* module to manage delay using sleep function.
To avoid all the warnings, set it to false, set pin 14 as output. Once code enters the while loop, first LED is set to high, it prints LED id ON on the terminal, after delay of 1 second, it is set to low, again delay of 1 second. This while loop continues till we stop it with a command or disconnect the circuit.

## Code

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
GPIO.setup(14,GPIO.OUT)

while True:
        GPIO.output(14,GPIO.HIGH)
        print "LED is ON"
```

```
time.sleep(1)


GPIO.output(14,GPIO.LOW)
print "LED is OFF"
time.sleep(1)
```

## Problems faced

Firstly, there were errors due to indentation and syntax. Next there was no response in the LED, as it was connected wrongly.
Some ways to avoid problems:

- Indentation: we all know python works only with proper indentation, while writing code on nano file, proper indentation should be provided.
- Make sure to provide proper power supply.
- Make sure to check that all the connections are proper.

## Output

LED blinks with a delay of 1 second.