Make two arduino communicate using Pubnub

Made by: Youssef EL KHACHANI

Objective:

In this project, my goal is to establish real-time communication between two Arduinos using the PubNub platform to exchange data seamlessly.

Tools:

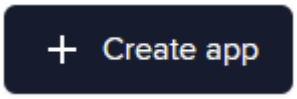
- Two Arduinos
- Two Ethernet shields
- PubNub account
- I am using the university's LAN network, where two MAC addresses and IP addresses are already assigned to the Arduino shields.
- Arduino IDE

PubNub is an IoT platform that stands out for its real-time capabilities. In the context of an IoT platform, the notion of real-time refers to ensuring a maximum latency of 250 ms between the publication of a message and its reception by the recipient. Watch the following video to better understand the capabilities of PubNub.

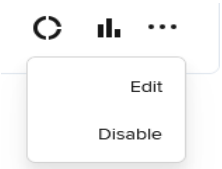
This platform is a real-time data network that allows users to develop applications while reducing development costs and the complexity of the task. It provides a web services API that enables the creation of scalable applications on mobile phones, tablets, and the web. Although the first applications were focused on finance and marketing contexts, it is now possible to connect many types of connected devices to PubNub.

Creating a PubNub Account:

Go to the PubNub website (Try it for free) and create an account (Sign in) using an email, password, and by filling out a brief form. After verifying your "humanity," an interface will allow you to create an application in addition to the standard "Demo Project."



First, disable the "Demo Project" application by clicking on "Disable."

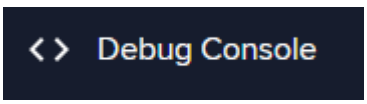


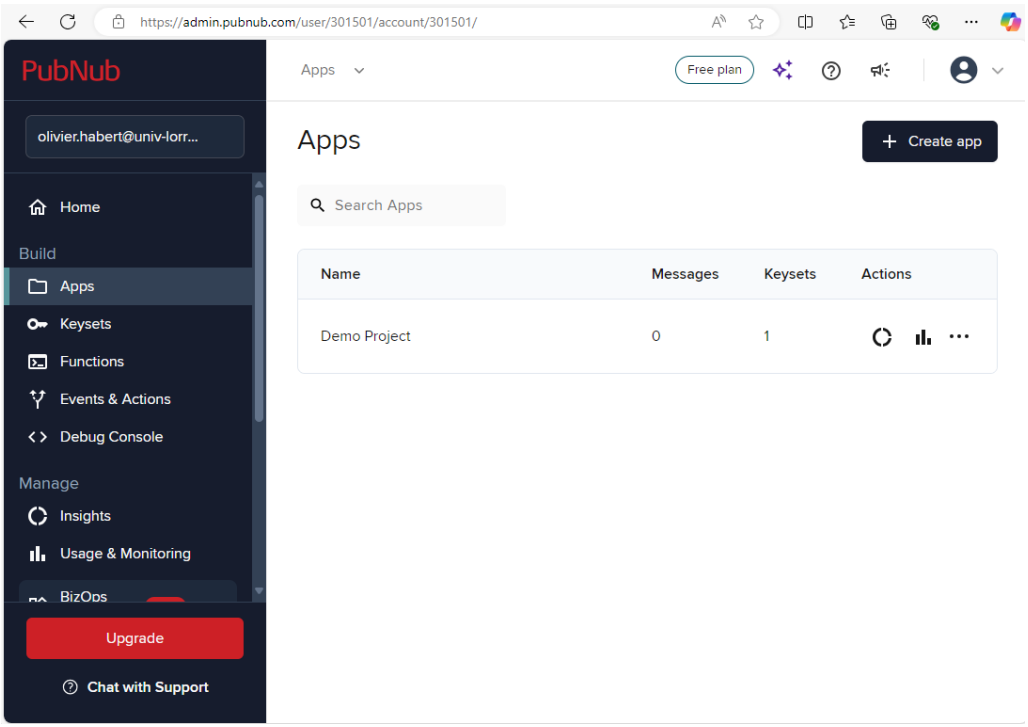
Name the new application AppTPIOT. In the "Search Keyset" section, click on "AppTPIOT Keyset" to retrieve the publish and subscribe keys.

You will need these keys for the connected devices.

