

**Chancheep Mahacharoensuk 6288092****Kantapong Matangkarat 6288160****ESP32 as an MQTT Publisher and NETPIE2020 as an MQTT Broker**

Wire up the following program and complete the following questions

AHT20 SDA → GPIO21

AHT20 SCL → GPIO22

AHT20 Vcc → 3.3V

AHT20 GND → GND

LED → GPIO4

- Complete Lab10-NETPIE2020-AHT20.ino to send temperature and humidity values and show on NETPIE2020 dashboard and freeboard and submit the complete program on mycourses in the filename of Activity12-NETPIE2020-AHT20-<ID>-<Firstname>.ino
- What is the output message in Arduino IDE program for ESP32 to publish the temperature and humidity values to NETPIE2020?

The screenshot shows the Arduino IDE interface with the file `NETPIE2020_AHT20_LED.ino` open. The code includes libraries for WiFi, PubSubClient, and Adafruit\_AHT20. It defines a LED pin (4) and sets up an MQTT client. The `loop` function reads temperature and humidity from the AHT20 sensor and publishes them as JSON messages to the MQTT broker. The serial monitor (COM4) shows the output of the program, displaying the JSON messages being published.

```
#include <WiFi.h>
#include <PubSubClient.h>
#include <Adafruit_AHT20.h>

#include "config.h"

#define LED 4
#define FREQ 5000
#define LED_CH0 0
#define LED_RES 8

/* MQTT Instance */
WiFiClient espClient;
PubSubClient client(espClient);

bool wifiConnected = true;

Adafruit_AHT20 aht;
sensor_event_t humidity, temp;

/* Value Buffer */
char buf[200]; //Reserved for 200 bytes
long now, lastMsg;
String output;

void setup() {
  Serial.begin(115200);

  ledcSetup(LED_CH0, FREQ, LED_RES);
}
```

Serial Monitor Output (COM4):

```
12:20:114.427 -> {"data": {"temperature":128.08, "humidity":62.53,"place":"Timmy"}}
12:20:119.425 -> {"data": {"temperature":128.08, "humidity":62.58,"place":"Timmy"}}
12:20:124.421 -> {"data": {"temperature":128.10, "humidity":62.62,"place":"Timmy"}}
12:20:129.415 -> {"data": {"temperature":128.07, "humidity":62.59,"place":"Timmy"}}
12:20:134.407 -> {"data": {"temperature":128.08, "humidity":62.61,"place":"Timmy"}}
12:20:139.435 -> {"data": {"temperature":128.10, "humidity":62.66,"place":"Timmy"}}
12:20:144.423 -> {"data": {"temperature":128.06, "humidity":62.56,"place":"Timmy"}}
12:20:149.415 -> {"data": {"temperature":128.07, "humidity":62.65,"place":"Timmy"}}
12:20:154.438 -> {"data": {"temperature":128.09, "humidity":62.66,"place":"Timmy"}}
12:20:159.419 -> {"data": {"temperature":128.06, "humidity":62.59,"place":"Timmy"}}
12:21:104.404 -> {"data": {"temperature":128.06, "humidity":62.61,"place":"Timmy"}}
12:21:109.429 -> {"data": {"temperature":128.11, "humidity":62.62,"place":"Timmy"}}
12:21:114.423 -> {"data": {"temperature":128.09, "humidity":62.56,"place":"Timmy"}}
12:21:119.445 -> {"data": {"temperature":128.09, "humidity":62.59,"place":"Timmy"}}
12:21:124.436 -> {"data": {"temperature":128.08, "humidity":62.65,"place":"Timmy"}}
12:21:129.430 -> {"data": {"temperature":128.11, "humidity":62.67,"place":"Timmy"}}
12:21:134.425 -> {"data": {"temperature":128.05, "humidity":62.64,"place":"Timmy"}}
12:21:139.428 -> {"data": {"temperature":128.09, "humidity":62.61,"place":"Timmy"}}
```

- Create the schema in NETPIE2020 to show temperature, humidity, and LED status from ESP32 in Feed.

- d. Create widgets for temperature and humidity and also LED status.  
Copy and paste each widget properties.

**WIDGET**

TYPE

Text

▼

TITLE

Temperature

SIZE

Regular

▼

VALUE

datasources["Timmy"]["shadow"]["temperature"]

+ DATASOURCE

⌘ .JS EDITOR

INCLUDE SPARKLINE

☐ NO

ANIMATE VALUE CHANGES

YES ☒

UNITS

C

SAVE

CANCEL

**WIDGET**

TYPE

Text

▼

TITLE

Humidity

SIZE

Regular

▼

VALUE

datasources["Timmy"]["shadow"]["humidity"]

+ DATASOURCE

⌘ .JS EDITOR

INCLUDE SPARKLINE

☐ NO

ANIMATE VALUE CHANGES

YES ☒

UNITS

%

SAVE

CANCEL

**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["LED"].publish("@msg/led", "0")` [+ DATASOURCE](#) [JS EDITOR](#)  
Add some Javascript here.

ONCREATED ACTION:   
JS code to run after a button is created

[SAVE](#) [CANCEL](#)

**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["LED"].publish("@msg/led", "255")` [+ DATASOURCE](#) [JS EDITOR](#)  
Add some Javascript here.

ONCREATED ACTION:   
JS code to run after a button is created

[SAVE](#) [CANCEL](#)

**WIDGET**

TYPE: **Gauge**

TITLE: **LED Status**

VALUE: `datasources["Timmy"]["shadow"]["led"]` [+ DATASOURCE](#) [JS EDITOR](#)

UNITS:

MINIMUM: **0**

MAXIMUM: **255**

[SAVE](#) [CANCEL](#)

e. Copy and paste the result from your NETPIE2020 Dashboard here:

