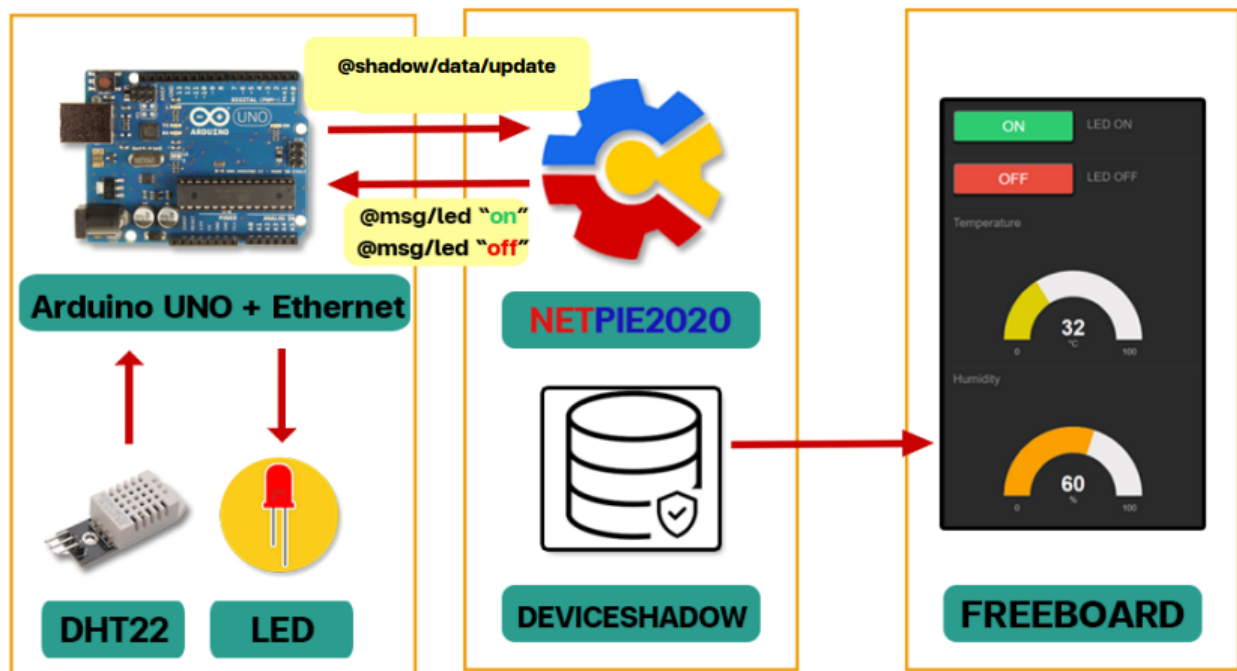


Example of Connecting and Using Arduino Uno + Ethernet with NETPIE2020

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 Arduino Uno + Ethernet.



Working model

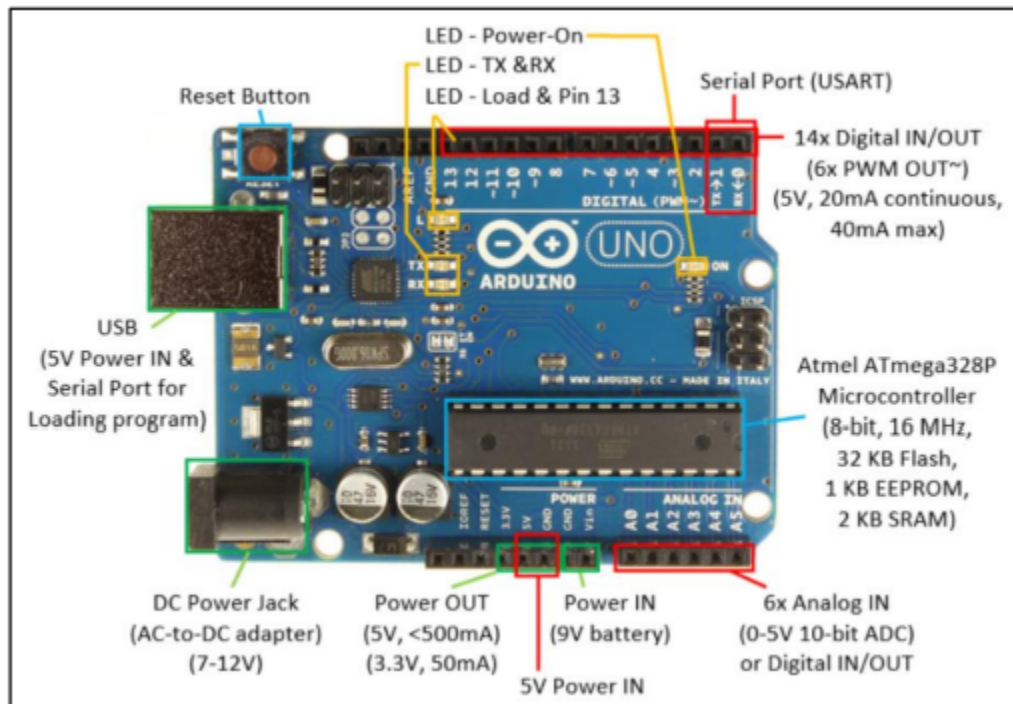
Require Basic Knowledge of

- 1.Using NETPIE2020
- 2.Using NETPIE2020 Freeboard

Equipment Used

1.Arduino Uno + Ethernet

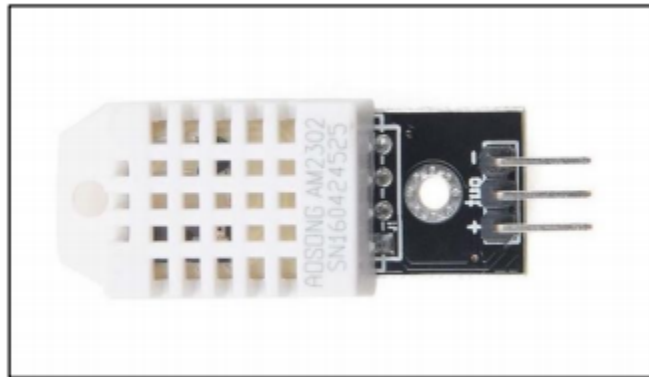
Arduino Uno + Ethernet is the most popular board among the Arduino family, because it's easy to learn. The Uno board comes with different versions such as R2 and R3 depending on the type of the microcontroller used.



Features of Arduino Uno

2.DHT22 Temperature and Humidity Sensor

The DHT22 is a sensor module that measures temperature and humidity. It requires a voltage of 3-5V and can measure temperature ranging from $-40 - 80^{\circ}\text{C}$ (tolerance $\pm 0.5^{\circ}\text{C}$), air humidity ranging from 0 - 100% (tolerance 2 - 5%).



DHT22 sensor

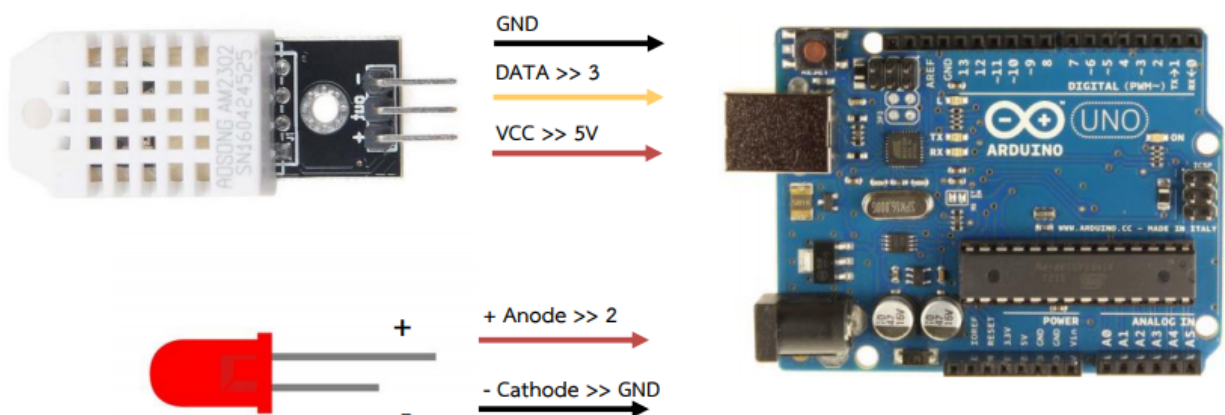
Pin Description

Pin1: Power +Ve (5V Max wrt. GND)

