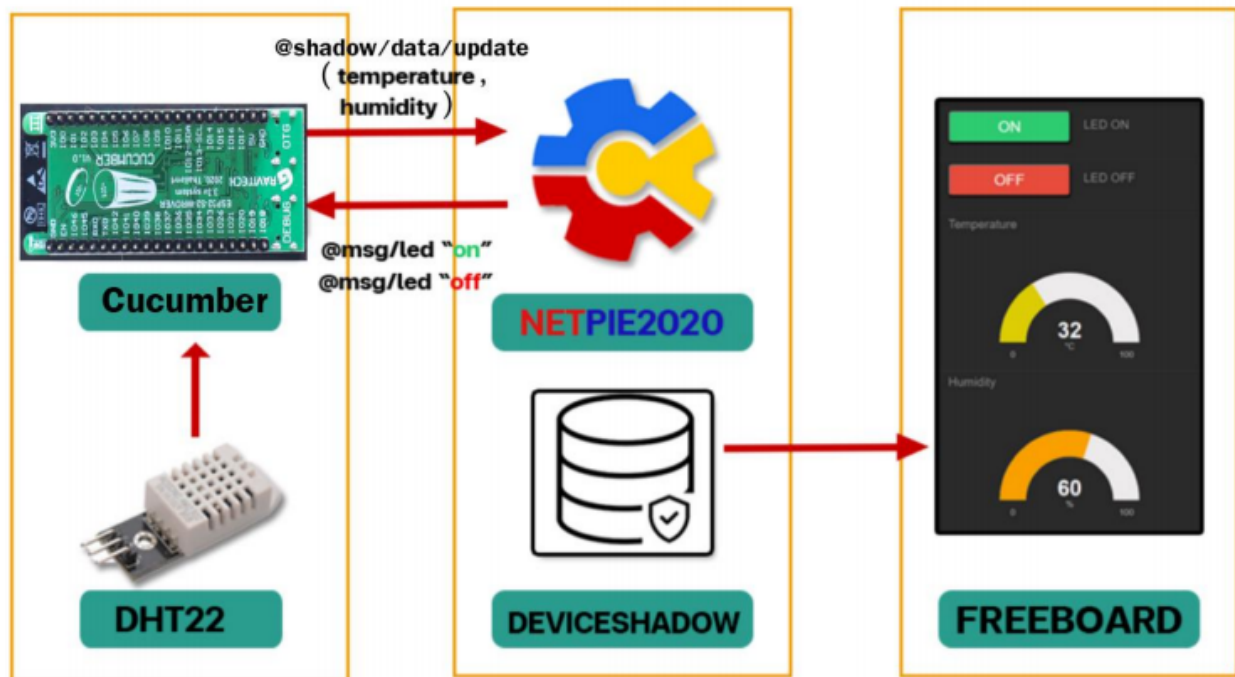


Example of Connecting and Using Cucumber with NETPIE2020

Project Description

This example demonstrates how to transmit temperature and humidity readings from DHT22 to NETPIE2020, to present them on Freeboard and control the on/off action of the led connected to the Cucumber.



