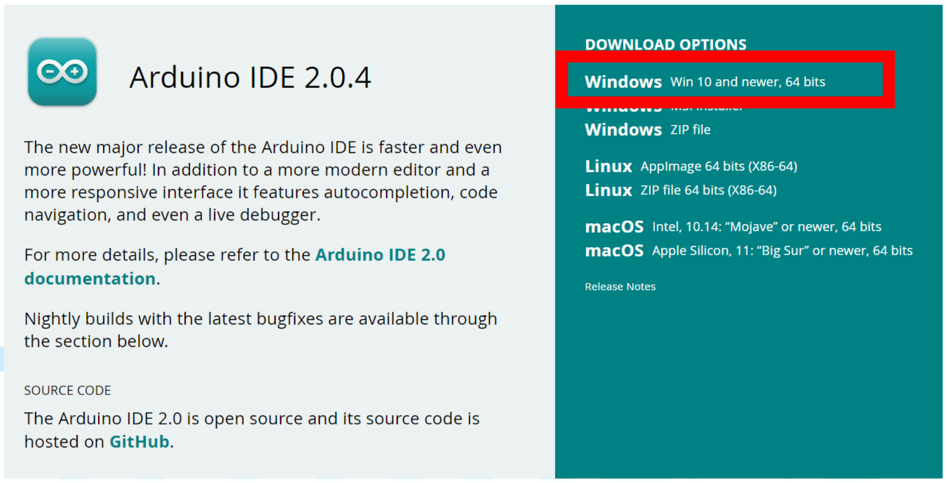
HANDS ON WORKSHEET

DAY I: Fundamentals of ESP32, sensors and actuators.

Prerequisite: Download a github repo from following link and extract its contents on desktop.

1. Installing Arduino IDE, ESP32 Board and Libraries.

Step 1: Visit Arduino website to download Arduino IDE 2.0.



Arduino IDE 2.0.4

The new major release of the Arduino IDE is faster and even more powerful! In addition to a more modern editor and a more responsive interface it features autocompletion, code navigation, and even a live debugger.

For more details, please refer to the Arduino IDE 2.0 documentation.

Nightly builds with the latest bugfixes are available through the section below.

SOURCE CODE

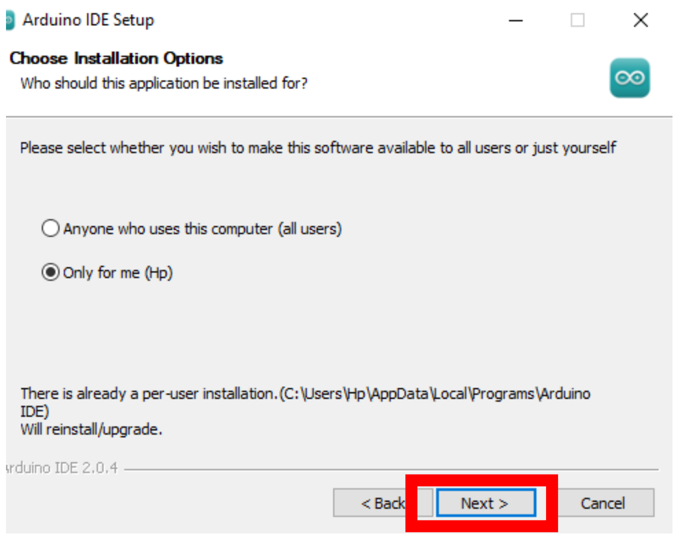
The Arduino IDE 2.0 is open source and its source code is hosted on GitHub.

DOWNLOAD OPTIONS

Windows Win 10 and newer, 64 bits
Windows ZIP file
Linux AppImage 64 bits (X86-64)
Linux ZIP file 64 bits (X86-64)
macOS Intel, 10.14: "Mojave" or newer, 64 bits
macOS Apple Silicon, 11: "Big Sur" or newer, 64 bits
Release Notes

https://www.arduino.cc/en/software

Step 2: Open the downloaded file and proceed to installation.



Arduino IDE Setup

Choose Installation Options

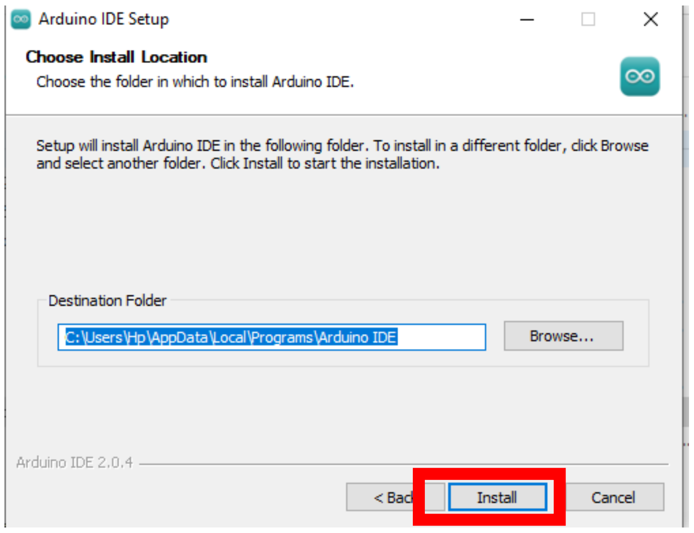
Who should this application be installed for?

Please select whether you wish to make this software available to all users or just yourself

( ) Anyone who uses this computer (all users)

(●) Only for me (Hp)

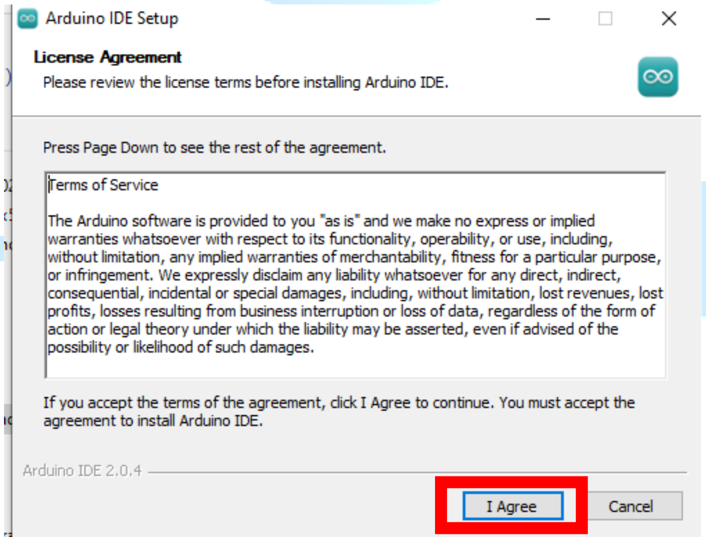
There is already a per-user installation.(C:\Users\Hp\AppData\Local\Programs\Arduino IDE)
Will reinstall/upgrade.



Setup will install Arduino IDE in the following folder. To install in a different folder, click Browse and select another folder. Click Install to start the installation.

Destination Folder

C:\Users\Hp\AppData\Local\Programs\Arduino IDE


Arduino IDE Setup
License Agreement
Please review the license terms before installing Arduino IDE.

Press Page Down to see the rest of the agreement.

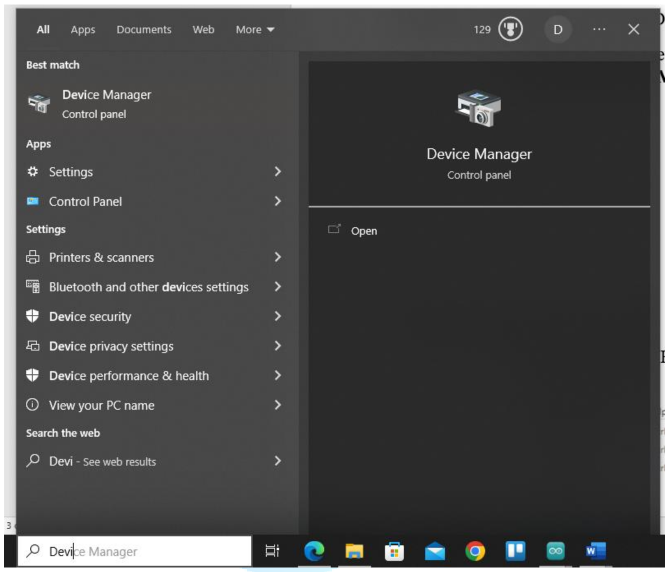
Terms of Service

The Arduino software is provided to you "as is" and we make no express or implied warranties whatsoever with respect to its functionality, operability, or use, including, without limitation, any implied warranties of merchantability, fitness for a particular purpose, or infringement. We expressly disclaim any liability whatsoever for any direct, indirect, consequential, incidental or special damages, including, without limitation, lost revenues, lost profits, losses resulting from business interruption or loss of data, regardless of the form of action or legal theory under which the liability may be asserted, even if advised of the possibility or likelihood of such damages.

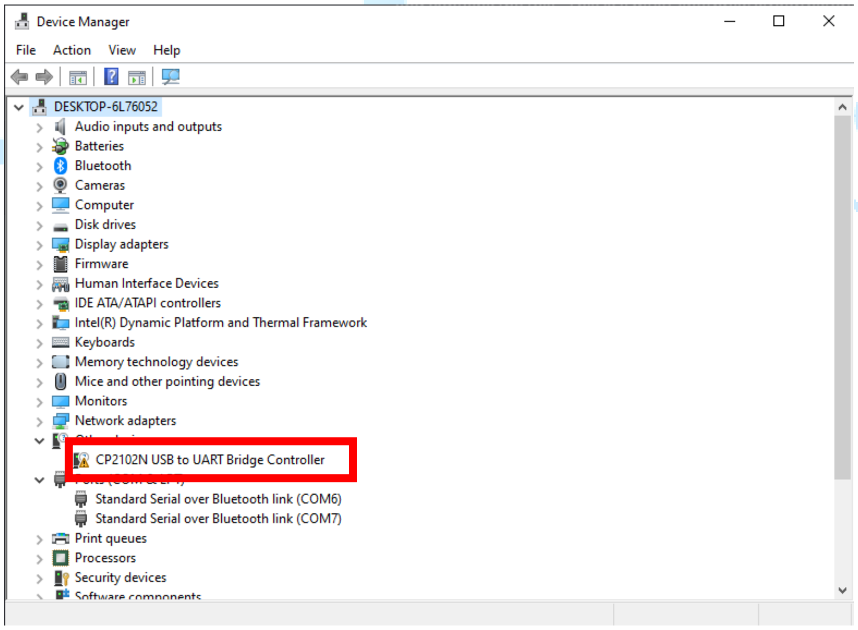
If you accept the terms of the agreement, click I Agree to continue. You must accept the agreement to install Arduino IDE.

