MQTT protocol & RFID raspberry and ESP8266

FÁBIO GONÇALVES[1] and JULIEN JUNCKER[2]

University of technologies, CASTELO BRANCO, Portugal,
julien.juncker@viacesi.fr

Abstract. In this document, we will talk about installation of MQTT protocol and RFID system. The goal is to read RFID card with ESP8266 and to send ID to rapsberry with MQTT protocol.

Keywords: MQTT, RFID, ESP8266, raspberry, linux, port forwarding

1 Installation and configuration

First, we need to install arduino IDE on windows and to configure it for ESP8266. Also, we will install MQTT protocol on raspberry and open port on router.

1.1 Installation on arduino IDE

To install arduino IDE, open web browser and go to this link: https://www.arduino.cc/en/Main/Software and download the version adapted to operational system (exist versions for Windows, Linux and Mac Os X). Install the program using the package downloaded in Arduino.cc Web Site.
After the package download, start the installation, running the installation file on the system. It will open a window with installation options, select all of them.
After this step, select the file location on destination folder by click on button

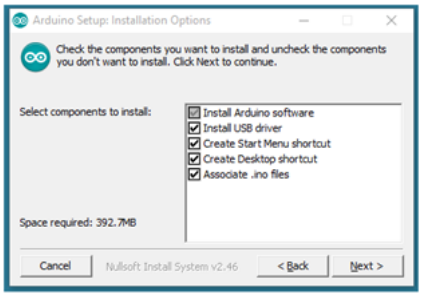

Fig. 1. Installation Options

"Browse ...", or click on install.
The Installation will proceed, and the installer will extract all necessary files to execute the "Arduino Software (IDE)".
The installation will finish and it's now possible to start Arduino IDE clicking

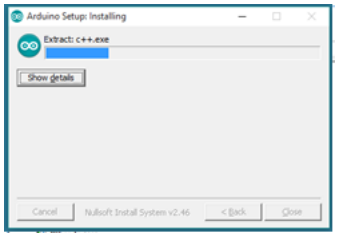

Fig. 2. Arduino IDE installing

on shortcut on Desktop or search in windows start menu by "Arduino".

1.2 Installation DRIVERS FOR ESP8266

Probably in this step, Node MCU Board are not recognized for windows, so it's necessary download the drivers using an executable file that could be downloaded here:
http://www.wch.cn/download/CH341SER_EXE.html note that the site could be in a Chinese version.

For an "official" driver installation on Windows 10 it's need to access the "Device manager" clicking with right button mouse on start menu and select "device manager", an windows will be open and its possible to see a "trouble" in "other devices", as you can see on Fig. 3.
By click with the right button on there, it's possible to "install controllers"

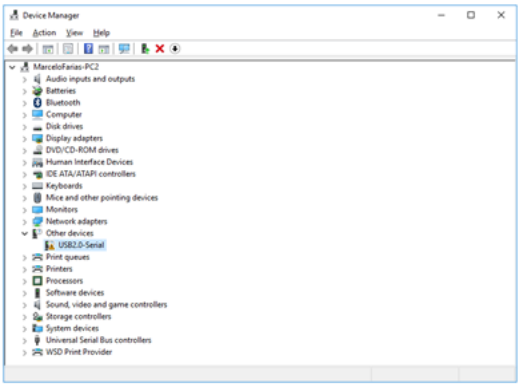

Fig. 3. Windows Device Manager

and windows will search a specific official driver. Alternatively, it's possible to update controllers, and windows will search by the last software drivers.
After these steps, the "Device Manager" will update and now it's possible to see an update windows that contains ports corresponding to Node MCU Board registered as COM & LPT ports, in this example the connected port its COM 21, note that COM Port number is automatically administered by the operation system.

1.3 CONFIGURE IDE TO ALLOW PROGRAMMING NODE MCU BOARD

In this phase is now possible to start the configuration of Arduino IDE to programming Node MCU Board. It means that the configuration of Arduino doesn't support by default the board corresponding to ESP8266, so it has a lot of configurations to do essentials to programming and access the node MCU board.

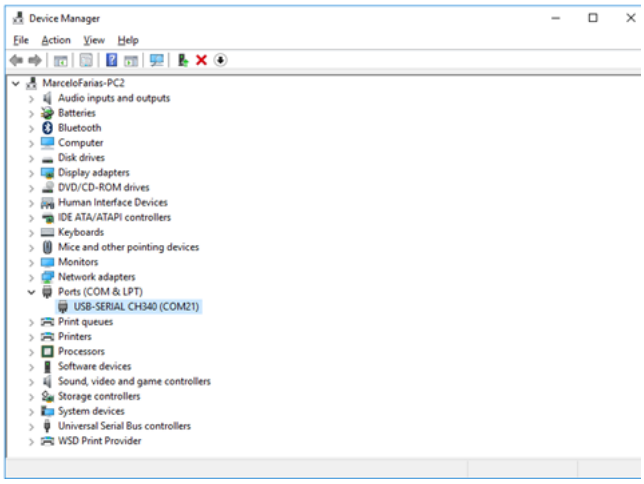

Fig. 4. Device manager updated with the installed Controllers

Note that this configurations steps allow the possibility to access some example files of codes that are not "accessible" for default on Arduino IDE.
In first step launch the Arduino IDE by clicking on IDE shortcut on desktop if it creates or launch by start menu on windows system by typing "Arduino".
After running of Arduino IDE go to "File" → "Preferences". It will open the windows of Fig.8 and input this link:
http://arduino.esp8266.com/stable/package_esp8266com_index.json on "Additional Board Manager URLs" as is showed on Fig.5.
The next step is installing the board on Arduino IDE, to do that, go to "Tools" -¿ Board: . . . " → "Board Manager", as it's possible to see on Fig.6.
Then a window called "Board Manager will open and it's possible to search by the board, for that type "esp8266" on search box. When the search is complete, install the board called "esp8266 by esp8266 community" and "PubSubClient" select the last version and click on install button. It is possible to see this step on Fig.7 and Fig.8.
Wait for the program complete the installation. And after close "Board Manager" window and return to Arduino IDE interface. Go to "Tools" → "Board. . . " and select the "NodeMCU 1.0 (ESP-12E Module)" as observed on Fig.8.
Now the last step is select on Arduino IDE the correspondent COM for ESP8266 programming and see the Serial Monitor. For that go Back to Arduino Interface and select "Tools" → "Port. . . " and click on correspondent com of ESP8266 that is the same installed on Fig.4 of the last section of this document , as it's possible to see on Fig.9. Exist the possibility of have different ports registered on Arduino IDE because of other services like Bluethooth.

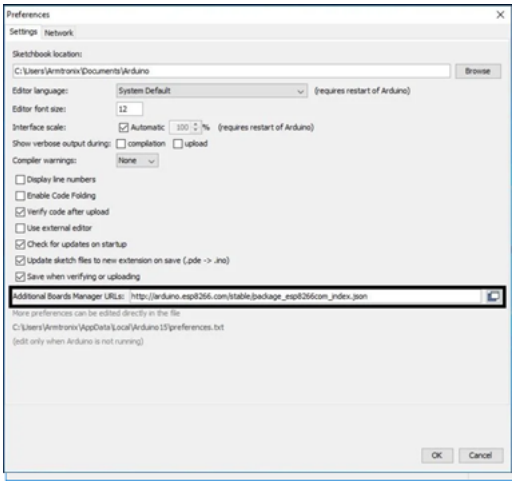

Fig. 5. Arduino Preferences

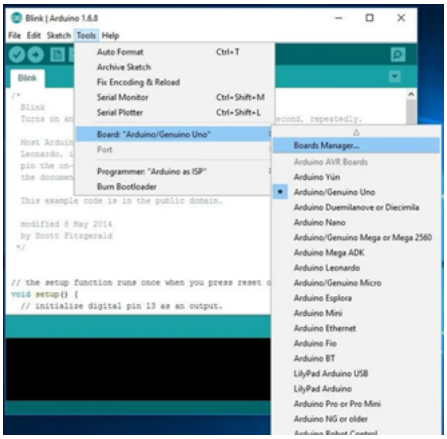

Fig. 6. Arduino Tools → Board Manager

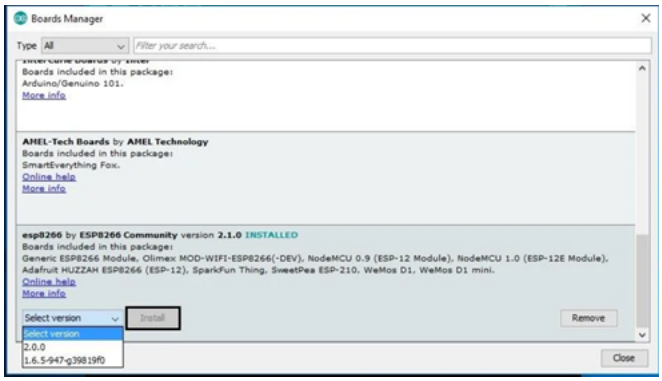

Fig. 7. Install ESP8266

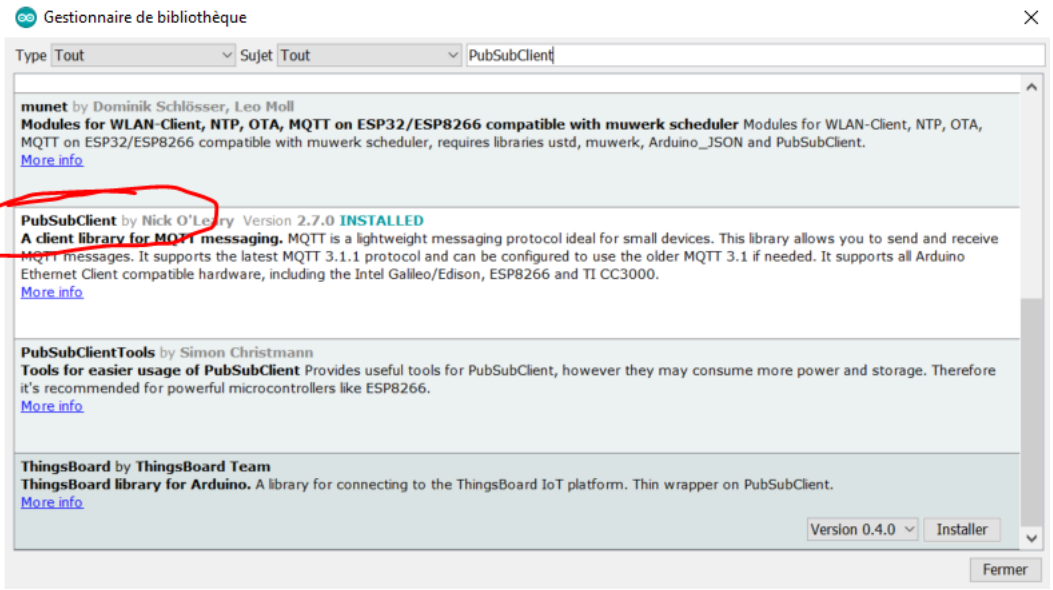

Fig. 8. Install PubSubClient

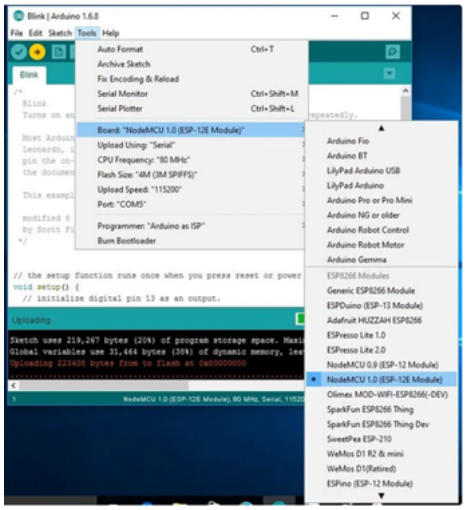

Fig. 9. Board selection

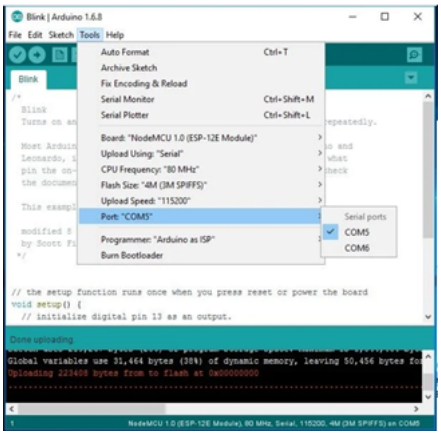

Fig. 10. Select COM port of ESP8266

1.4 INSTALL RFID library on arduino

To use the RFID reader, you need to install MFRC522 library. Go to “Tools” -¿ Board: . . . ” → “Board Manager”, as it’s possible to see on Fig.6.
Write *”mfrc522”* on searchbar and install library.

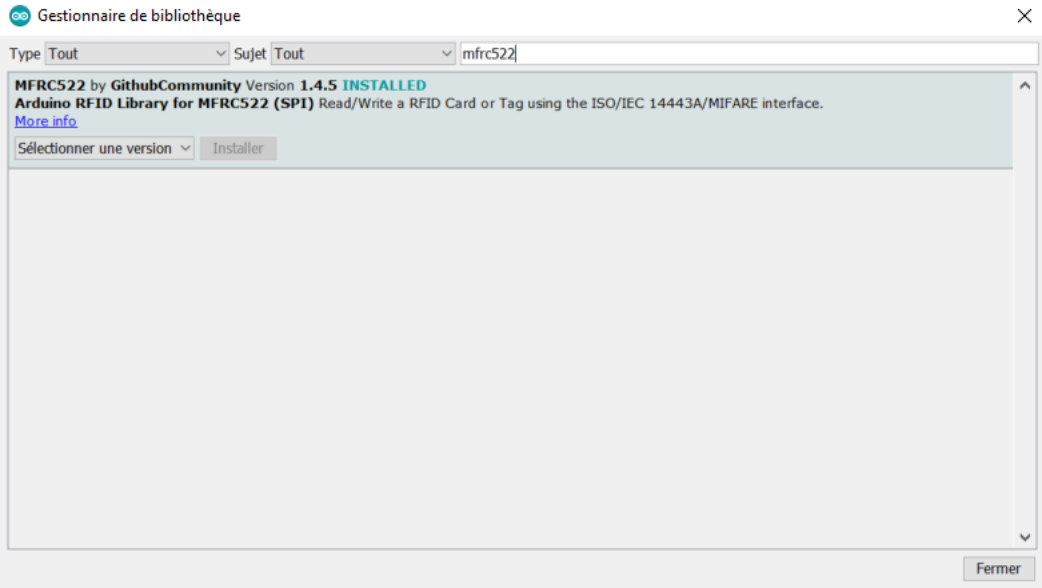

Fig. 11. Installing mfrc522 library

1.5 INSTALL AND CONFIGURE RASPBERRY PI AS MQTT BROKER

To setup our MQTT system, we need a broker. For the Raspberry Pi, we will be using the ”Mosquitto” MQTT broker.
We used this for allowing data transmission between Node MCU and Raspberry. Before we install this, it is always best to update our Raspberry Pi.

sudo apt-get update
sudo apt-get upgrade

Once you’ve done this, install mosquitto and then the mosquitto-clients packages.

sudo apt-get install mosquitto -y
sudo apt-get install mosquitto-clients -y

When you’ve finished installing these two packages, we are going to need to

configure the broker. The mosquitto broker's configuration file is located at **/etc/mosquitto/mosquitto.conf**, so open this with your favourite text editor. If you don't have a favourite text editor or don't know how to use any of the command line editors, I'll be using nano so you can follow along:

sudo nano /etc/mosquitto/mosquitto.conf

At the bottom of this file, you should see the line:

include_dir /etc/mosquitto/conf.d

Delete this line. Add the following lines to the bottom of the file.

allow_anonymous false
password_file /etc/mosquitto/pwfile
listener 1883

By typing those lines, we've told mosquitto that we don't want anyone connecting to our broker who doesn't supply a valid username and password (we'll get on to set these in a second) and that we want mosquitto to listen for messages on port number 1883.

If you don't want the broker to require a username and password, don't include the first two lines that we added (i.e. allow_anonymous... and password_file...). If you have done this, then skip to rebooting the Raspberry Pi.

Now close (and save) that file. If you are following along with the nano example, press CTRL+X, and type Y when prompted.
Because we've just told mosquitto that users trying to use the MQTT broker need to be authenticated, we now need to tell mosquitto what the username and password are! So, type the following command - replacing **username** with the username that you would like - then enter the password you would like when prompted (Note: if, when editing the configuration file, you specified a different **password_file path**, replace the path below with the one you used).

sudo mosquitto_passwd -c /etc/mosquitto/pwfile username

As we've just changed the mosquitto configuration file, we should reboot the Raspberry Pi.

sudo reboot

Once the Raspberry Pi has finished rebooting, you should have a fully functioning MQTT broker!

1.6 PORT FORWARDING ON VIRTUAL SERVER

The port forwarding allow to access an specific port by outside of the router. So, forward packets that are sent from one port to another.

If you want more details of this part, see report *raspberry_WebServer_installation*.

In this procedure, I used the Asus router WL-500G prenium V2. First you need to connect your raspberry to the router and connect to HTTP://192.168.1.1/. The default user and password of Asus router is **ADMIN**.
In the main page of configuration, click on **CLIENT ICON** to see all private

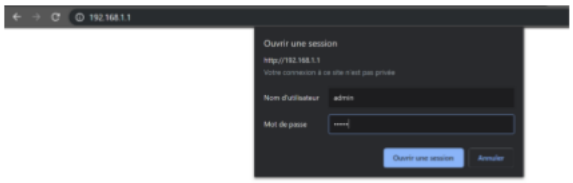

Fig. 12. Connection to the Asus router

IP of connected devices. Keep in mind the adress of Raspberry. In this exemple, It's 192.168.1.12.
When you are logged to the Asus configuration webportal, you go to *ADVANCED SETTINGS* → *WAN*. After that, you go on *VIRTUAL SERVER* tab.
Click to *"YES"* on *"ENABLE VIRTUAL SERVER"*. You'll add the **1883** PORTS to the virtual server list. You need to put the IP of raspberry that I introduced you before. You can keep TCP PROTOCOL for this ports. When you're done the addition of ports, you click of *APPLY BUTTON* to implement changes.
Now, when you want to communicate with public IP of your router and MQTT protocol, the request automatically redirect on your raspberry.

2 programming communication

In this part, we will see how to create program to communicate with MQTT protocol. First, we just try to send a message, after that we send RFID number.