Working model

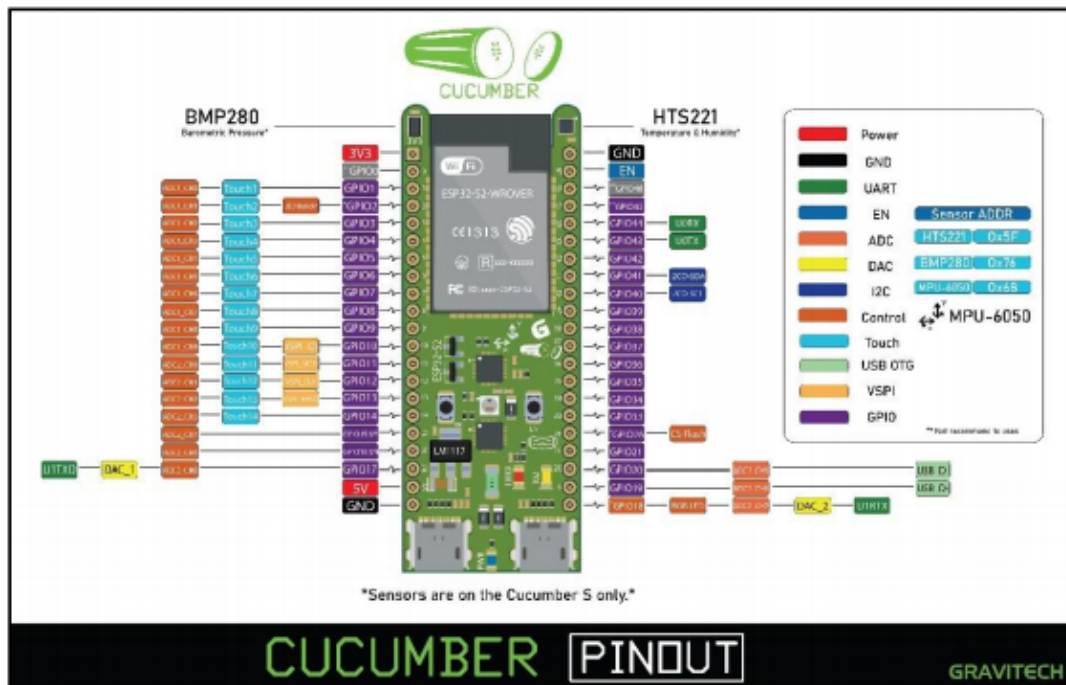
Require Basic Knowledge of

- 1.Using NETPIE2020
- 2.Using NETPIE2020 Freeboard

Equipment used

1.Cucumber

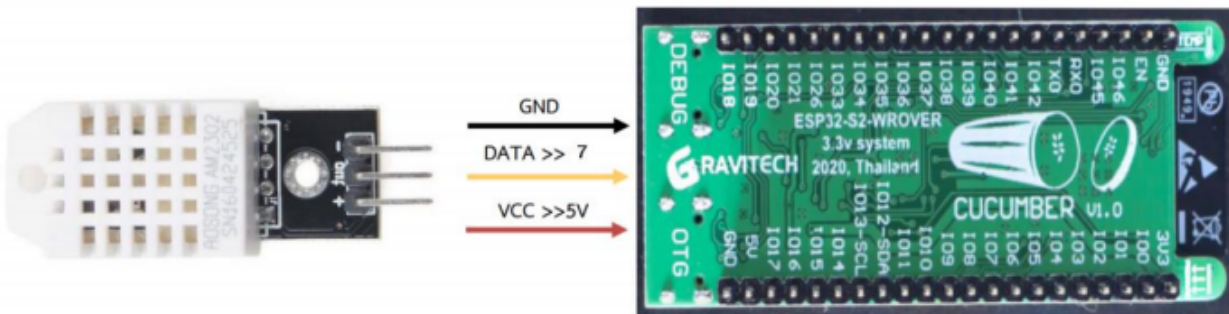
Cucumber RS is a WiFi IoT development board. It features the latest ESP32-S2 chipset from Espressif (ESP32-S2-WROVER). The RS version is a special version which includes many on-board sensors like HS221 Temperature & Humidity sensor, MPU-6050 3-axis Accelerometer & 3-axis Gyroscope, and BMP280 Pressure sensor. It is packed with many goodies like x2 USB Type-C (debug and OTG), FTDI chip for USB-to-Serial converter which highly compatible with most computers, RGB LED which use only one signal line and can mix up to over 16 million color, TX/RX, IO2 and PWR LED.