Pin2: Digital I/O interface connected to the microcontroller

Pin3: Power Ground or Power -Ve

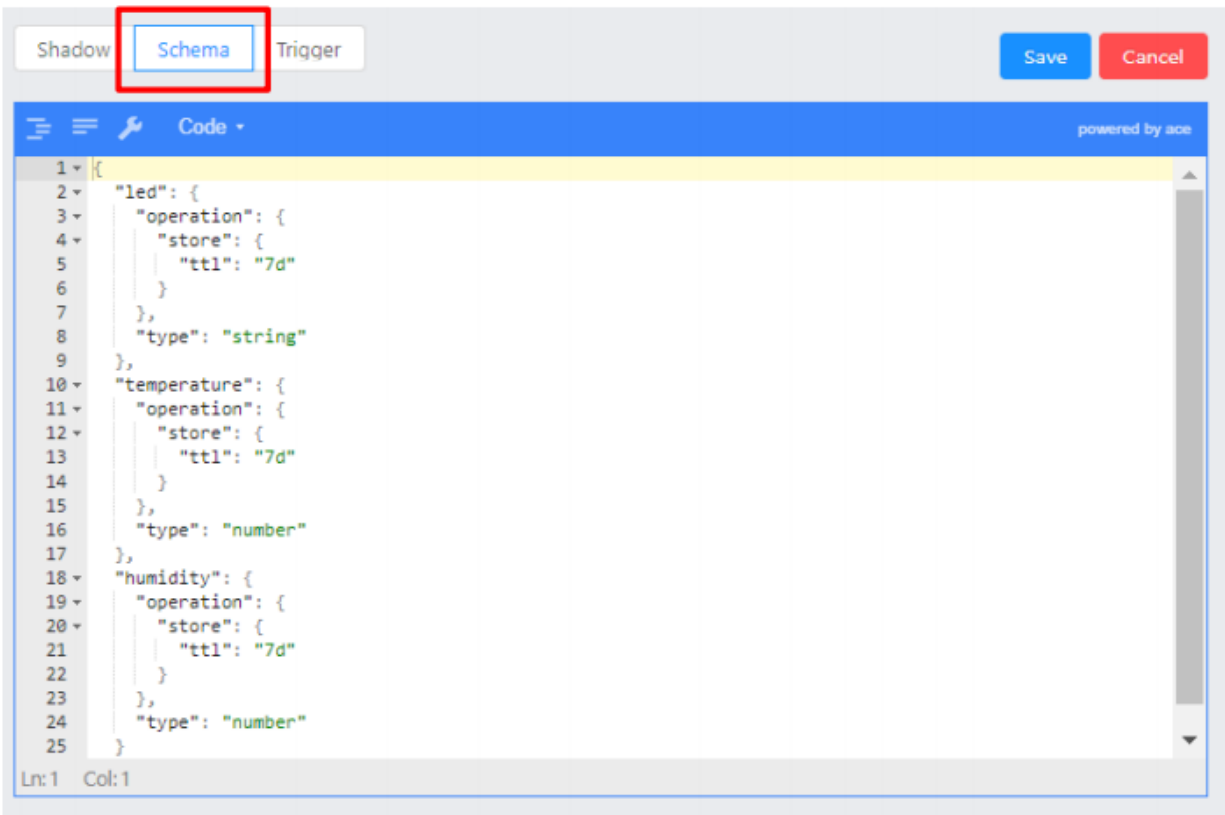
Arduino Uno Circuit Connection



Circuit connection of LED and DHT22 to Arduino Uno

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 'led' of type string 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 'humidity' of type number 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 <Ethernet.h>
#include <PubSubClient.h>
#include <Wire.h>
#include <DHT.h>

#define LED 2
#define DHTTYPE DHT22
#define DHTPIN 3

byte My_MAC_address[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED }; // 2070 MAC address U4UD702D Ethernet Shield
const char* mqtt_server = "broker.netpie.io";
const int mqtt_port = 1883;
const char* mqtt_Client = "2ff5ac36-f337-4bd0-b2f9-575b49cc02b5"; //ArduinoUNOR3
const char* mqtt_username = "keRDpienAkmnFEH4viQ989xTDzU8ZzZo";
const char* mqtt_password = "oJEpjGLICuip$BmmQRqnF~DF*5j#u#-A";
```

The second part is to run the MQTT commands.

