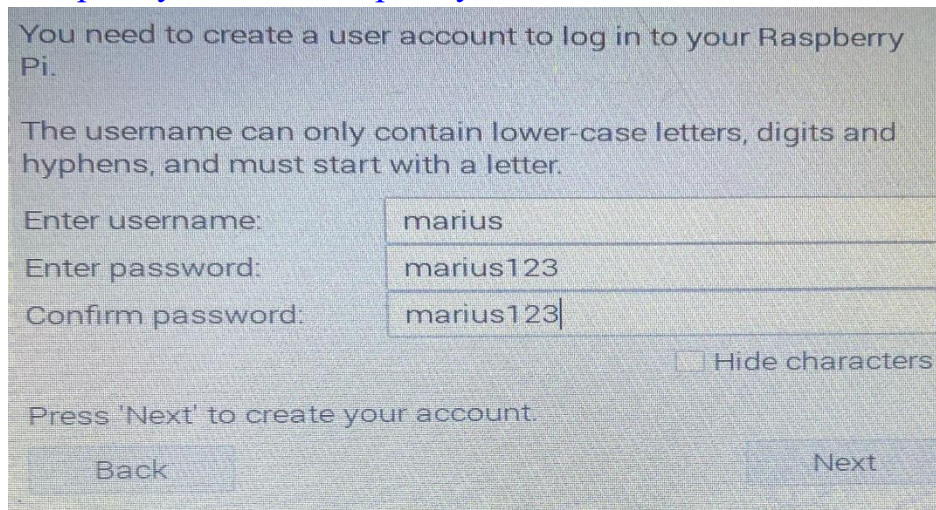


# Tutorial setup pentru aplicatie IoT

*Autor: Codreanu Dan*

## 1. Instalare mosquitto MQTT broker:

1. **Pasul 1 (optional):** – Reinstalam Raspberry PI OS versiunea **Bullseye 32bit pentru Raspberry PI 3 B** dupa instructiunile din site-ul oficial: [Raspberry Pi OS – Raspberry Pi](#)



You need to create a user account to log in to your Raspberry Pi.

The username can only contain lower-case letters, digits and hyphens, and must start with a letter.

Enter username:

Enter password:

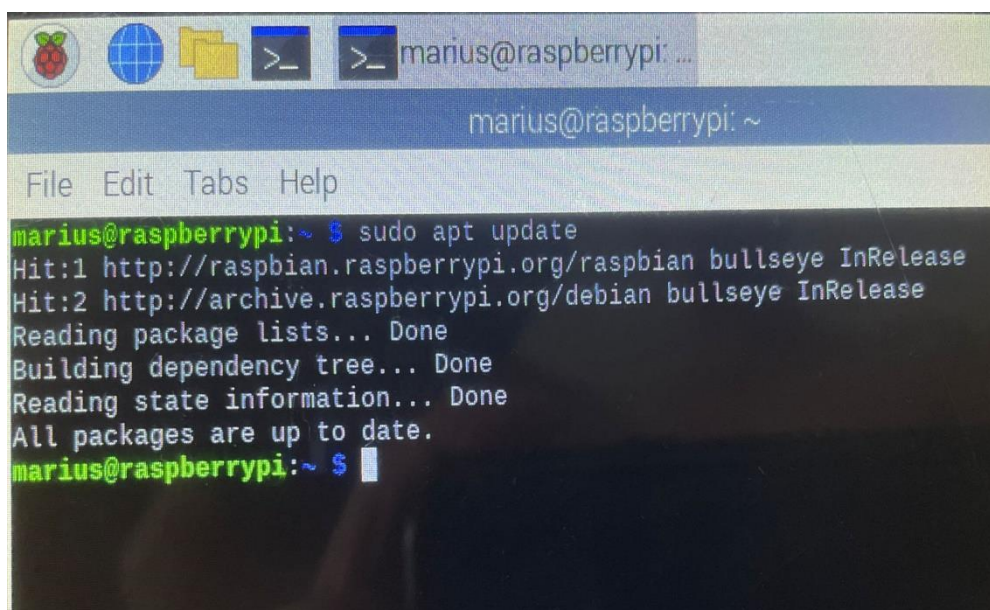
Confirm password:

☐ Hide characters

Press 'Next' to create your account.

Fig 1. Noul username si parola pentru Raspberry PI

2. **Pasul 2:** Facem un update la sistemul de operare folosind comanda in consola: **sudo apt update**

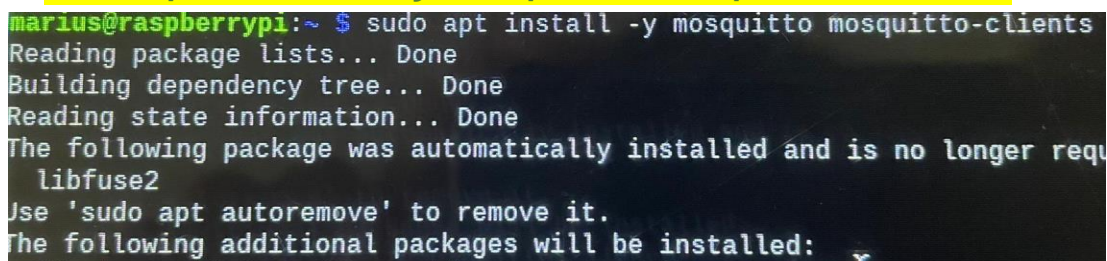


```
marius@raspberrypi: ~  
File Edit Tabs Help  
marius@raspberrypi:~ $ sudo apt update  
Hit:1 http://raspbian.raspberrypi.org/raspbian bullseye InRelease  
Hit:2 http://archive.raspberrypi.org/debian bullseye InRelease  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
All packages are up to date.  
marius@raspberrypi:~ $
```

Fig 2. Comanda executata cu success

3. **Pasul 3:** Instalăm broker-ul mosquitto MQTT folosind comanda:

**sudo apt install -y mosquitto mosquitto-clients**

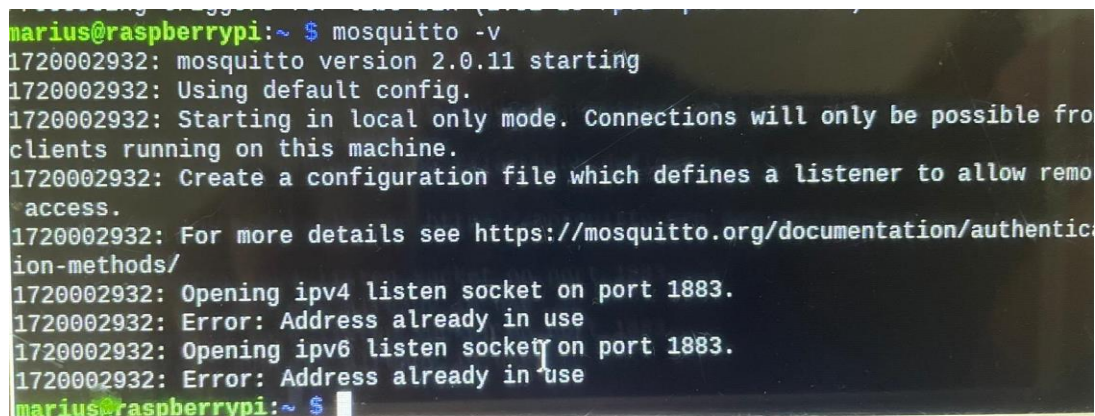


```
marius@raspberrypi:~$ sudo apt install -y mosquitto mosquitto-clients
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following package was automatically installed and is no longer required:
  libfuse2
Use 'sudo apt autoremove' to remove it.
The following additional packages will be installed:
```

Fig 3. Instalare mqtt broker mosquitto

4. **Pasul 4:** verificăm versiunea pentru mosquitto mqtt folosind comanda:

**mosquitto -v**



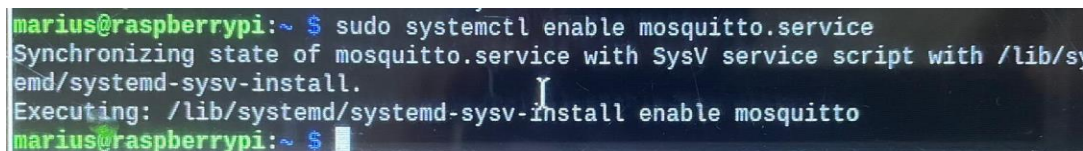
```
marius@raspberrypi:~$ mosquitto -v
1720002932: mosquitto version 2.0.11 starting
1720002932: Using default config.
1720002932: Starting in local only mode. Connections will only be possible from
clients running on this machine.
1720002932: Create a configuration file which defines a listener to allow remote
access.
1720002932: For more details see https://mosquitto.org/documentation/authentication-methods/
1720002932: Opening ipv4 listen socket on port 1883.
1720002932: Error: Address already in use
1720002932: Opening ipv6 listen socket on port 1883.
1720002932: Error: Address already in use
marius@raspberrypi:~$
```

Fig 4. Versiune verificată cu succes

!!! Observăm că serverul lucrează pe **portul 1883**, pe acesta îl folosim când dorim să ne conectăm ca să îi accesăm serviciile.

5. **Pasul 5:** Activăm serviciul mosquitto mqtt pentru a porni odată cu Raspberry Pi utilizând comanda în consolă:

**sudo systemctl enable mosquitto.service**

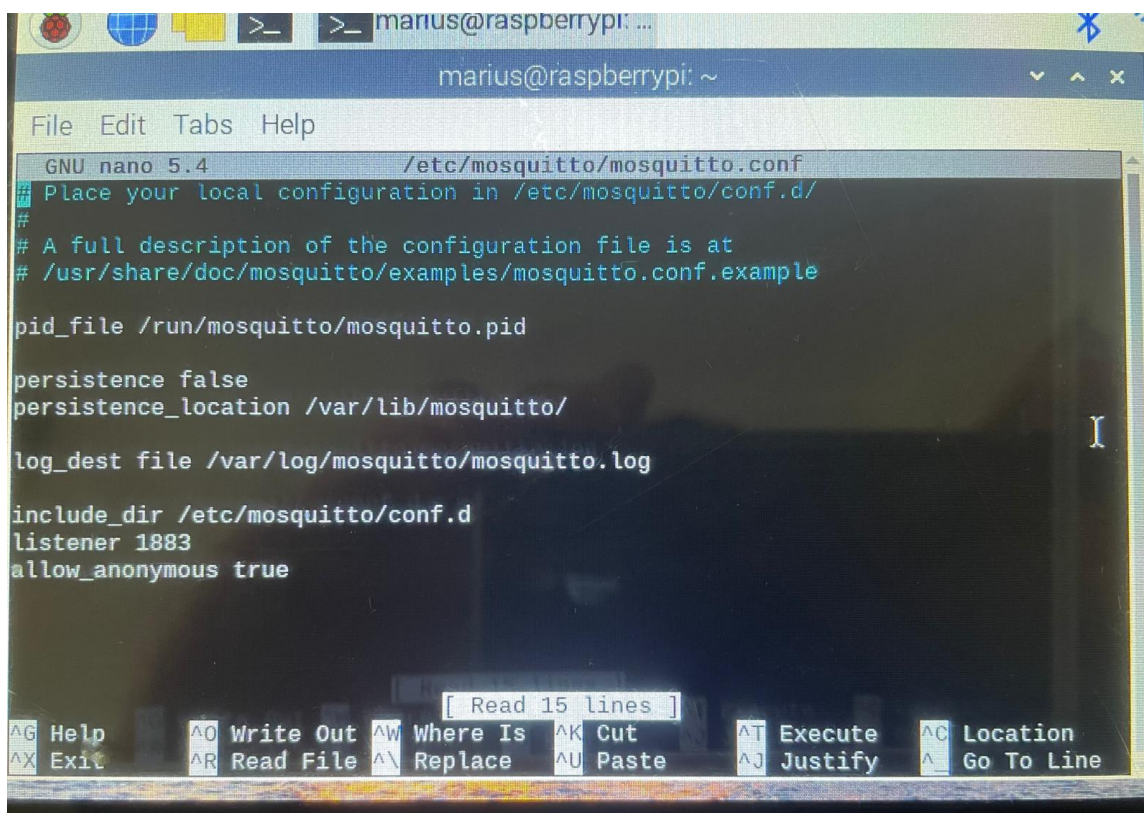


```
marius@raspberrypi:~$ sudo systemctl enable mosquitto.service
Synchronizing state of mosquitto.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable mosquitto
marius@raspberrypi:~$
```

Fig 5. Serviciu activat



6. **Pasul 6:** Accesam fisierul config pentru server folosind comanda in consola: **sudo nano /etc/mosquitto/mosquitto.conf**



```
GNU nano 5.4 /etc/mosquitto/mosquitto.conf
# Place your local configuration in /etc/mosquitto/conf.d/
#
# A full description of the configuration file is at
# /usr/share/doc/mosquitto/examples/mosquitto.conf.example

pid_file /run/mosquitto/mosquitto.pid

persistence false
persistence_location /var/lib/mosquitto/

log_dest file /var/log/mosquitto/mosquitto.log

include_dir /etc/mosquitto/conf.d
listener 1883
allow_anonymous true
```

**Fig 6.** Fisierul mosquitto.conf cu noile valori

Aici am modificat urmatoarele:

- Am sters: **per\_listener\_settings** si **password\_file**.
- Am modificat **persistence true** in **persistence false** pentru a elimina stocarea in baza de date interna si a nu a supraincarca serverul, evitam crash-ul.
- Am configurat conectare „anonymous” fara a utiliza credentiale adica am adaugat **listener 1883** si **allow\_anonymous true**.

In final apasam **Ctrl+O** pentru a scrie modificarile, apasam **Enter** si dupa **Ctrl+X** pentru a inchide fisierul dupa care OS-ul ne intoarce in consola. (Atentie setarile sa fie facute cu litera mica, e case sensitive !!!)

7. **Pasul 7:** Verificam status-ul mosquitto server folosind comanda: **systemctl status mosquitto.service**

```
marius@raspberrypi:~$ systemctl status mosquitto.service
● mosquitto.service - Mosquitto MQTT Broker
   Loaded: loaded (/lib/systemd/system/mosquitto.service; enabled; vendor pre
   Active: active (running) since Wed 2024-07-03 13:08:56 EEST; 2h 6min ago
     Docs: man:mosquitto.conf(5)
           man:mosquitto(8)
   Main PID: 4220 (mosquitto)
      Tasks: 1 (limit: 1595)
         CPU: 2.161s
    CGroup: /system.slice/mosquitto.service
            └─4220 /usr/sbin/mosquitto -c /etc/mosquitto/mosquitto.conf

Jul 03 13:08:56 raspberrypi systemd[1]: Starting Mosquitto MQTT Broker...
Jul 03 13:08:56 raspberrypi systemd[1]: Started Mosquitto MQTT Broker.
lines 1-13/13 (END)
```

Fig 7. Status server: server-ul functioneaza

8. **Pasul 8:** Pentru a afla IP-ul pentru a ne putea conecta la server cu ajutorul ESP32 folosim comanda: **hostname -I**
9. **Pasul 9:** Testam conexiunea la server ca in figura 9:

### Testing Mosquitto Broker and Client on Raspberry Pi

Pornim mosquitto pentru a rula in fundal (daemon): **mosquitto -d**

```
dan@raspberrypi:~$ mosquitto -d
dan@raspberrypi:~$
```

Cream un topic: **mosquitto sub -d -t testTopic**

```
dan@raspberrypi:~$ mosquitto sub -d -t testTopic
Client (null) sending CONNECT
Client (null) received CONNACK (0)
Client (null) sending SUBSCRIBE (Mid: 1, Topic: testTopic, QoS: 0, Options: 0x00)
Client (null) received SUBACK
Subscribed (mid: 1): 0
```

Subscribe la topic si trimtem mesaj "Test !" din alt terminal

**mosquitto pub -d -t testTopic -m "Test !"**

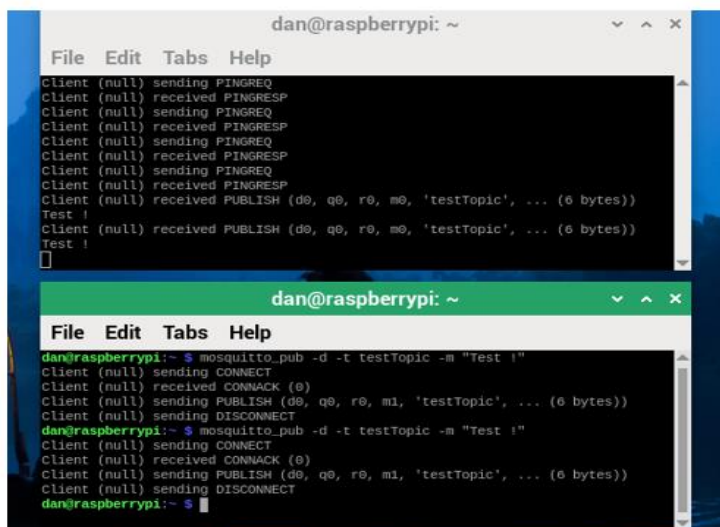


Fig 9. Testarea conexiunii pentru server-ul MQTT

!!! Comenzi utile:

`sudo systemctl restart mosquitto` – pentru restart

`sudo journalctl -u mosquitto.service` – pentru crash log

`sudo systemctl is-enabled mosquitto` – pentru a verifica daca mosquitto porneste odata cu RaspPi

## 2. Instalare NodeRed:

1. **Pasul 1:** Instalam NodeRed folosind utilizand comanda in consola:

`bash<(curl -sL https://raw.githubusercontent.com/node-red/linux-installers/master/deb/update-nodejs-and-nodered)`

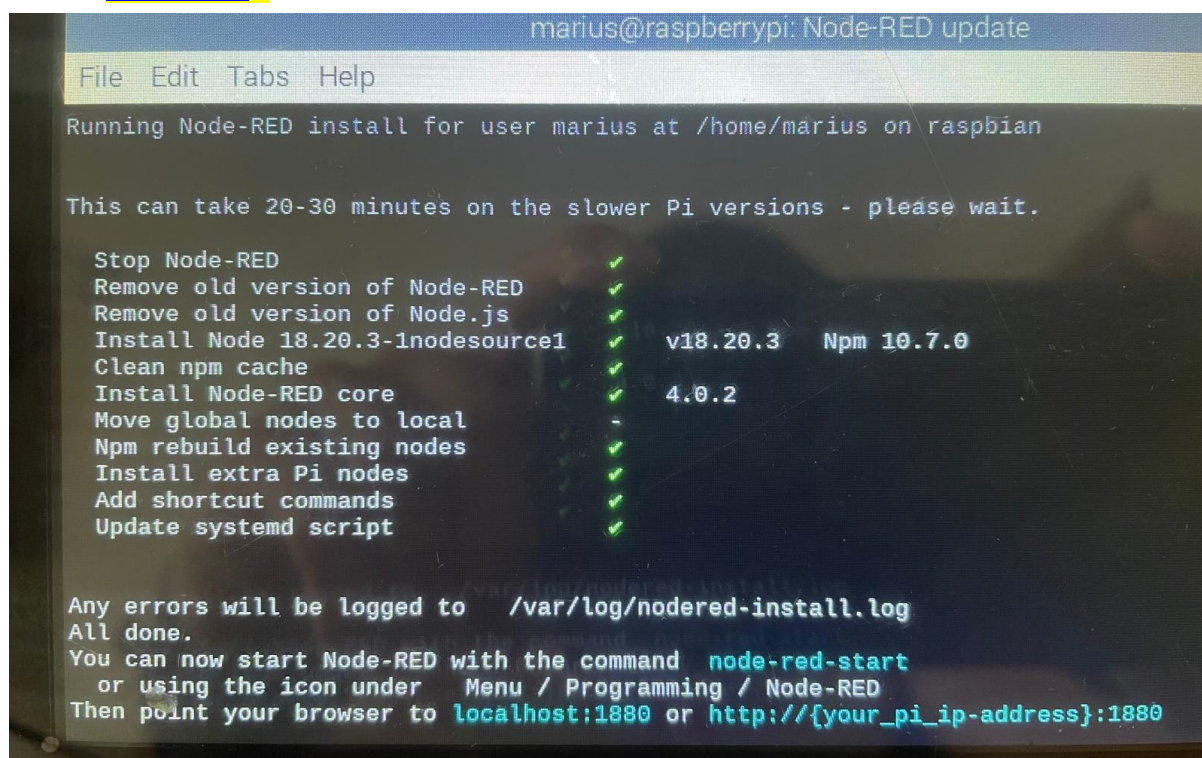
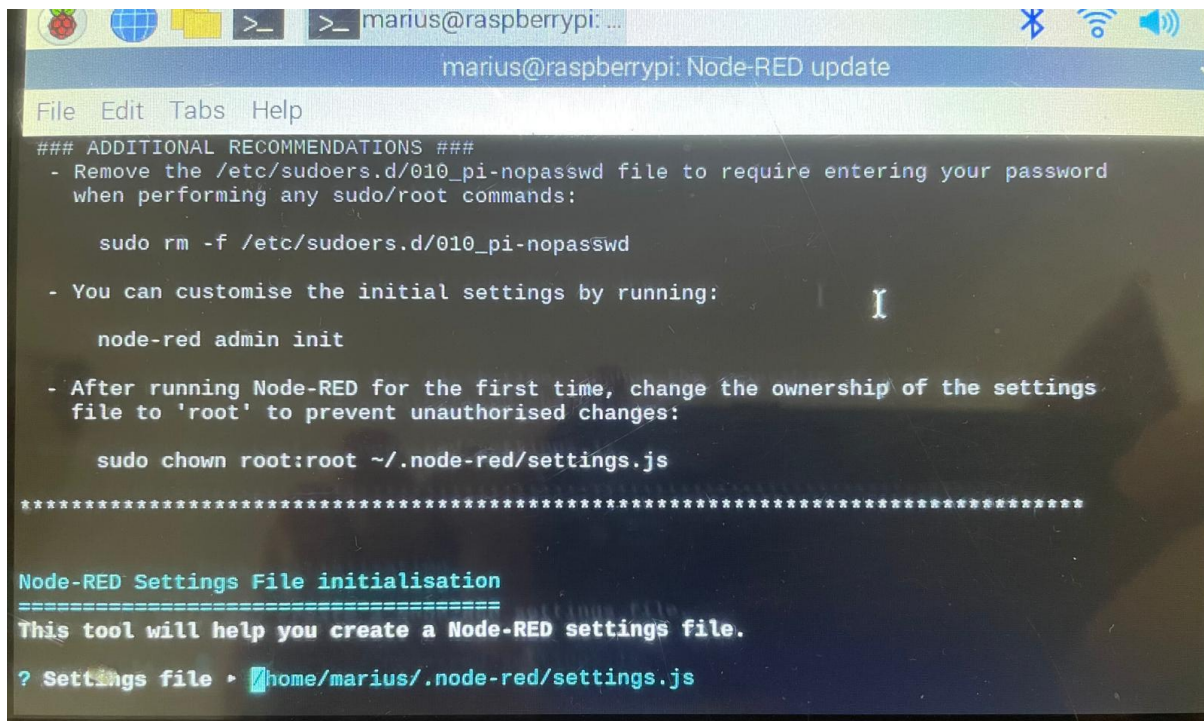