2.DHT22 Temperature and Humidity Sensor



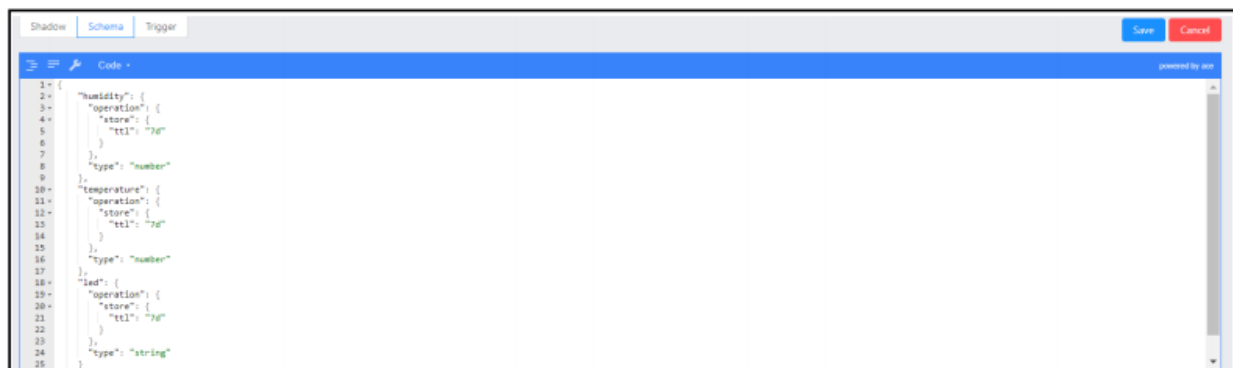
Cucumber Circuit Connection



Circuit connection between DHT22 and Cucumber

Defining Device Schema

The first part is to define the device schema, which is the data structure for the devices generating data. The server checks the data structure defined in the device schema and performs the required actions before storing the data in the Timeseries database. These actions include converting the data units and data validation.



Description of Schema

Stores the first variable 'humidity' of type number in the Timeseries database with a retention period of 7 days.

Stores the second variable 'temperature' of type number in the Timeseries database with a retention period of 7 days.

Stores the third variable 'led' of type string in the Timeseries database with a retention period of 7 days.

Description of Arduino Program

First, the required libraries for executing the program should be imported, followed by defining the program parameters like LED PIN and DHT PIN. Next, configure the network parameters like server, port, client ID, username, and password for connecting to the NETPIE2020.

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

#define DHTPIN 7
#define DHTTYPE DHT22
#define LED1 2

const char* ssid = "your wifi";
const char* password = "your wifi password";
const char* mqtt_server = "broker.netpie.io";
const int mqtt_port = 1883;
const char* mqtt_Client = "ClientID";
const char* mqtt_username = "Token";
const char* mqtt_password = "Secret";
```

The second part is to run the MQTT commands.

```
WiFiClient espClient;
PubSubClient client(espClient);
DHT dht(DHTPIN, DHTTYPE);
char msg[50];
```

The MQTT connect function is used to connect to the MQTT server. If the connection is successful, it will display the message saying 'connected'. But, if the connection is unsuccessful, 'failed' message is displayed and will try to reconnect automatically.

```

void reconnect() {
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    if (client.connect(mqtt_Client, mqtt_username, mqtt_password)) {
      Serial.println("connected");
      client.subscribe("@msg/led");
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println("try again in 5 seconds");
      delay(5000);
    }
  }
}

```

The callback function is used to check messages received from the Freeboard, which first checks the topic name, which is '@msg/led', then checks the message payload. If the received message contains 'on', led is turned on and if the message contains 'off', led is turned off. The status of the led is published in the JSON format to update the Shadow.

```

void callback(char* topic, byte* payload, unsigned int length) {
  Serial.print("Message arrived [");
  Serial.print(topic);
  Serial.print("] ");
  String message;
  for (int i = 0; i < length; i++) {
    message = message + (char)payload[i];
  }
  Serial.println(message);
  if (String(topic) == "@msg/led") {
    if (message == "on") {
      digitalWrite(LED1, 0);
      client.publish("@shadow/data/update", "{\"data\" : {\"led\" : \"on\"}}");
      Serial.println("LED on");
    }
    else if (message == "off") {
      digitalWrite(LED1, 1);
      client.publish("@shadow/data/update", "{\"data\" : {\"led\" : \"off\"}}");
      Serial.println("LED off");
    }
  }
}

```

The setup function attempts to use the internet by running the command Ethernet.begin and initializes the dht by running the dht.begin command.

```

void setup() {
  pinMode(LED1, OUTPUT);
  Serial.begin(115200);
  digitalWrite(LED1, 0); //LED OFF
  Serial.print("Connecting to ");

  if (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());
  client.setServer(mqtt_server, mqtt_port);
  client.setCallback(callback);
  dht.begin();
}

```

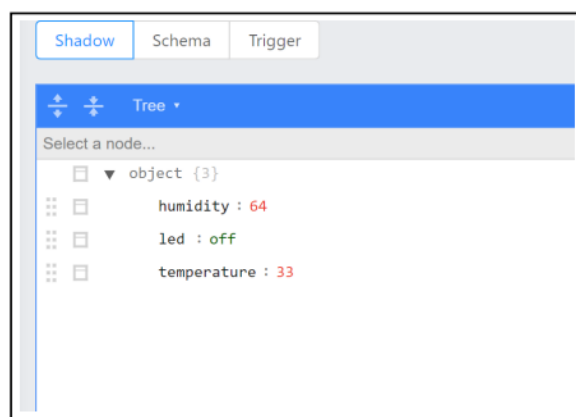
The last part is to define the variables Temperature and Humidity and publish the values on Topic: shadow/data/update for every 2 seconds.

```

void loop() {
  int humidity = dht.readHumidity();
  int temperature = dht.readTemperature();
  if (!client.connected()) {
    reconnect();
  }
  client.loop();
  String data = "{\"data\": {\"humidity\": " + String(humidity) + ", \"temperature\": " + String(temperature) + "}}";
  Serial.println(data);
  data.toCharArray(msg, (data.length() + 1));
  client.publish("@shadow/data/update", msg);
  delay(2000);
}