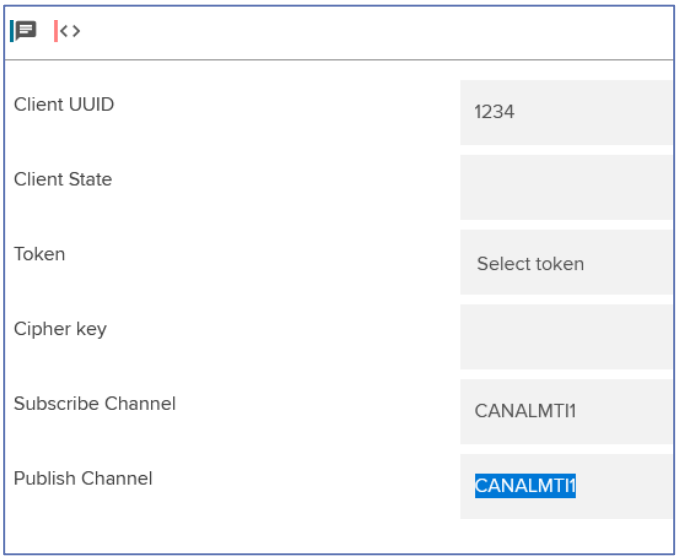
We will initially work within the PubNub console interface. On the left vertical toolbar, click on the icon to open the PubNub console. Select AppTPIOT and its keysets.

In the left vertical toolbar, click on ‘’ Debug Console’’ to display the PubNub console. Select AppTPIOT and its keysets.





Add a client by filling in the form as shown in the following figure+ The channel name



Connecting a Device to PubNub :

We will connect an Arduino to PubNub. For this, you will use Arduino Uno boards with an Ethernet Shield. Each device must have a unique MAC address (Media Access Control) on the network, which will allow the university's DHCP server to automatically assign it an IP address.

Arduino IDE Code: Turning an Arduino Board with an Ethernet Module into a Local Web Server

so what we try to do now is to connect the arduino and its shield to LAN network that we use (I use the LAN of the university the addresses that I use are already created by the IT of the university) this code should be used for one arduino and not both.

```
#include <SPI.h>

#include <Ethernet.h>

 #include <PubNub.h>

byte mac[] = { 0xXX, 0xXX, 0xXX, 0xXX, 0xXX, 0xXX };

char pubkey[] = "pub-c-XXXXXXXXXXXXXXXXXXXXXX"; char subkey[] = "sub-c-df4e6906-fa0f-11e8-9231-4abfa1972993";

char channel[] = "CanalMTI1";

void setup()

{
```

```
{
Serial.println("Ethernet setup error"); delay(1000);
}
Serial.println("Parametrage d'Ethernet");
PubNub.begin(pubkey, subkey);
Serial.println("Parametrage de PubNub");
}
void loop()
{
EthernetClient *client;
Serial.println("Publication d'un Message");
Serial.println("{\"BONJOUR\": \"Je suis l'Arduino 1 qui publie\"}");
client = PubNub.publish(channel, "{\"BONJOUR\": \"Je suis l'Arduino 1 qui publie\"}");
if (!client)
{
Serial.println("publishing error");
delay(1000);
}
while (client->connected()) {
while (client->connected() && !client->available()) ;
 char c = client->read();
Serial.print(c);
}
client->stop();
Serial.println();
 delay(10000);
```

}

Key Elements:

- #include <Ethernet.h>:This library is used to manage network connections (such as MAC address, IP address, web server, etc.).
- byte mac[] & IPAddress ip(xxxx):These define the MAC address and IP address for the Ethernet connection.
- EthernetServer server(80):
- A web server is created on port 80. A web server listens for incoming client requests and responds by sending content (usually an HTML page).
- setup():
    - Serial communication initialization: The Arduino communicates with the computer to display messages in the serial monitor.
    - Network initialization: The Ethernet connection is initialized with the defined MAC and IP addresses.
- Ethernet.begin(mac, ip):This initializes the Ethernet connection using the predefined MAC address and IP.
- server.begin(): This starts the web server and makes it listen for incoming connections on port 80.
- Ethernet.localIP():This command displays the IP address assigned to the Arduino, allowing users to know where to connect to access the web server.
- loop(): This function runs in a continuous loop on the Arduino. It performs the following actions:
    - Waits for an HTTP client (e.g., a web browser) to connect.
    - Reads and displays the HTTP request sent by the client through the serial monitor.
    - Sends an HTTP response with an HTML page to the client.
    - Closes the connection with the client after sending the response.