Step 3: Install ESP32 Serial Chip Drivers

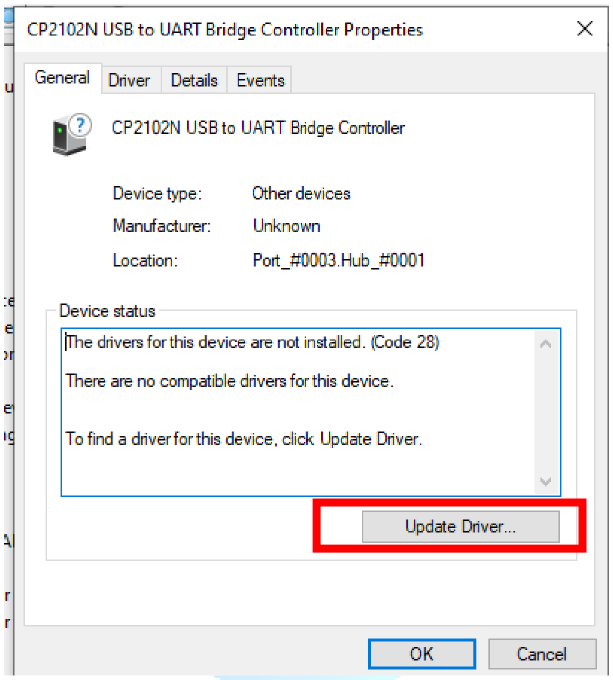
i. Connect your esp32 to laptop. Ensure that you have downloaded and extracted the GitHub repo "IoT Workshop" and connect the ESP32 Development board to your laptop.



ii. Search for **"Device Manager"** in search bar and open it. Look for the CP2102 device in other devices.



iii. Double Click and select Update Driver Option.


CP2102N USB to UART Bridge Controller Properties

CP2102N USB to UART Bridge Controller

Device type:	Other devices
Manufacturer:	Unknown
Location:	Port_#0003.Hub_#0001

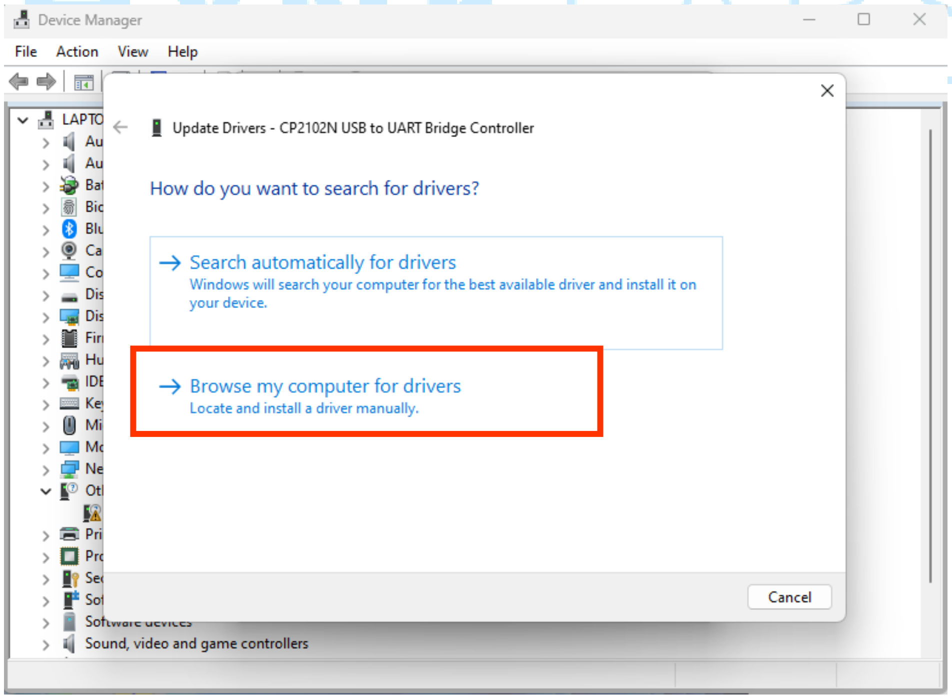
Device status

The drivers for this device are not installed. (Code 28)

There are no compatible drivers for this device.

To find a driver for this device, click Update Driver.

iv. Now browse your computer for the folder you just downloaded from github. Pass the path of folder “Driver CP” and press Install. Wait for the installation to complete.