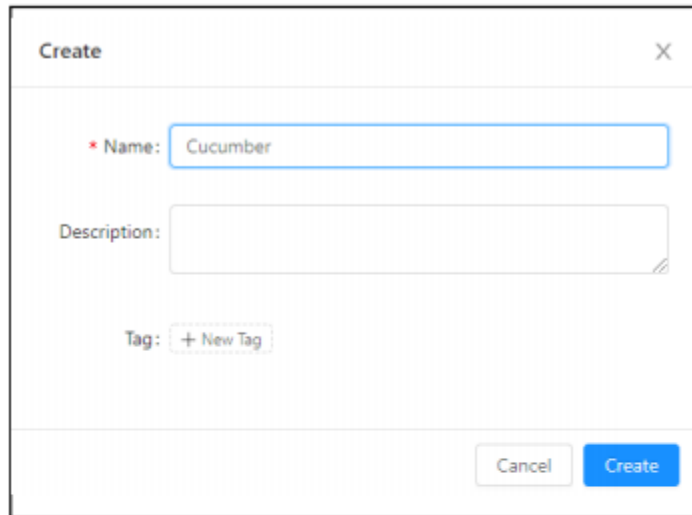
```

Messages sent to the NETPIE2020 and the values are saved in the shadow.

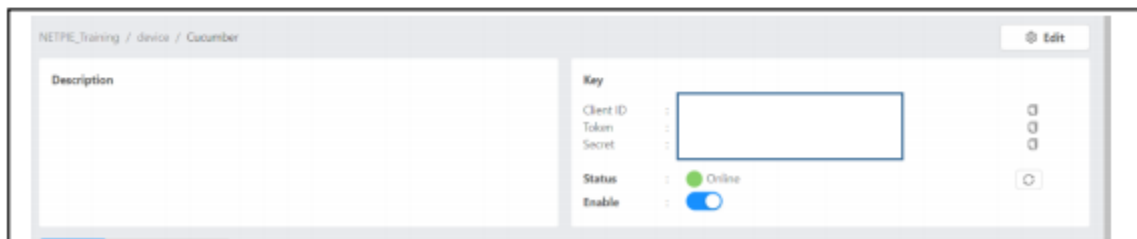


Creating a Device on NETPIE2020

1.Start by selecting the menu Device List > Create and name the device as shown in the figure below.



A 'Create' dialog box with a close button (X) in the top right corner. It contains three input fields: 'Name' with the value 'Cucumber', 'Description' (empty), and 'Tag' with a '+ New Tag' button. At the bottom right are 'Cancel' and 'Create' buttons.



A screenshot of the 'NETPIE_Training / device / Cucumber' page. It features a table with a 'Description' header. To the right, there is a 'Key' section with fields for 'Client ID', 'Token', and 'Secret'. Below these, the 'Status' is shown as 'Online' with a green dot, and the 'Enable' toggle is turned on. An 'Edit' button is in the top right corner.

Creating device on NETPIE2020

Creating a Freeboard for Monitoring Temperature, Humidity, and Controlling the LED

1.Start by selecting the menu Freebord > Create and name the Freeboard as shown in the figure below.

Create Dashboard

Name *

cucumber

Description

Cancel

Create



Name	Create Date
cucumber	2021-05-11 07:21
ipst	2021-05-10 01:57

1-2 of 2 items < 1 > 10 / page

2.Fill in the details, which include Name, Device ID, and Device Token that can be found in the DeviceList named cucumber.

(NOTE: If the user enables the Feed, Freeboard keeps on calling the API. So, it is better to turn off the Feed if the user is not intended to use it.)

DATASOURCE

NAME

cucumber

DEVICE ID

Client ID see Device Registration page

DEVICE TOKEN

Token see Device Registration page

SUBSCRIBED TOPICS

Topic Request Subscribe

FEED

YES

☒

SINCE

6

Hour

Display data points since ... ago

DOWN SAMPLING

1

Minute

Resolution of the data points

SAVE

CANCEL

3. Create a widget of type Gauge for displaying the Temperature and Humidity values by clicking on 'ADD PANE' and fill in the details as shown in the figure below.

WIDGET

TYPE

Gauge

TITLE

Temperature

VALUE

`datasources["cucumber"]["shadow"]["temperature"]`

+ DATASOURCE

✕ JS EDITOR

UNITS

C

MINIMUM

0

MAXIMUM

100

SAVE

CANCEL

WIDGET

TYPE

Gauge

TITLE

Humidity

VALUE

`datasources["cucumber"]["shadow"]["humidity"]`

+ DATASOURCE

✕ JS EDITOR

UNITS

%H

MINIMUM

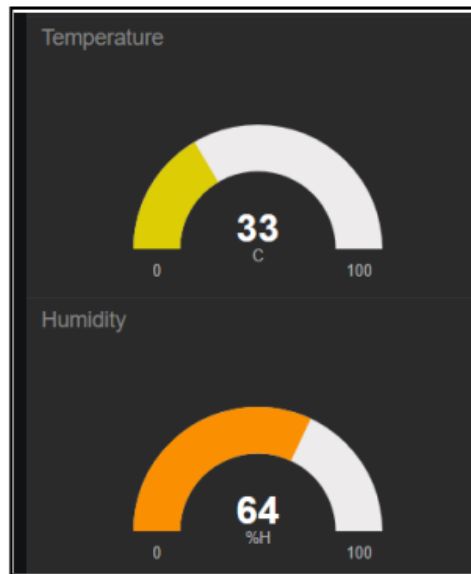
0

MAXIMUM

100

SAVE

CANCEL



Displaying Temperature and Humidity values stored in Shadow

4. Create ON/OFF buttons and set the ONCLICK ACTION on the ON button to
netpie["DataSourceName"].publish("@msg/led","on") and
netpie["DataSourceName"].publish("@msg/led","off") for the OFF button.

WIDGET

A simple button widget that can perform Javascript action.

TYPE

Button

BUTTON CAPTION

on

LABEL TEXT

LED_ON