```
EthernetClient Etherclient;
PubSubClient client(Etherclient);
DHT dht(DHTPIN, DHTTYPE);
```

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()) {

    if (client.connect(mqtt_Client, mqtt_username, mqtt_password)) {
      Serial.println("connected");
      client.subscribe("@msg/led"); // subscribe Topic ที่ Freeboard ส่งมา
    }

    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);          // [@msg/led]
  Serial.print(" ");

  String message;
  for (int i = 0; i < length; i++) {
    message = message + (char)payload[i];
  }
  Serial.println(message); // on , off

  if(String(topic) == "@msg/led") {
    if (message == "on"){
      digitalWrite(LED,1);
      client.publish("@shadow/data/update", "{\"data\" : {\"led\" : \"on\"}}");
      Serial.println("LED ON");
    }
    else if (message == "off") {
      digitalWrite(LED,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() {
  Serial.begin(115200);
  pinMode(LED, OUTPUT);
  digitalWrite(LED, 0); // เริ่มต้น LED ปิด
  while(Ethernet.begin(My_MAC_address) != 1) // start the Ethernet connection:
  {
    Serial.print(".");
  }
  Serial.print("My IP :");
  Serial.println(Ethernet.localIP());

  client.setServer(mqtt_server, mqtt_port);
  client.setCallback(callback);
  dht.begin(); // start DHT22
}