2.1 Simple MQTT communication program

The first program is to test MQTT communication. For that you need to put *Simple MQTT communication program/PythonMQTT_Subscribe.py* on your raspberry and change the value:

- mqtt_username

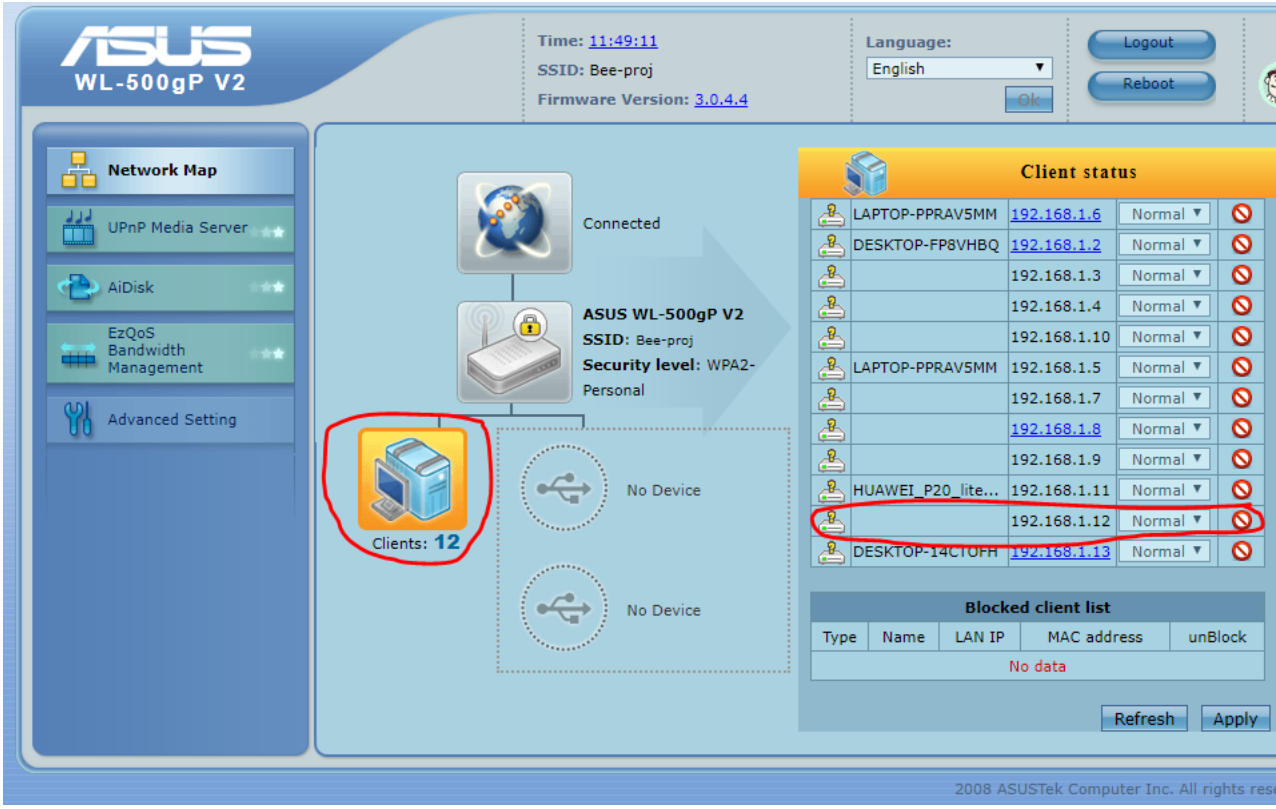

Fig. 13. IP of raspberry in Asus configuration menu

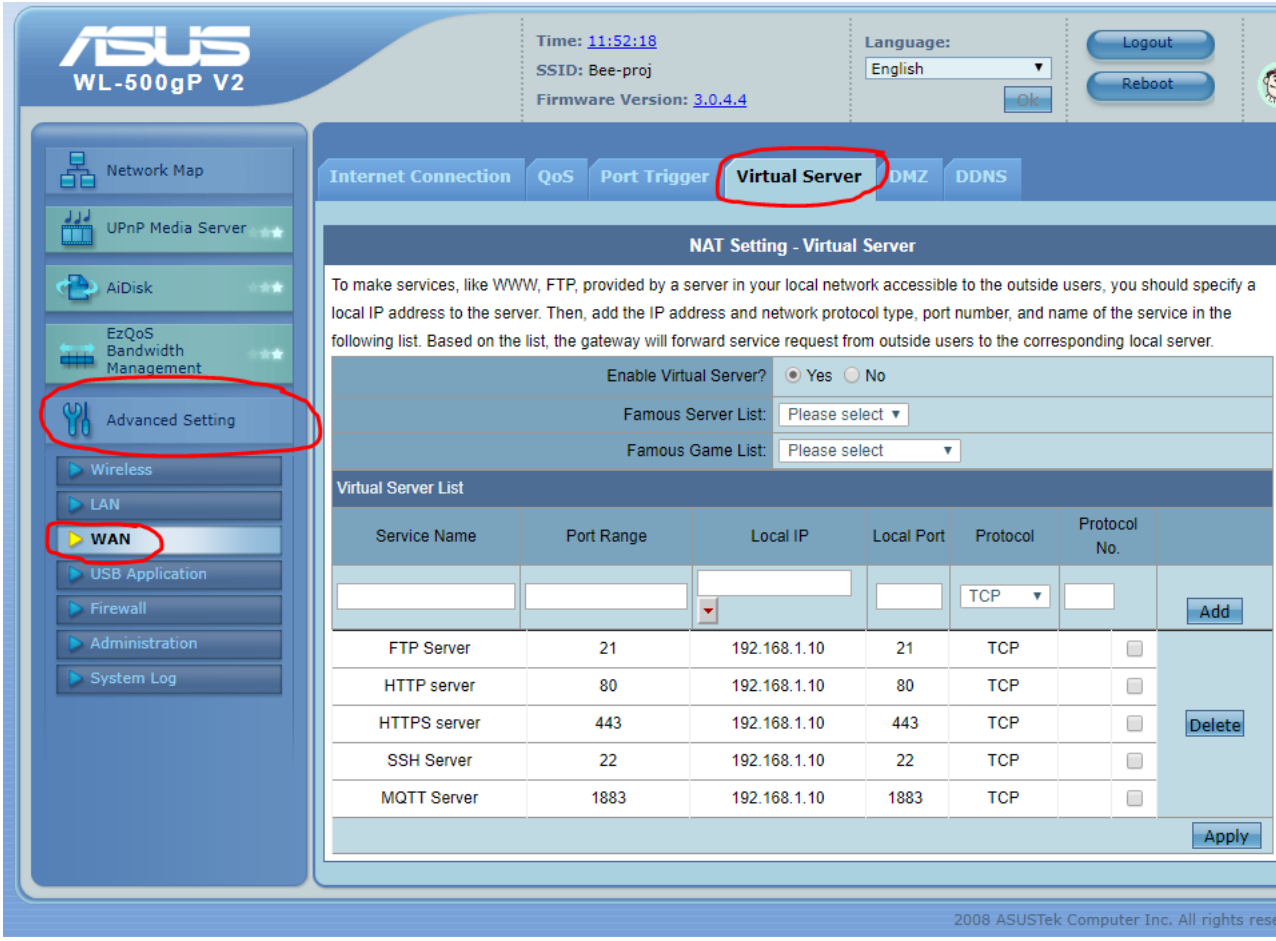

Fig. 14. Path to Port forwarding configuration

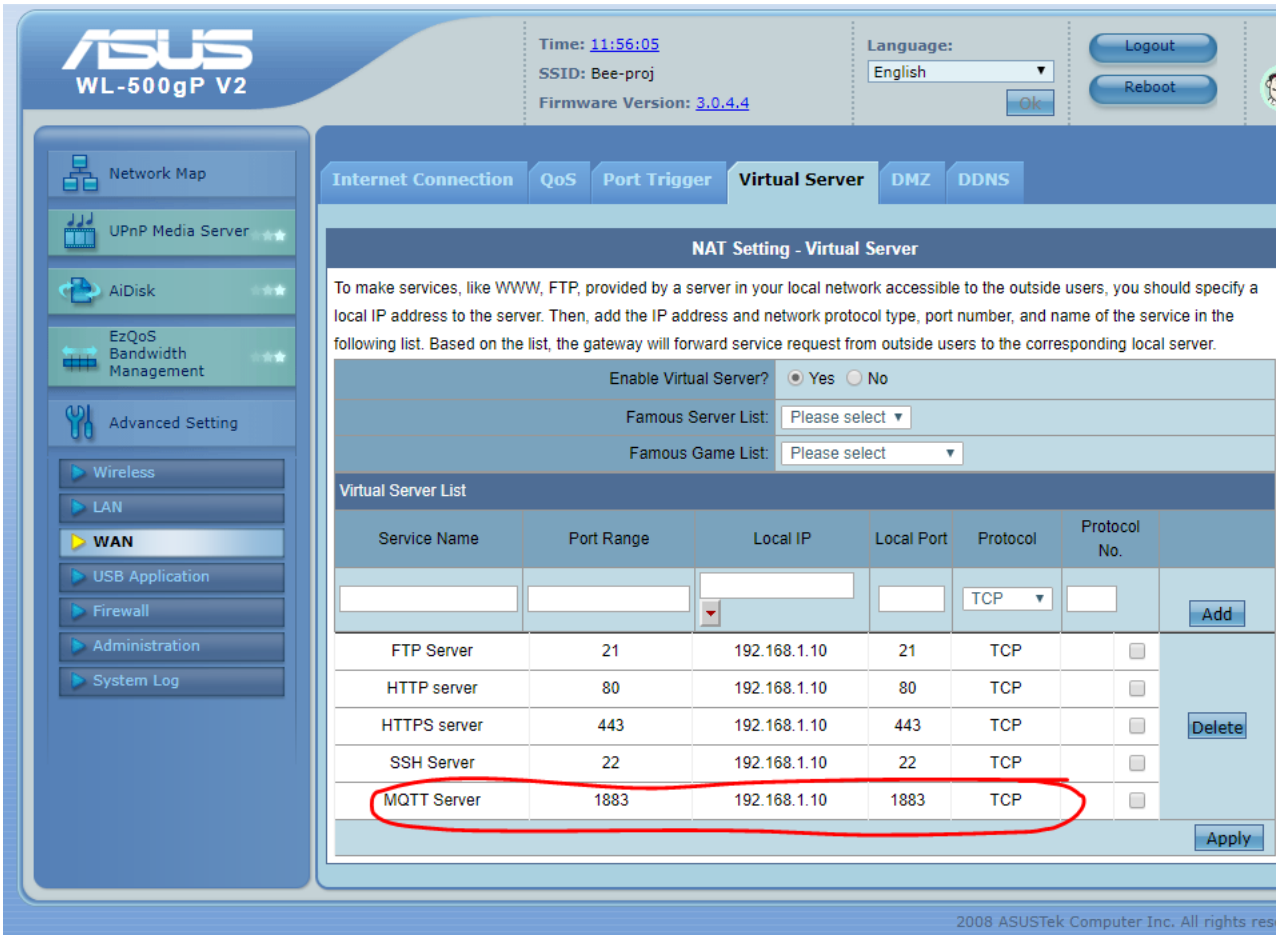

Fig. 15. Port forwarding configuration

- mqtt_password
- mqtt_broker_ip

by your own parameters. Also, you need to install the library paho-mqtt:
pip install paho-mqtt

Also, change user rights in this file:
sudo chmod 777 PythonMQTT_Subscribe.py

In arduino, you need to launch *MQTT_Publish.ino* program. Change constant:

- ssid
- wifi_password
- mqtt_server
- mqtt_username
- mqtt_password

by your own parameters.

Launch both programs :

- raspberry: python PythonMQTT_Subscribe.py
- arduino: click to upload button and wait "hard resetting via RTS pin..." message. After that, click on serial monitor button on the right. In bottom you will see dropdown with "... baud", change to "115200 baud"

Normally you will see:

2.2 MQTT communication with RFID program

In the second program, you will transfer RFID number by MQTT protocol.
In raspberry, put *MQTT communication with RFID program/PythonMQTT_Subscribe.py* on your raspberry and change the value:

- mqtt_username
- mqtt_password
- mqtt_broker_ip

by your own parameters.
Also, change user rights in this file:
sudo chmod 777 PythonMQTT_Subscribe.py

In arduino, you need to launch *MQTT_Publish.ino* program. Change constant:

- ssid
- wifi_password
- mqtt_server
- mqtt_username

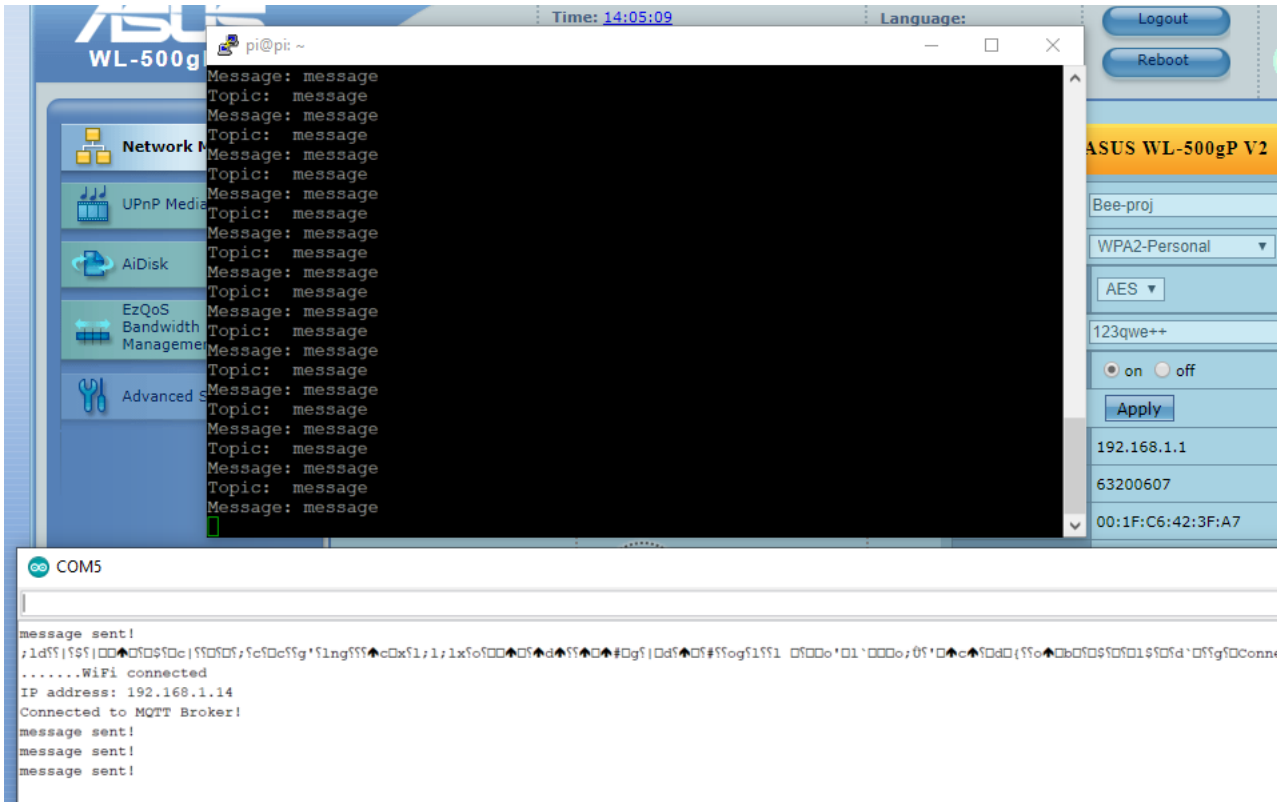

Fig. 16. First program result

– mqtt_password

by your own parameters.

Launch both programs :

– raspberry: python PythonMQTT_Subscribe.py
– arduino: click to upload button and wait "hard resetting via RTS pin..." message. After that, click on serial monitor button on the right. In bottom you will see dropdown with "... baud", change to "9600 baud"

Normally, you will see:

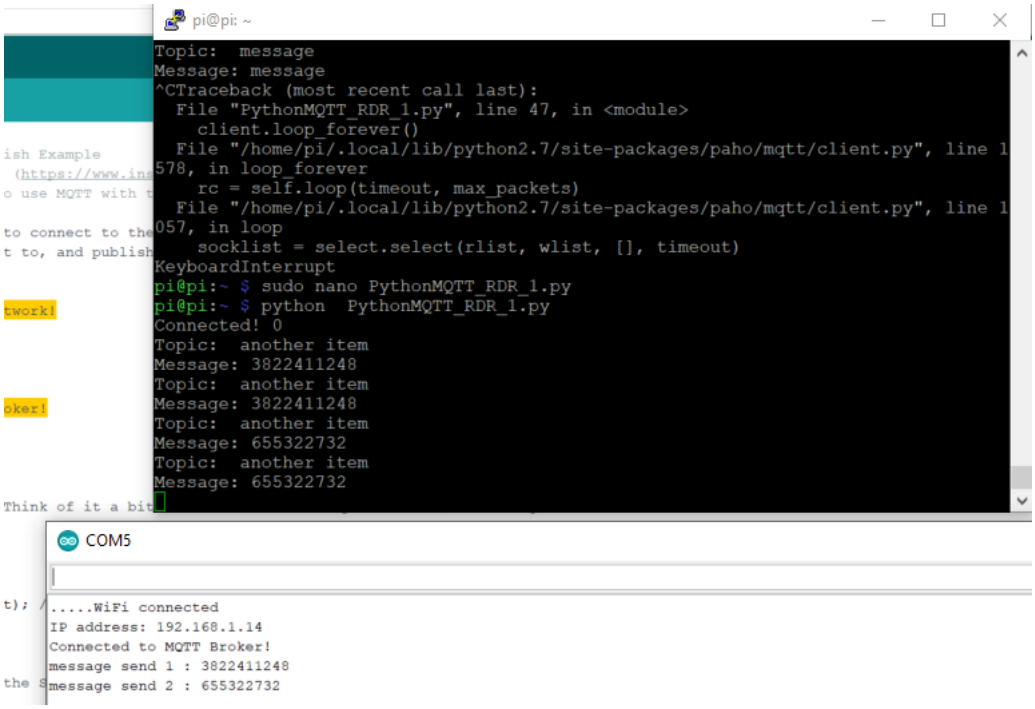

Fig. 17. Second program result