BUTTON COLOR

Green

ONCLICK ACTION

netpie["cucumber"].publish("@msg/led","on")

+ DATASOURCE ✕ JS EDITOR Add some Javascript here

ONCREATED ACTION

JS code to run after a button is created

SAVE

CANCEL

Settings for ON button

WIDGET

A simple button widget that can perform Javascript action.

TYPE

Button

BUTTON CAPTION

off

LABEL TEXT

LED_OFF

BUTTON COLOR

Red

ONCLICK ACTION

netpie["cucumber"].publish("@msg/led","off")

+ DATASOURCE ✕ JS EDITOR Add some Javascript here

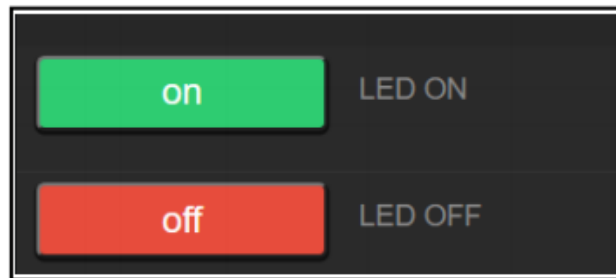
ONCREATED ACTION

JS code to run after a button is created

SAVE

CANCEL

Settings for OFF button

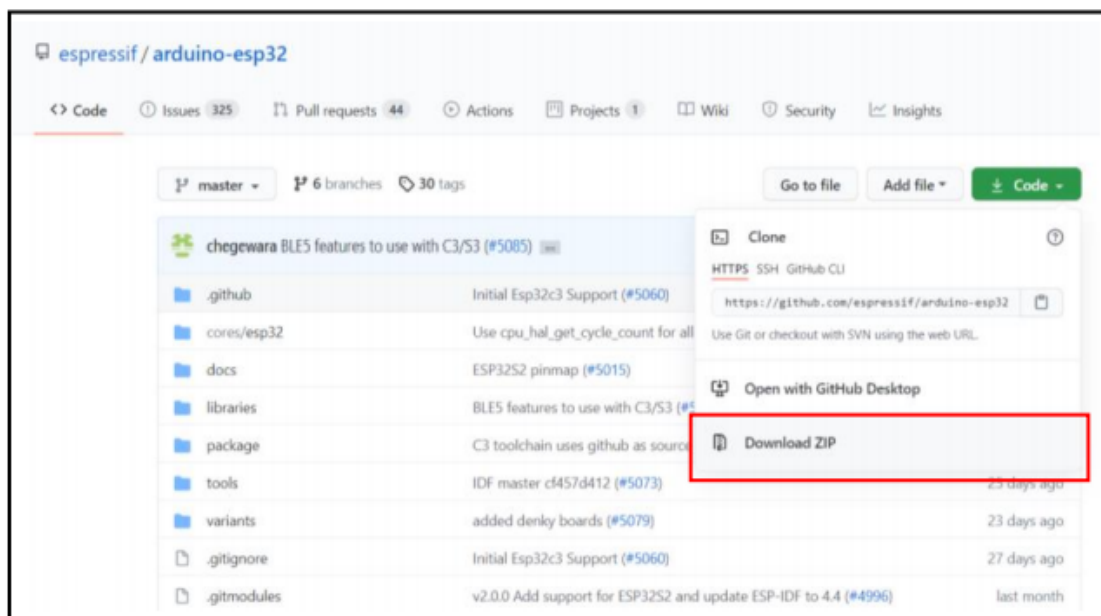


Appendix

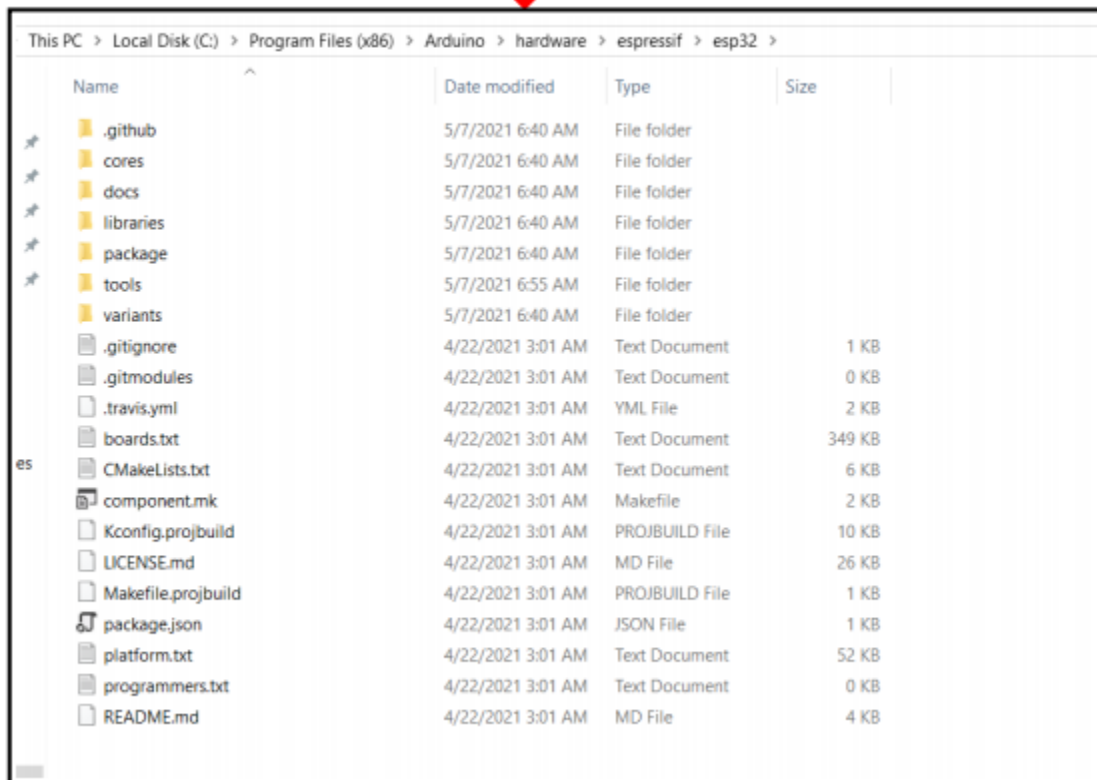
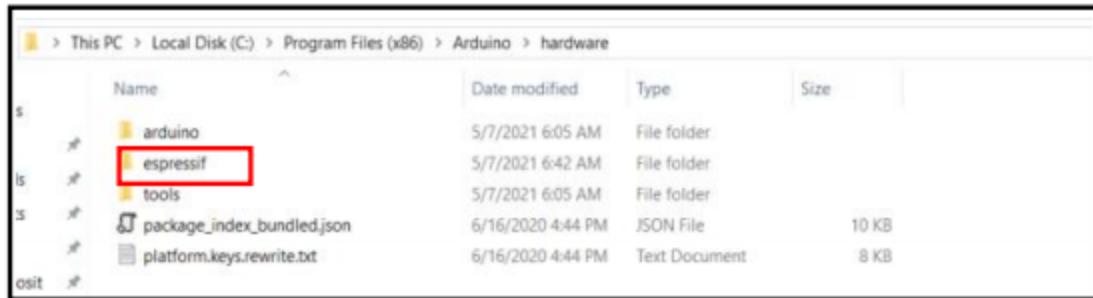
Setting up Cucumber board in the Arduino IDE

Download the zip file of arduino-esp32 developed by Espressif from github.

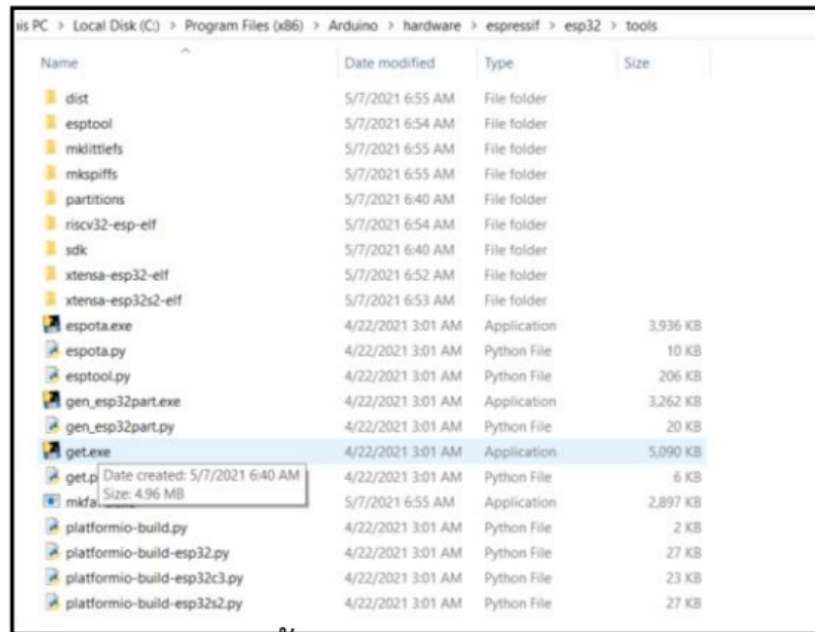
Link: <https://github.com/espressif/arduino-esp32>



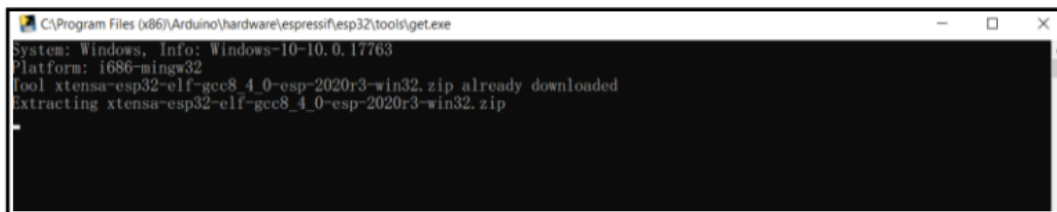
Once downloaded, extract the .zip file and give a name to the folder as esp32. Next, go inside the Arduino folder. Create a folder with the name Espressif in the 'hardware' folder and copy the esp32 folder into it.



Next, go inside the tools folder (esp32 > tools) and double click on the .exe file.