```

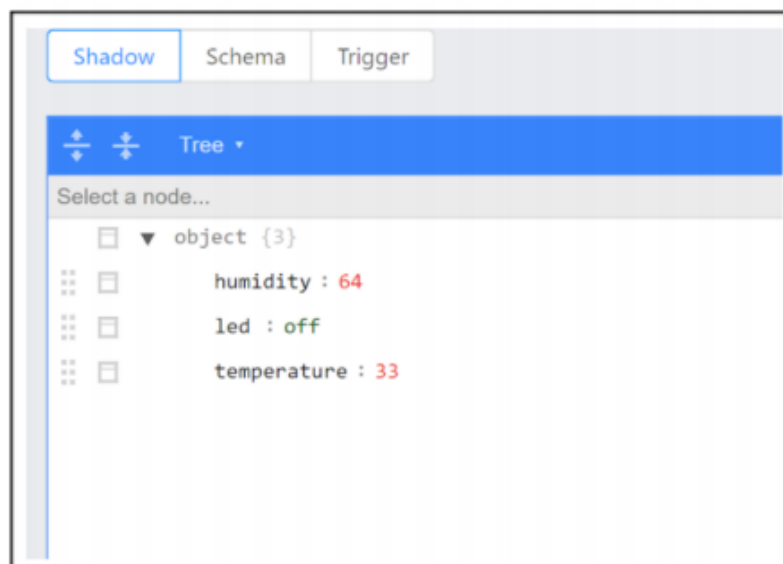
The last part is to define the variables Temperature and Humidity and publish the values on Topic: shadow/data/update for every 2 seconds.

```

void loop() {
  if (!client.connected()) {
    reconnect();
  }
  client.loop();
  int humidity = dht.readHumidity();
  int temperature = dht.readTemperature();
  String data = "{\"data\": {\"temperature\": " + String(temperature) + ", \"humidity\": \"" + String(humidity) + "\"}}";
  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.

The image shows two parts of the NETPIE2020 interface. The top part is a 'Create' dialog box with the following fields:

- Name:** A text input field.
- Description:** A text input field.
- Tag:** A button labeled '+ New Tag'.
- Buttons:** 'Cancel' and 'Create' buttons at the bottom right.

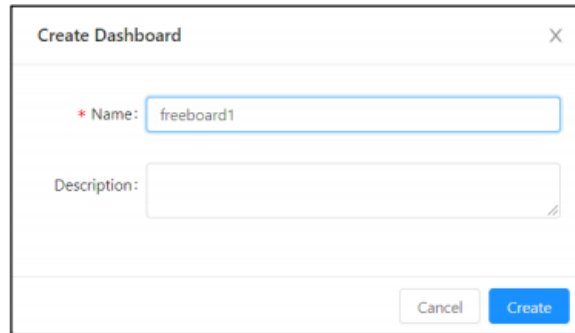
The bottom part is the main interface for a device named 'ArduinoUNOR3'. It includes a breadcrumb 'NETPIE_Training / device / ArduinoUNOR3' and an 'Edit' button. The interface is divided into two sections:

- Description:** A large empty text area.
- Key:** A section containing the following information:
 - Client ID :** 2ff5ac36-f337-4bd0-b2f9-575b... (with a copy icon)
 - Token :** keRDpienAkmnFEH4viQ989xTD... (with a copy icon)
 - Secret :** oJEpjGLICuip\$BmmQKqnF~DF*... (with a copy icon)
 - Status :** Online (indicated by a green dot and a refresh icon)
 - Enable :** A toggle switch that is currently turned on.

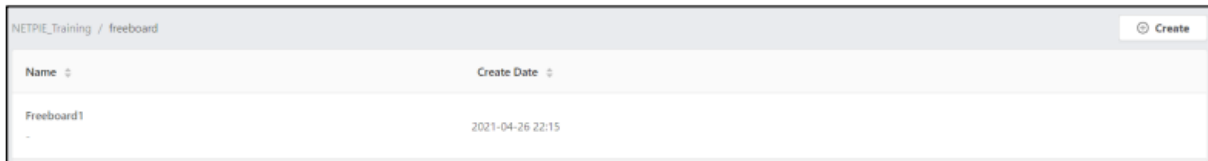
Creating device on NETPIE2020

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 value "freeboard1" and "Description:" which is empty. At the bottom right, there are two buttons: "Cancel" and "Create".



A table with a header row and one data row. The header row has columns "Name" and "Create Date". The data row shows "Freeboard1" and "2021-04-26 22:15".

Name	Create Date
Freeboard1	2021-04-26 22:15

2. Fill in the details, which include Name, Device ID, and Device Token that can be found in the DeviceList named ArduinoUNOR3.

DATASOURCE

NAME

ArduinoUNOR3_D3

DEVICE ID

2ff5ac36-f337-4bd0-b2f9-575b49cc02b5

Client ID for Device [ArduinoUNOR3_D3](#)

DEVICE TOKEN

keRDpienAknnFEH4viQ989xTDzU8ZzZo

Token for Device [ArduinoUNOR3_D3](#)

SUBSCRIBED TOPICS

Topic [ArduinoUNOR3_D3](#) 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

Temperature

VALUE

datasources["ArduinoUNOR3_D3"]["shadow"]["temperature"]

+ DATASOURCE

JS EDITOR

UNITS

°C

MINIMUM

0

MAXIMUM

100

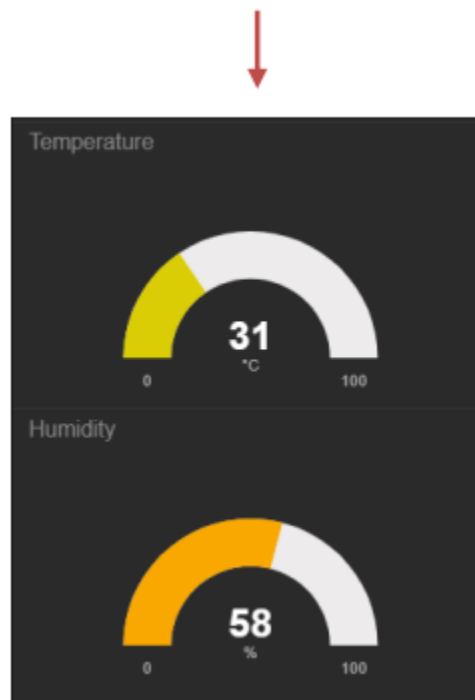
SAVE

CANCEL

WIDGET

TYPE	Gauge	+	DATASOURCE	JS EDITOR
TITLE	humidity			
VALUE	datasources["ArduinoUNOR3_D3"]["shadow"]["humidity"]			
UNITS	%			
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	<code>netpie["ArduinoUNOR3_D3"].publish("@msg/led", "on")</code>	+ DATASOURCE JS EDITOR
	<small>Add some Javascript here.</small>	
ONCREATED ACTION		
	<small>JS code to run after a button is created</small>	

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	<code>netpie["ArduinoUNOR3_D3"].publish("@msg/led", "off")</code>	+ DATASOURCE JS EDITOR
	<small>Add some Javascript here.</small>	
ONCREATED ACTION		
	<small>JS code to run after a button is created</small>	

SAVE CANCEL

Settings for OFF button



ON	LED ON
OFF	LED OFF