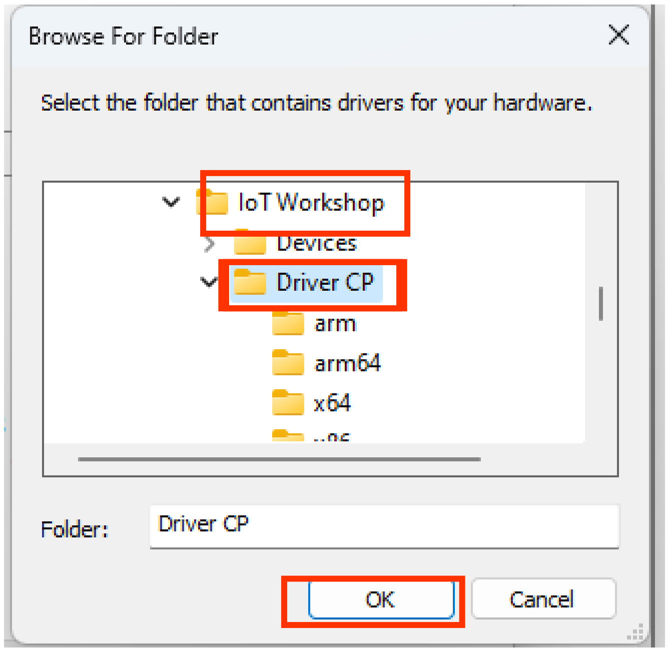


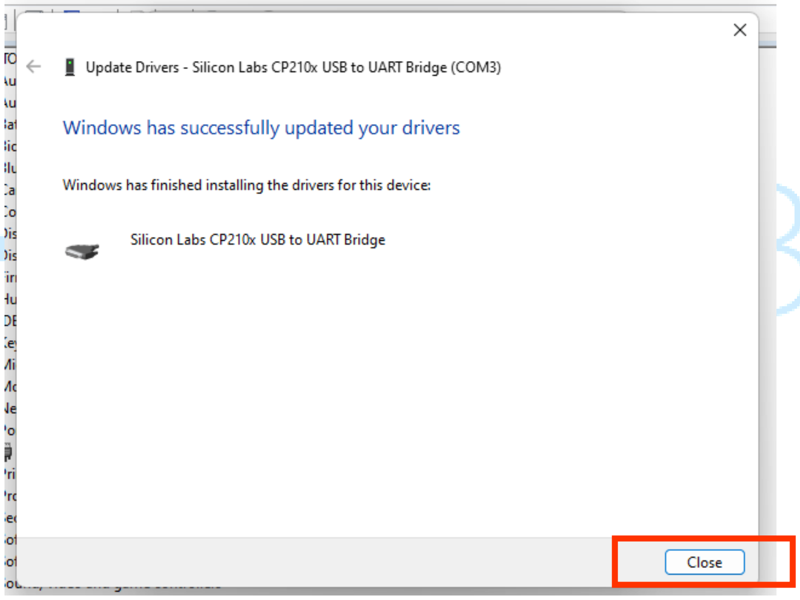
Update Drivers - CP2102N USB to UART Bridge Controller

How do you want to search for drivers?

→ Search automatically for drivers
Windows will search your computer for the best available driver and install it on your device.

→ Browse my computer for drivers
Locate and install a driver manually.




Update Drivers - Silicon Labs CP210x USB to UART Bridge (COM3)
Close

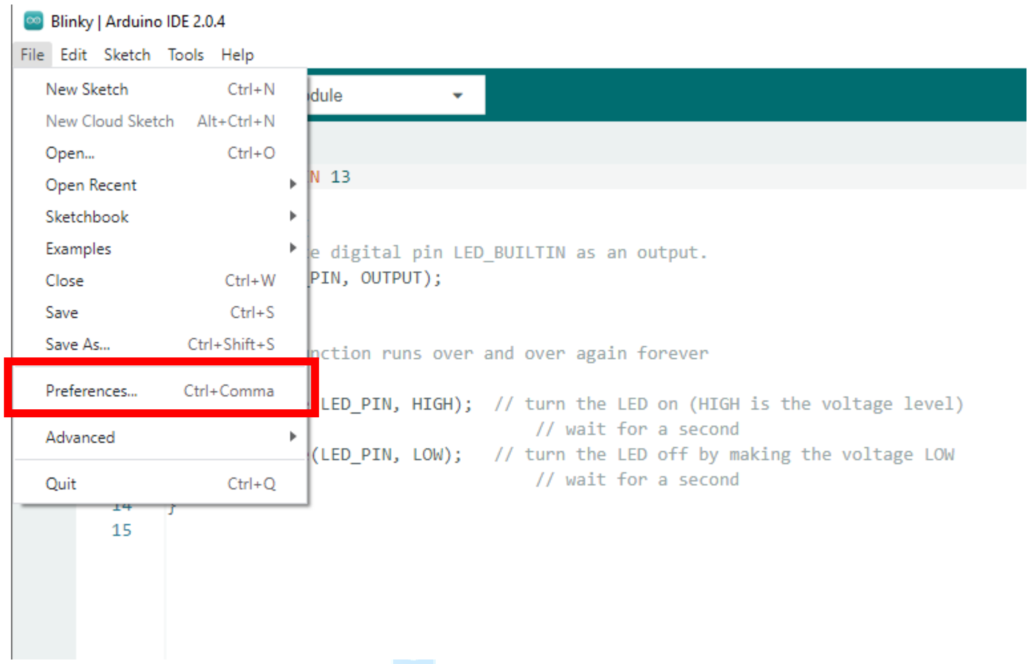
Windows has successfully updated your drivers

Windows has finished installing the drivers for this device:

Silicon Labs CP210x USB to UART Bridge

Step 4: Install ESP32 board in IDE

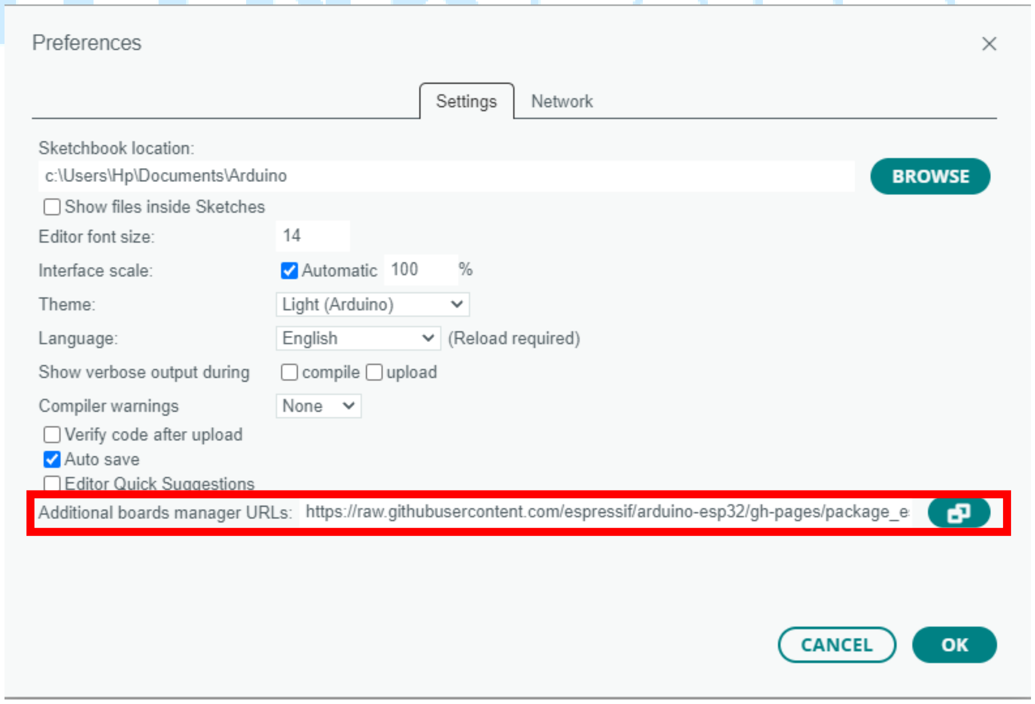
i. Open Arduino IDE. Head over to **File>Preferences**



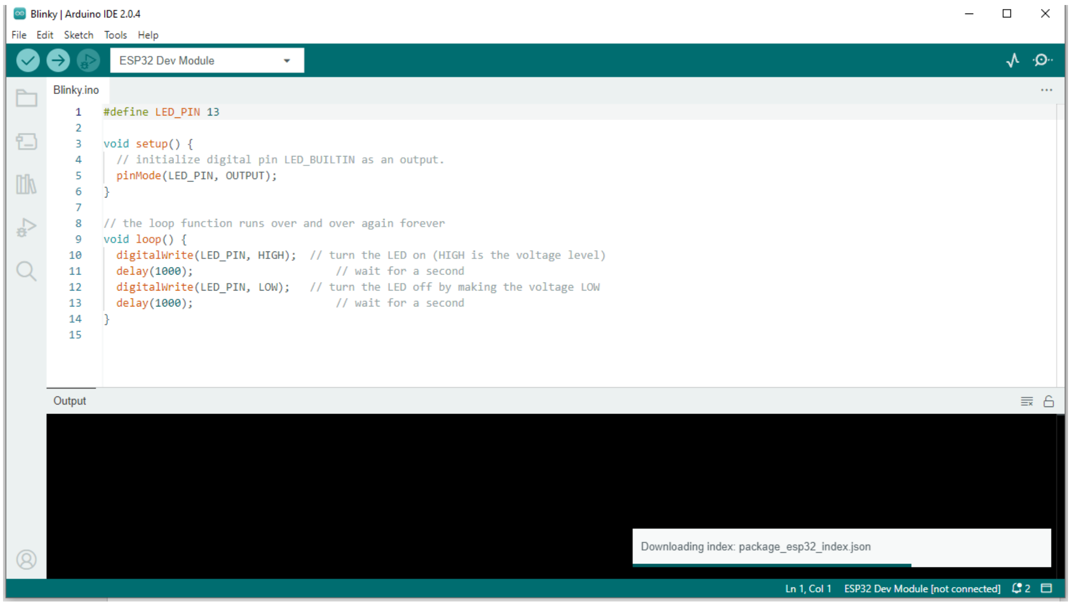
```
N 13
e digital pin LED_BUILTIN as an output.
PIN, OUTPUT);
nction runs over and over again forever
LED_PIN, HIGH);  // turn the LED on (HIGH is the voltage level)
                    // wait for a second
(LED_PIN, LOW);   // turn the LED off by making the voltage LOW
                    // wait for a second
}
```

ii. Add the following URL in the "Additional Boards Manager" and Click "OK"

https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json



Sketchbook location:
c:\Users\Hp\Documents\Arduino
☐ Show files inside Sketches
Editor font size: 14
Interface scale: ☑ Automatic 100 %
Theme: Light (Arduino)
Language: English (Reload required)
Show verbose output during ☐ compile ☐ upload
Compiler warnings None
☐ Verify code after upload
☑ Auto save
☐ Editor Quick Suggestions
Additional boards manager URLs: https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_e



```
#define LED_PIN 13

void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_PIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_PIN, HIGH);  // turn the LED on (HIGH is the voltage level)
  delay(1000);                      // wait for a second
  digitalWrite(LED_PIN, LOW);   // turn the LED off by making the voltage LOW
  delay(1000);                      // wait for a second
}
```

```
Downloading index: package_esp32_index.json
```

```
Ln 1, Col 1   ESP32 Dev Module [not connected]   2
```

iii. Wait for the installation to finish.

iv. Once done, ESP32 can be used in Arduino IDE.

Step 5: Install following libraries in Arduino IDE.

a. **DHT sensor library** by Adafruit
b. **PubSubClient** by Nick O'Leary
c. ArduinoJson
d. MQTT by Joel Gaehwiler

2. Verify installation with LED blinking Code.

Create a new sketch in Arduino IDE, save as “Blinky” and type the following code.

i. At the top of code define a pin for LED

```
#define LED_PIN 13
```

ii. Now in “Setup ()” function set pin mode for LED_PIN as OUTPUT. Use following function.

```
pinMode(LEDPIN, OUTPUT);
```

Place it in the setup function.

```
void setup() {
 // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_PIN, OUTPUT);
}
```

iii. Next, the requirement is to blink the LED continuously every second.

To do this you need to set the pin HIGH and wait for 1 second and then set it to LOW and then again wait for 1 second.

You can use the following method to set pin HIGH or LOW.

```
digitalWrite(LED_PIN, HIGH);
```

Place this method in the main loop.

```
void loop() {
  digitalWrite(LED_PIN, HIGH);  // turn the LED on
  delay(1000);                  // wait for a second
  digitalWrite(LED_PIN, LOW);   // turn the LED off
  delay(1000);                  // wait for a second
}
```

3. Interface DHT11 to read Temperature and Humidity.

Create a new sketch in Arduino IDE, save as "DHT11" and type the following code.

i. At the top, import the library "DHT" installed in earlier section.

```
#include <DHT.h>
```

ii. Now, define the sensor pin, sensor type.

```
#define DHTPIN 27     // GPIO pin connected to DHT11 data pin
#define DHTTYPE DHT11 // DHT11 sensor type
```

iii. Initialize the DHT class from library. And Pass the DHTPIN and DHTTYPE values

```
DHT dht(DHTPIN, DHTTYPE);
```

iv. In, Setup() function, begin the Serial Monitor and initialize the DHT sensor using following code.

```
void setup() {
  Serial.begin(115200);
  dht.begin();
}
```

v. The goal is to read humidity and temperature values from the sensor every 5 seconds. To achieve this, the DHT library provides two methods respectively.

```
dht.readTemperature() -> Used to read Temperature.

dht.readHumidity()    -> Used to read Humidity
```

Use the methods to get values from the sensor and print it on serial monitor, every 5 seconds.

```
void loop() {
  float temperature = dht.readTemperature();
  float humidity = dht.readHumidity();

  Serial.print("Temperature: ");
  Serial.print(temperature);
  Serial.print(" °C\tHumidity: ");
  Serial.print(humidity);
  Serial.println(" %");

  delay(2000); // wait for 2 seconds
}
```

4. Temperature and Humidity controlled Fan.

Control a fan automatically, based upon the

Create a new sketch in Arduino IDE, save as "Automatic_Fan" and type the following code. Copy the full code from DHT, this will serve as a base for this section.

i. Define an output pin at the top for the FAN

```
#define FAN 4     // GPIO pin connected to Relay Module
```

ii. Place a condition in main loop,

If (Humidity >= 35 && temperature>=25) - Turn ON the fan.

Else – Turn OFF the fan

(Tip: You may use the "`digitalWrite`" function to turn ON/OFF the fan)

5. DHT11 Beacon based on HTTP.

Report Temperature and Humidity to a http server every 5 seconds.

Create a new sketch in Arduino IDE, save as “DHT11_Beacon” and type the following code. Copy the full code from DHT, this will serve as a base for this section.

i. Include following libraries

```
#include <WiFi.h>
#include <HTTPClient.h>
#include <DHT.h>
```

ii. Define DHT pin and your allotted group number. Now turn ON Hotspot of your mobile phones and define your ssid and password in the code as follows

```
#define GROUP 1
#define DHTPIN 4      // DHT11 data pin
#define DHTTYPE DHT11   // DHT11 sensor type

const char* ssid = "Test"; // Replace with your ssid
const char* password = "Test1234"; // Replace with your password
const char* serverUrl = "http://api.thynklabs.in:5000/sensor-data";

DHT dht(DHTPIN, DHTTYPE);
HTTPClient http;
```

iii. In setup, begin serial monitor. Also start wifi and attempt connection with the router through ssid and password.

```
WiFi.begin(ssid, password);
```

Now, check if wifi is connected, if it is not connected then wait indefinitely, that is do not proceed. We can use the following method to check the status.

```
WiFi.status();
```

Since we have to wait indefinitely if not connected, it can be placed in while loop as follows:

```
while (WiFi.status() != WL_CONNECTED) {
  delay(1000);
  Serial.println("Connecting...");
}
```

The complete setup function will look like follows:

```
void setup() {
  Serial.begin(115200);
  delay(1000);
  Serial.println("Connecting to WiFi...");

  // Connect to WiFi network
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting...");
  }

  Serial.println("WiFi connected.");
}
```

iv. Our aim is to report Humidity and Temperature via HTTP for every 5 seconds. So, in main loop we will need to read the temperature and humidity as well as report it to the server.

a. Read temperature and humidity

```
float temperature = dht.readTemperature();
float humidity = dht.readHumidity();
```

b. Convert the file to JSON format

JSON stands for JavaScript Object Notation, it is a lightweight format for exchanging data between different systems. It is a text-based format that is easy to read and write, making it popular for use in web applications and other data-driven projects.

JSON is based on key-value pairs and is structured in a way that is easy to parse and generate using a variety of programming languages. It is often used to transmit data between a server and a web application, and can be used to store and exchange data in a variety of formats, including strings, numbers, arrays, and objects.

JSON is often used in web services and APIs, as it allows data to be transmitted quickly and efficiently between different systems. It is also a popular format for storing configuration data and other application settings.

Example: **{“temperature”:25, “humidity”:40}**

```
Use following code:

String data = "{";
  data += "\"group\":";
  data += String(GROUP);
  data += ",";
  data += "\"humidity\":";
  data += String(humidity);
  data += ",";
  data += "\"temperature\":";
  data += String(temperature);
  data += "}";
```

c. Now, initialize http and make a POST request to the server.

```
http.begin( serverUrl);
http.addHeader("Content-Type", "application/json");
int httpResponseCode = http.POST(data);
```

d. Check for response and log the response code. If the code is 200 then our post request was successfully made. End the HTTP initializer at the end.

```
  // Check for errors
  if (httpResponseCode > 0) {
    Serial.print("HTTP response code: ");
    Serial.println(httpResponseCode);
  } else {
    Serial.print("HTTP POST request failed, error: ");
    Serial.println(http.errorToString(httpResponseCode).c_str());
  }

  http.end();   // Close HTTP connection

  delay(5000); // Wait for 5 seconds to send another request
```

6. Make a post request with DeviceID

Use the code from DHT11 beacon to send data to the server. However, this time include a random deviceId like: **group1_beacon**.

You will have to make necessary changes in the following part of code.

```
String data = "{";
  data += "\"group\":";
  data += String(GROUP);
  data += ",";
  data += "\"humidity\":";
  data += String(humidity);
  data += ",";
  data += "\"temperature\":";
  data += String(temperature);
  data += "}";
```

Add - “deviceId” : “group1_beacon”

DAY II: Cloud Platforms for IoT.

1. Connect to HiveMQ public broker.

i. Open the Arduino IDE and create a new sketch.

ii. Include the WiFi.h and PubSubClient.h libraries in your sketch by adding the following lines at the top of your code:

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

iii. Define the SSID and password of the WiFi network you want to connect to using the following lines of code:

```
const char* ssid = "YourNetworkSSID";
const char* password = "YourNetworkPassword";
```

Replace "YourNetworkSSID" and "YourNetworkPassword" with the actual SSID and password of the WiFi network you want to connect to.

iv. Define the MQTT broker details by adding the following lines of code:

```
const char* mqtt_server = "broker.hivemq.com";
const int mqtt_port = 1883;
```

These lines define the address and port of the HiveMQ open broker.

v. Create a WiFi client object and a PubSubClient object by adding the following lines of code:

```
WiFiClient espClient;
PubSubClient client(espClient);
```

vi. In the setup() function, initialize the WiFi connection by calling WiFi.begin() and passing in the SSID and password you defined earlier. Then, connect to the MQTT broker by calling client.setServer() and passing in the broker details:

```
void setup() {
  Serial.begin(115200);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi...");
  }
  Serial.println("Connected to WiFi");

  client.setServer(mqtt_server, mqtt_port);
}
```

vii. In the loop() function, check if the ESP32 is connected to the MQTT broker using client.connected(). If not, attempt to reconnect using client.connect() and passing in any required authentication details. Then, publish a message to a topic using client.publish()

```
void loop() {
  if (!client.connected()) {
    Serial.println("Connecting to MQTT broker...");
    if (client.connect("group_1"){ // Replace this as per your group num
      Serial.println("Connected to MQTT broker");
    } else {
      Serial.print("Failed to connect to MQTT broker, rc=");
      Serial.print(client.state());
      Serial.println(" retrying in 5 seconds");
      delay(5000);
      return;
    }
  }
  client.loop();

  client.publish("iot_workshop/bncoe/group/1", "Hello from ESP32");

  delay(1000);
}
```

Replace the “group/1” in topic with your group number

viii. Visit to HiveMQ browser client, subscribe to the same topic and check if the logs appear.

2. Control a relay on ESP32 from HiveMQ

Use the template code “connect_mqtt”.

i. Head over to the comment “Set Callback” and add the following code.

```
client.setCallback(callback);
```

ii. Head over to the comment “Subscribe code here” and add code to subscribe to the following topic.

Topic: **“`iot_workshop/bncoe/group/1/relay`”**

Use the following line of code to subscribe.

```
client.subscribe(“iot_workshop/bncoe/group/1/relay”);
```

iii. Now paste the following lines of code after comment “Callback function here”

Guidelines:

a. Set pinMode as Output for pin 4 in “Setup” function
b. Use digitalWrite to set the pin HIGH or LOW

```
// Callback Function here
void callback(char* topic, byte* payload, unsigned int length) {
    // Log incoming message and topic

  Serial.print("Topic:");
  Serial.println(topic);
  Serial.print("Message:");
  Serial.println((char*)payload);


  // Check if the message is 1
  if((char)payload[0] == '1'){
    // Set Esp32 pin 4 HIGH
  }
  else if((char)payload[0] == '0'){
    // Set Esp32 pin 4 LOW
  }
}
```

3. Connect to AWS IoT

Use the template code “aws_iot_connect”. Get certificates for your devices from the devices folder. Use certificates group wise.

i. Open the template code in Arduino IDE and open **secrets.h.**

Use your group name as thingname, as follows

```
#define THINGNAME "group_1"
```

```
Add SSID and Password of your mobile hotspot.

const char WIFI_SSID[] = "";
const char WIFI_PASSWORD[] = "";
```

```
Copy and paste the endpoint OF AWS IOT
```

“aynomhw2tgffs-ats.iot.ap-south-1.amazonaws.com”

```
const char AWS_IOT_ENDPOINT[] = "aynomhw2tgffs-ats.iot.ap-south-
1.amazonaws.com";
```

Replace the Private key, Certificate and RootCA with the one provided to you.

Head over to this part of code:

```
static const char AWS_CERT_CA[] PROGMEM = R"EOF(
-----BEGIN CERTIFICATE-----
-----END CERTIFICATE-----
)EOF";
```

Replace the contents inside parenthesis with your certificate as follows.

```
static const char AWS_CERT_CA[] PROGMEM = R"EOF(
-----BEGIN CERTIFICATE-----
MIIDQTCCAimgAwIBAgITBmyfz5m/jAo54vB4ikPmljZbyjANBgkqhkiG9w0BAQsF
ADA5MQswCQYDVQQGEwJVUzEPMA0GA1UEChMGQW1hem9uMRkwFwYDVQQDExBBbWF6
b24gUm9vdCBDQSAxMB4XDTE1MDUyNjAwMDAwMFoXDTM4MDExNzAwMDAwMFowOTEL
MAkGA1UEBhMCVVMxDzANBgNVBAoTBkFtYXpvbjEZMBcGA1UEAxMQQW1hem9uIFJv
b3QgQ0EgMTCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBALJ4gHHKeNXj
ca9HgFB0fW7Y14h29Jlo91ghYPl0hAEvrAIthtOgQ3pOsqTQNroBvo3bSMgHFzZM
9O6II8c+6zf1tRn4SWiw3te5djgdYZ6k/oI2peVKVuRF4fn9tBb6dNqcmzU5L/qw
IFAGbHrQgLKm+a/sRxmPUDgH3KKHOVj4utWp+UhnMJbulHheb4mjUcAwhmahRWa6
VOujw5H5SNz/0egwLX0tdHA114gk957EWW67c4cX8jJGKLhD+rcdqsq08p8kDi1L
93FcXmn/6pUCyziKrlA4b9awiuehrxcceVOF34GfID5yHI9Y/QCB/IIDEgEw+OyQm
jgSubJrIqg0CAwEAAaNCMEAwDwYDVR0TAQH/BAUwAwEB/zAOBgNVHQ8BAf8EBAMC
AYYwHQYDVR0OBBYEFIQYzIU07LwMlJQuCFmcx7IQTgoIMA0GCSqGSIb3DQEBCwUA
A4IBAQCY8jdaQZChGsV2USggNiMOruYou6r4lK5IpDB/G/wkjUu0yKGX9rbxenDI
U5PMCCjjmCXPI6T53iHTfIUJrU6adTrCC2qJeHZERxhlbI1Bjjt/msv0tadQ1wUs
N+gDS63pYaACbvXy8MWy7Vu33PqUXHeeE6V/Uq2V8viTO96LXFvKWlJbYK8U90vv
o/ufQJVtMVT8QtPHRh8jrdkPSHCa2XV4cdFyQzR1bldZwgJcJmApzyMZFo6IQ6XU
5MsI+yMRQ+hDKXJioaldXgjUkK642M4UwtBV8ob2xJNDd2ZhwLnoQdeXeGADbkpy
rqXRfboQnoZsG4q5WTP468SQvvG5
-----END CERTIFICATE-----
)EOF";
```

Repeat this for Device Certificate and Private Key

ii. Include secrets.h at the beginning of main file.

```
#include "secrets.h"
```

iii. Pass certificates to the client. Head over to comment "//Configure WiFiClientSecure to use the AWS IoT device credentials"

```
net.setCACert(AWS_CERT_CA);
net.setCertificate(AWS_CERT_CRT);
net.setPrivateKey(AWS_CERT_PRIVATE);
```

iv. Create a function to publish the message to aws iot.

a. First, Serialize the data in JSON.

b. Then call publish method of MQTT client.

The complete function will be as follows

```
void publishMessage()
{
  StaticJsonDocument<200> doc;
  doc["time"] = millis();
  doc["group"] = String("group_1"); // replace this with your group num

  char jsonBuffer[512];
  serializeJson(doc, jsonBuffer); // print to client
  client.publish(AWS_IOT_PUBLISH_TOPIC, jsonBuffer);
}
```

Place this function after comment “Publish Message function here

v. Call the publish function in main loop every 1 second.

```
publishMessage();
```

4. Publish Humidity and Temperature to AWS IoT

Use template “aws_iot_pub” and make the following changes.

i. Copy **“secrets.h”** from previous code.

ii. At the top, import the library “DHT” installed in earlier section.

```
#include <DHT.h>
```

iii. Now, define the sensor pin, sensor type.

```
#define DHTPIN 27      // GPIO pin connected to DHT11 data pin
#define DHTTYPE DHT11 // DHT11 sensor type
```

iv. Initialize the DHT class from library. And Pass the DHTPIN and DHTTYPE values

```
DHT dht(DHTPIN, DHTTYPE);
```

v. Change the publishMessage function as follows.

```
void publishMessage()
{
  StaticJsonDocument<200> doc;
  doc["temperature"] = dht.readTemperature();
  doc["humidity"] = dht.readHumidity();
  char jsonBuffer[512];
  serializeJson(doc, jsonBuffer); // print to client

  client.publish(AWS_IOT_PUBLISH_TOPIC, jsonBuffer);
}
```

Final Task: Control Relay from AWS IoT