The code of the first arduino that will send tha data (random) to the pubnub

```
#include <SPI.h>

#include <Ethernet.h>

#include <PubNub.h>

#include <ArduinoJson.h>

byte mac[] = { 0x90, 0xA2, 0xDA, 0x11, 0x00, 0x26 };

char pubkey[] = "pub-c-22a8c323-51b8-4372-ab41-a9cc87d6f140";
```

```
char subkey[] = "sub-c-4b1e35c2-bc6b-4225-a6ff-35c3bd07a80c";
char channel[] = "CanalMTI6";
JsonDocument doc;
String Message;
void setup()
{
Serial.begin(9600);
while (!Serial);
while (!Ethernet.begin(mac))
{
Serial.println("Ethernet setup error");
delay(1000);
}
Serial.println("Parametrage d'Ethernet");
//Initialisation de Pubnub
PubNub.begin(pubkey, subkey);
Serial.println("Parametrage de PubNub");
}
void loop()
{
EthernetClient *client;
Serial.println("Serialisation");
JsonObject BME680 = doc["BME680"].to<JsonObject>();
BME680["Temperature"] = 23.85;
BME680["Pression"] = 1013.65;
BME680["Humidite"] = 33.8;
```

```
BME680["Pollution"] = 14.189;

serializeJson(doc, Message);

serializeJsonPretty(doc, Serial);

Serial.println();

client = PubNub.publish(channel, Message.c_str());

if (!client) {

Serial.println("publishing error");

delay(1000);

return;

}

// Récupération de la chaine de retour de la plateforme PubNub

while (client->connected()) {

while (client->connected() && !client->available()) ;

char c = client->read();

Serial.print(c);

}

client->stop();

Serial.println();

delay(10000);

}
```

Objective: Publishing a Message on PubNub

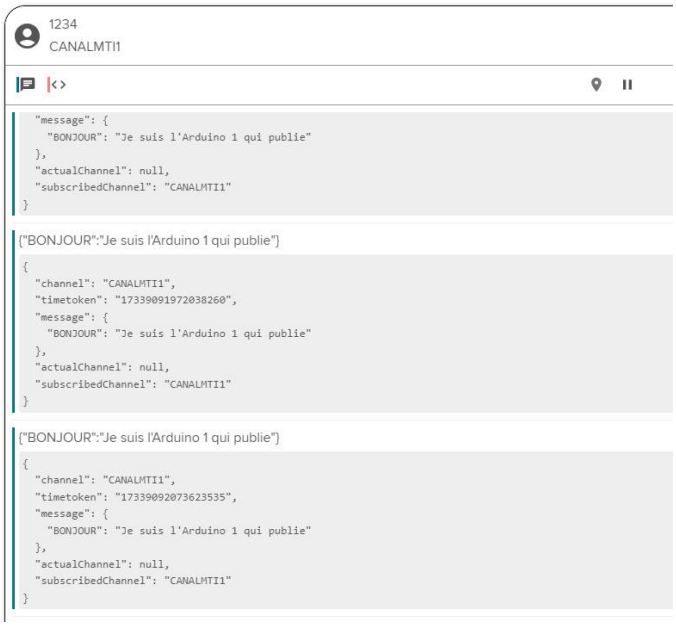
In this task, we will publish the message "Bonjour: Je suis l'arduino qui publie" to PubNub in JSON format. Below is a detailed explanation of the code.

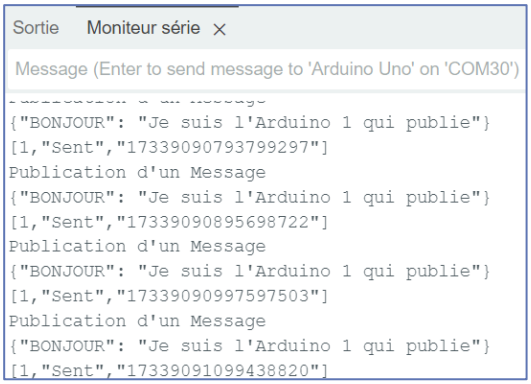
Key Elements:

- #include <PubNub.h>:This library is used to interact with the PubNub service and send messages to the platform.