When you double click on the file, it will bring up a window as shown in the below figure. Wait till the program runs. Once it has finished running, new folders with names as xtensa-esp32-elf and xtensa-esp32s2-elf will be created as shown in the below figure.



PC > Local Disk (C:) > Program Files (x86) > Arduino > hardware > espressif > esp32 > tools

Name	Date modified	Type	Size
dist	5/7/2021 6:55 AM	File folder	
esptool	5/7/2021 6:54 AM	File folder	
mklittlefs	5/7/2021 6:55 AM	File folder	
mkspliffs	5/7/2021 6:55 AM	File folder	
partitions	5/7/2021 6:40 AM	File folder	
riscv32-esp-elf	5/7/2021 6:54 AM	File folder	
sdk	5/7/2021 6:40 AM	File folder	
xtensa-esp32-elf	5/7/2021 6:52 AM	File folder	
xtensa-esp32s2-elf	5/7/2021 6:53 AM	File folder	
esptota.exe	4/22/2021 3:01 AM	Application	3,936 KB
esptota.py	4/22/2021 3:01 AM	Python File	10 KB
esptool.py	4/22/2021 3:01 AM	Python File	206 KB
gen_esp32part.exe	4/22/2021 3:01 AM	Application	3,262 KB
gen_esp32part.py	4/22/2021 3:01 AM	Python File	20 KB
get.exe	4/22/2021 3:01 AM	Application	5,090 KB
get.py	4/22/2021 3:01 AM	Python File	6 KB
mkfatfs.exe	5/7/2021 6:55 AM	Application	2,897 KB
platformio-build.py	4/22/2021 3:01 AM	Python File	2 KB
platformio-build-esp32.py	4/22/2021 3:01 AM	Python File	27 KB
platformio-build-esp32c3.py	4/22/2021 3:01 AM	Python File	23 KB
platformio-build-esp32s2.py	4/22/2021 3:01 AM	Python File	27 KB

Open the Arduino IDE and under the tools section you will find the new board with name ESP32S2.

