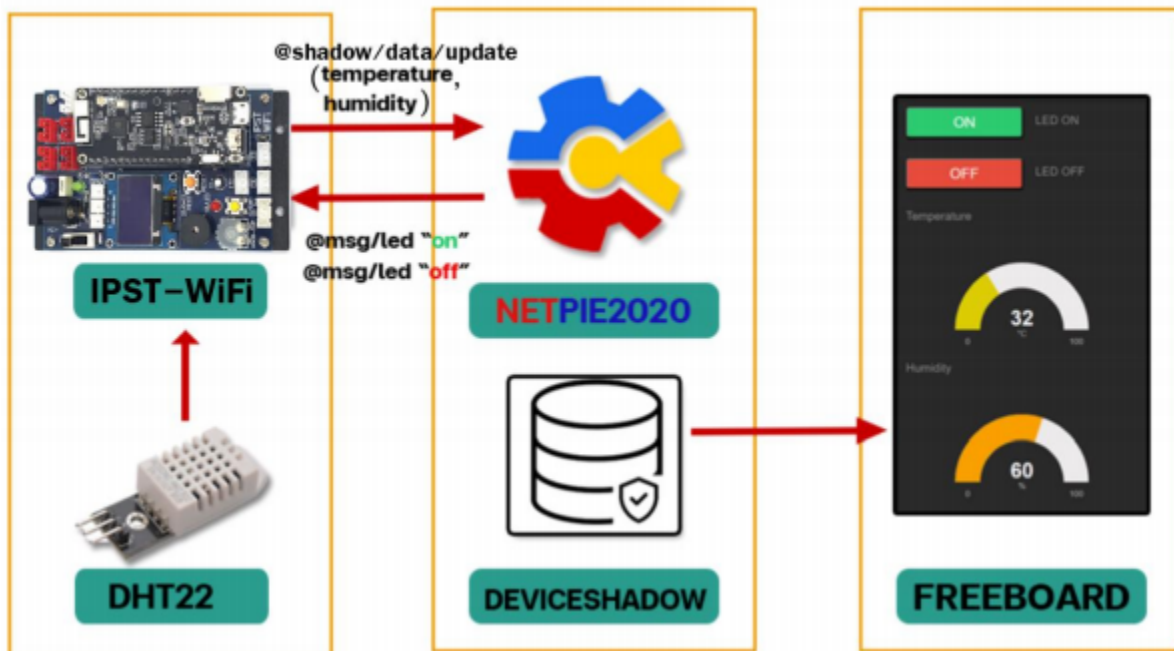


Example of Connecting and Using IPST-WiFi with NETPIE2020

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 IPST-WiFi.



Working model

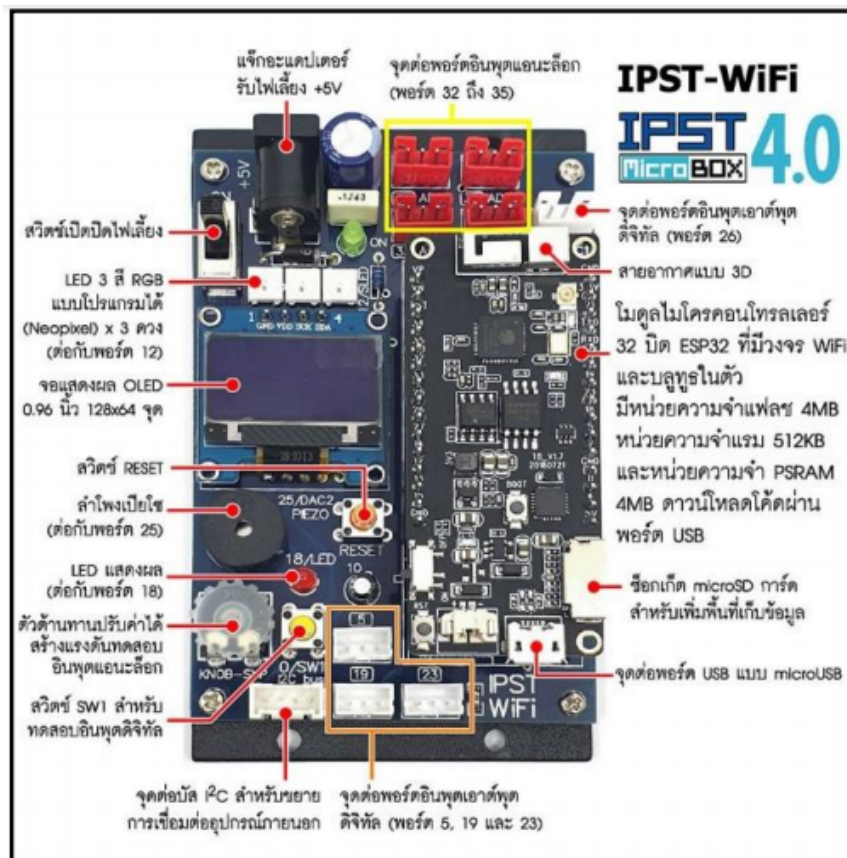
Require Basic Knowledge of

- 1.Using NETPIE2020
- 2.Using NETPIE Freeboard

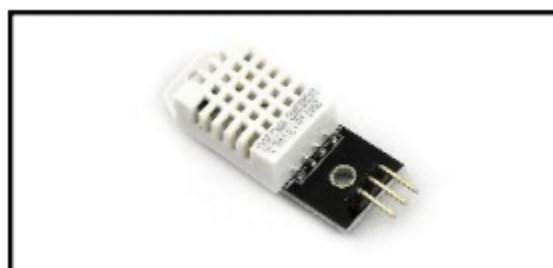
Equipment Used

1.IPST-WiFi

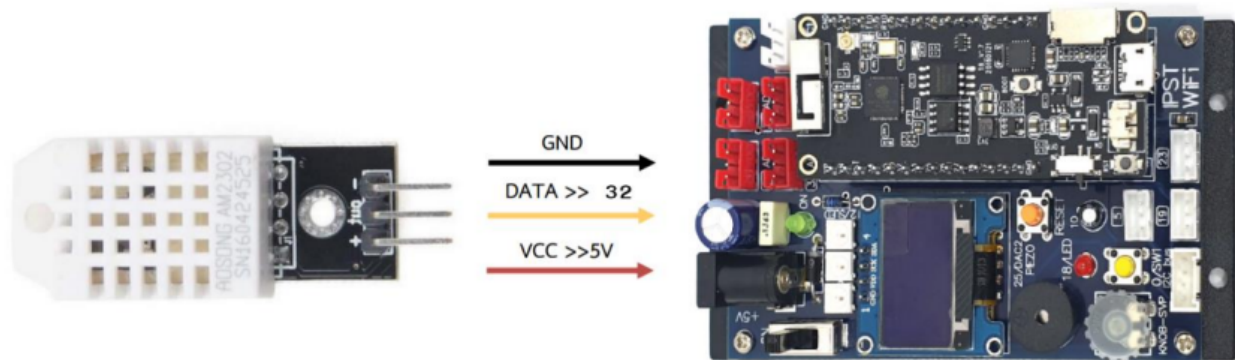
It is a circuit board used to connect to the internet and control the devices. It is equipped with the ESP32 microcontroller from the Espressif systems. It uses a 32-bit dual-core CPU. Each CPU core is referred to as PRO_CPU or Protocol CPU and APP_CPU or Application CPU. It has 448 KB of internal ROM memory, 520 KB of RAM, and 16 KB for a real-time base system built into the chip.



2.DHT22 Temperature and Humidity Sensor



IPST-WiFi Circuit Connection



Circuit connection between DHT22 and IPST-WiFi

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.

```
1 {
2   "additionalProperties": true,
3   "properties": {
4     "humidity": {
5       "operation": {
6         "store": {
7           "ttl": "7d"
8         }
9       },
10      "type": "number"
11    },
12    "temperature": {
13      "operation": {
14        "store": {
15          "ttl": "7d"
16        }
17      },
18      "type": "number"
19    },
20    "led": {
21      "operation": {
22        "store": {
23          "ttl": "7d"
24        }
25      }
26    }
27  }
28 }
```

Ln: 1 Col: 1

Description of Schema

- 1.Stores the first variable 'humidity' of type number in the Timeseries database with a retention period of 7 days.
- 2.Stores the second variable 'temperature' of type number in the Timeseries database with a retention period of 7 days.
- 3.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 32
#define DHTTYPE DHT22
#define LED1 18

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 reconnect function is used to reconnect to the MQTT server if the connection is lost. Once the connection has been established, 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(LED1,1);
      client.publish("@shadow/data/update", "{\"data\" : {\"led\" : \"on\"}}");
      Serial.println("LED on");
    }
    else if (message == "off"){
      digitalWrite(LED1,0);
      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();
}

```

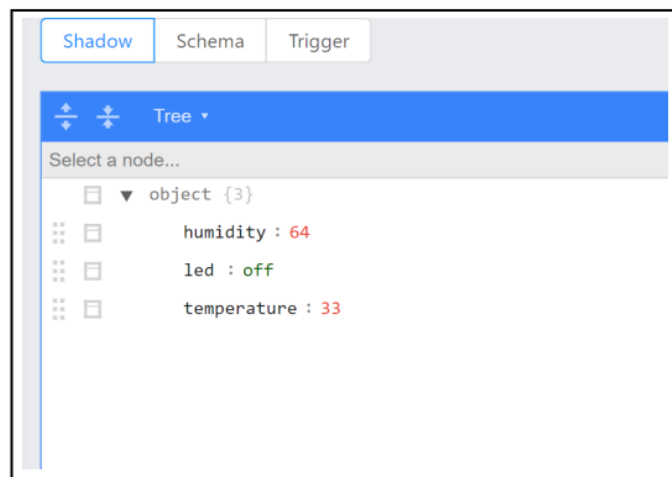
The last part is to define the variables Temperature and Humidity and publish the values on Topic: shadow/data/update for every 2 seconds.

```

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);
}

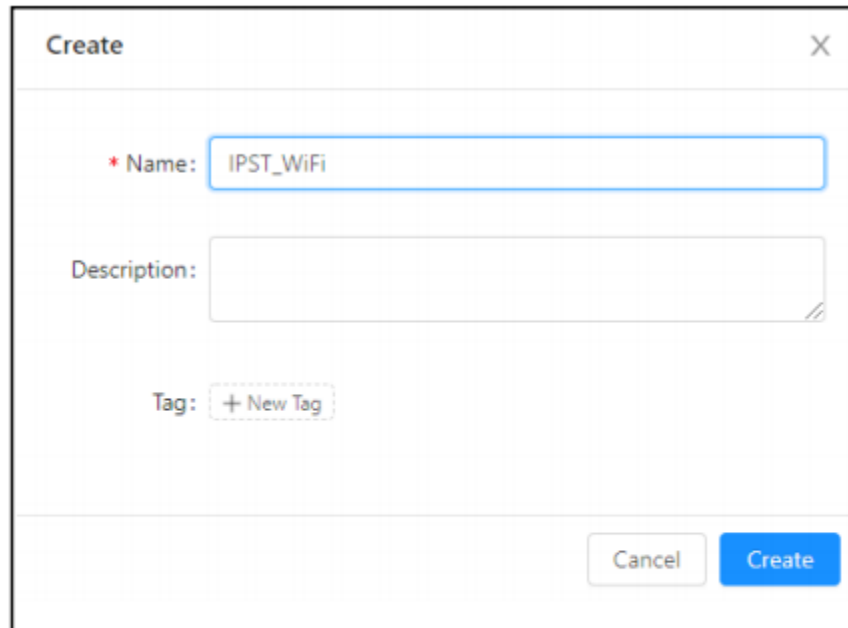
```

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

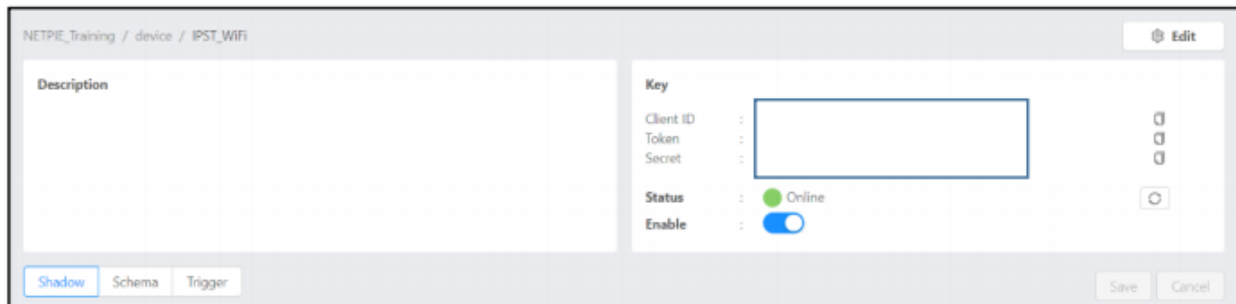


Creating a Device on NETPIE2020

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: a required 'Name' field with the value 'IPST_WiFi', an empty 'Description' text area, and a 'Tag' field with a '+ New Tag' button. At the bottom right are 'Cancel' and 'Create' buttons.

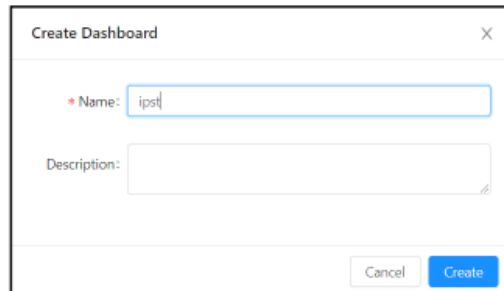


The 'NETPIE_Training / device / IPST_WiFi' configuration page. It features a left sidebar with 'Description', 'Shadow', 'Schema', and 'Trigger' tabs. The main area is divided into 'Key' and 'Status' sections. The 'Key' section has input fields for 'Client ID', 'Token', and 'Secret'. The 'Status' section shows 'Status' as 'Online' with a green dot and 'Enable' as a toggle switch. On the right, there are 'Edit', 'Save', and 'Cancel' buttons.

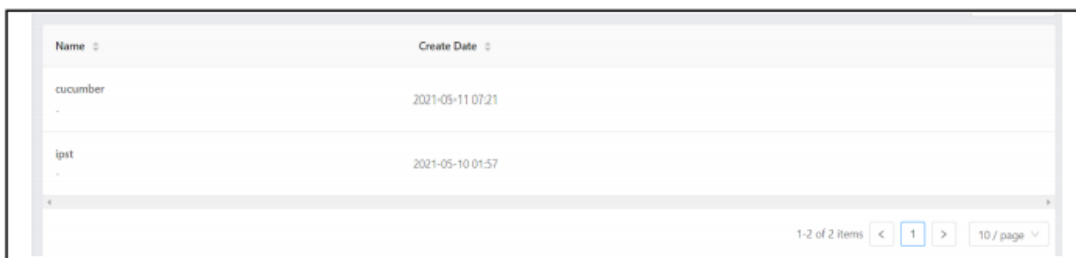
Creating device on NETPIE2020

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.



A dialog box titled "Create Dashboard" with a close button (X) in the top right corner. It contains two input fields: "Name:" with the text "ipst" entered, and "Description:" which is empty. At the bottom right, there are two buttons: "Cancel" and "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 IPST-WIFI.

DATASOURCE

NAME

ipst_wifi

DEVICE ID

Client ID for Device 🔗

DEVICE TOKEN

Token for Device 🔗

SUBSCRIBED TOPICS

Topic 🔗 Subscribe

FEED

YES ☒

SINCE

6

Hour

Display data points since ... ago

DOWN SAMPLING

1

Minute

Resolution of the data points

SAVE

CANCEL

(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.)

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

Humidity

VALUE

`datasources["ipst_wifi"]["shadow"]["humidity"]`

+ DATASOURCE ✕ JS EDITOR

UNITS

%H

MINIMUM

0

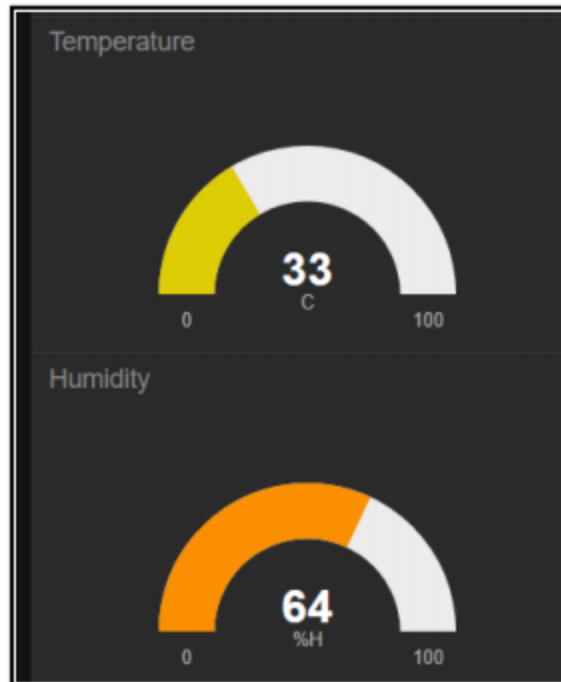
MAXIMUM

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["ipst_wifi"].publish("@msg/led","on")`
+ DATASOURCE X 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

ONCLICKACTION
netpie["ipst_wifi"].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