- Variable Declarations:
    - byte mac[]: Defines the MAC address of the Arduino for network communication.
    - char pubkey[]: The PubNub publish key used to send data to PubNub.
    - char subkey[]: The PubNub subscribe key used to receive data from PubNub (not required for this task but usually needed for full interaction).
    - char channel[]: Defines the name of the PubNub channel (e.g., CanalMTI1) where messages will be published.
- setup():This function initializes PubNub with the provided publish and subscribe keys, preparing the Arduino to send and receive messages.

- loop():
    - Display a Message: A message is shown in the serial monitor to signal that the Arduino is about to send a message.
    - Send JSON Message: A JSON message ("Bonjour: Je suis l'arduino qui publie") is sent to PubNub on the specified channel (CanalMTI1).
    - Error Handling: If the message is not successfully sent, an error message is displayed, and the Arduino will attempt to resend after 1 second.
    - Check for Response: The loop waits for a response from PubNub to confirm that the message was successfully sent.
    - Read and Display Response: The response is read and displayed character by character in the serial monitor.
    - Close Connection: After receiving a response, the connection is closed.
    - Delay: The Arduino waits for 10 seconds before attempting to send the message again.





Receiving a Message on a Second Arduino:

```
#include <ArduinoJson.h>
#include <SPI.h>
#include <Ethernet.h>
#include <PubNub.h>
byte mac[] = {0xXX, 0xXX, 0xXX, 0xXX, 0xXX, 0xXX};

char pubkey[] = "pub-XXXXXXXXXXXXXXXXXXXXXX";
char subkey[] = "sub-c-XXXXXXXXXXXXXXXXXXXXX";
```

```
char channel[] = "CanalMTI6";
void setup()
{
 //Initialisation du moniteur série
 Serial.begin(9600);
 //Initialisation d'Internet
 while (!Ethernet.begin(mac))
 {
 Serial.println("Ethernet setup error");
 delay(1000);
 }
 Serial.println("Parametrage d'Ethernet");
 //Initialisation de Pubnub
 PubNub.begin(pubkey, subkey);
 Serial.println("Parametrage de PubNub");
}
void loop()
{
//Ethernet.maintain();//Demande au serveur DHCP de maintenir l'adresse IP
EthernetClient *client;
Serial.println("ARDUINO 2 attend la réception d'un message");
PubSubClient *pclient = PubNub.subscribe(channel);
if (!pclient)
{
Serial.println("subscription error");
delay(1000);
return;
}
String Message = "";
StaticJsonDocument<256> doc;

while (pclient->wait_for_data())
{

char c = pclient->read();
// message = message.toString();
Message.concat(c);
}

pclient->stop();
deserializeJson(doc, Message);
```

```
serializeJsonPretty(doc, Serial);
//Serial.print(Message);
Serial.println(Message);
delay(500);
}
```

In this task, we will set up a second Arduino to subscribe to a PubNub channel (CanalMTI1) and receive messages that are sent by the first Arduino.

Below is the detailed explanation of the code, which includes handling PubNub subscription, receiving JSON messages, and processing them.

Key Elements:

- PubSubClient *pclient = PubNub.subscribe(channel);: This line subscribes the second Arduino to the PubNub channel named "CanalMTI1". The subscribe() function allows the Arduino to receive messages published to this channel.

- Error Handling:
  - If the subscription fails, the message "subscription error" is printed to the serial monitor, and the program waits for 1 second before trying to subscribe again.
- Reading the Received Message:
  - A variable Message is declared to store the incoming data.
  - A StaticJsonDocument<256> is used to parse the JSON data that the Arduino will receive.
  - The while (pclient->wait_for_data()) loop waits for incoming data and adds each received character to the Message variable.
- Processing the JSON Message:
  - After the full message is received, the connection to PubNub is closed using pclient->stop();.

Results:

```
Parametrage d'Ethernet
Parametrage de PubNub
Récupération des messages stockés
[{"BONJOUR":"Je suis l'Arduino 1 qui publie"}
,{"BONJOUR":"Je suis l'Arduino 1 qui publie"}
,{"BONJOUR":"Je suis l'Arduino 1 qui publie"}
,{"BONJOUR":"Je suis l'Arduino 1 qui publie"}
,{"BONJOUR":"Je suis l'Arduino 1 qui publie"}
,{"BONJOUR":"Je suis l'Arduino 1 qui publie"}
,{"BONJOUR":"Je suis l'Arduino 1 qui publie"}
,{"BONJOUR":"Je suis l'Arduino 1 qui publie"}
]
```