Fig 10. Instalare NodeRed cu succes

!!! Dupa cum scrie si in imagine pentru a accesa NodeRed folosim adresa de ca mai sus si aflam IP-ul folosind `hostname -I`





The screenshot shows a terminal window titled 'marius@raspberrypi: Node-RED update'. The window contains the following text:

```
File Edit Tabs Help
### ADDITIONAL RECOMMENDATIONS ###
- Remove the /etc/sudoers.d/010_pi-nopasswd file to require entering your password
  when performing any sudo/root commands:

  sudo rm -f /etc/sudoers.d/010_pi-nopasswd

- You can customise the initial settings by running:

  node-red admin init

- After running Node-RED for the first time, change the ownership of the settings
  file to 'root' to prevent unauthorised changes:

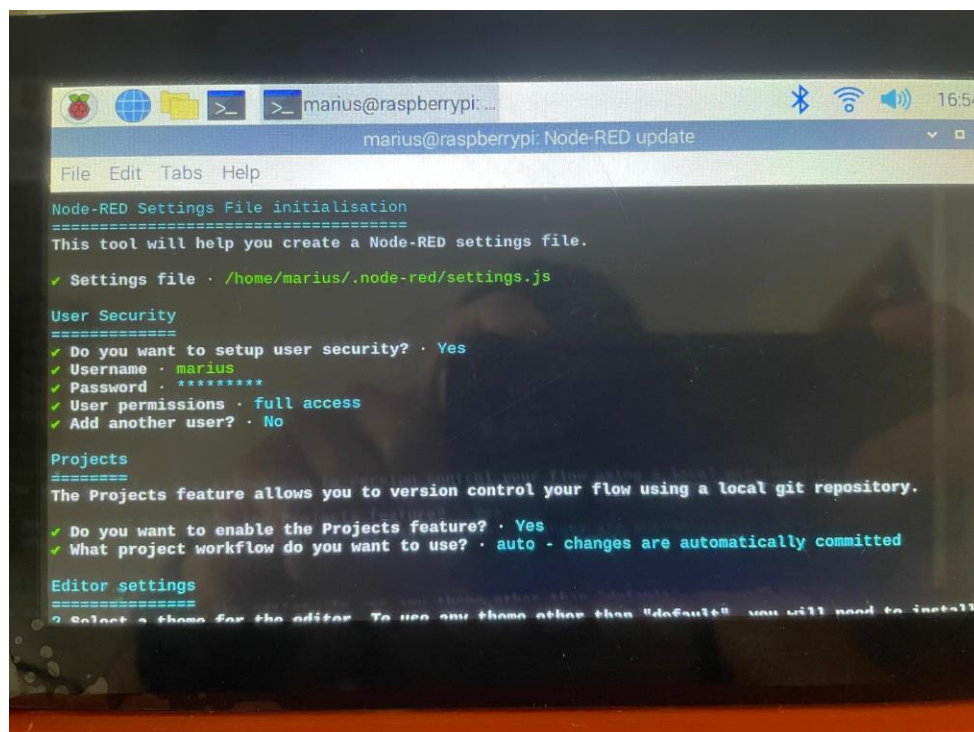
  sudo chown root:root ~/.node-red/settings.js

*****

Node-RED Settings File initialisation
=====
This tool will help you create a Node-RED settings file.
? Settings file · /home/marius/.node-red/settings.js
```

Fig 11. Setari aditionale pentru NodeRed

2. **Pasul 2:** Apasam enter dupa aceeaa secventa pentru a configura Node Red sau introducem `node-red admin init` in consola.



The screenshot shows a terminal window titled 'marius@raspberrypi: Node-RED update'. The window contains the following text:

```
File Edit Tabs Help
Node-RED Settings File initialisation
=====
This tool will help you create a Node-RED settings file.
✓ Settings file · /home/marius/.node-red/settings.js

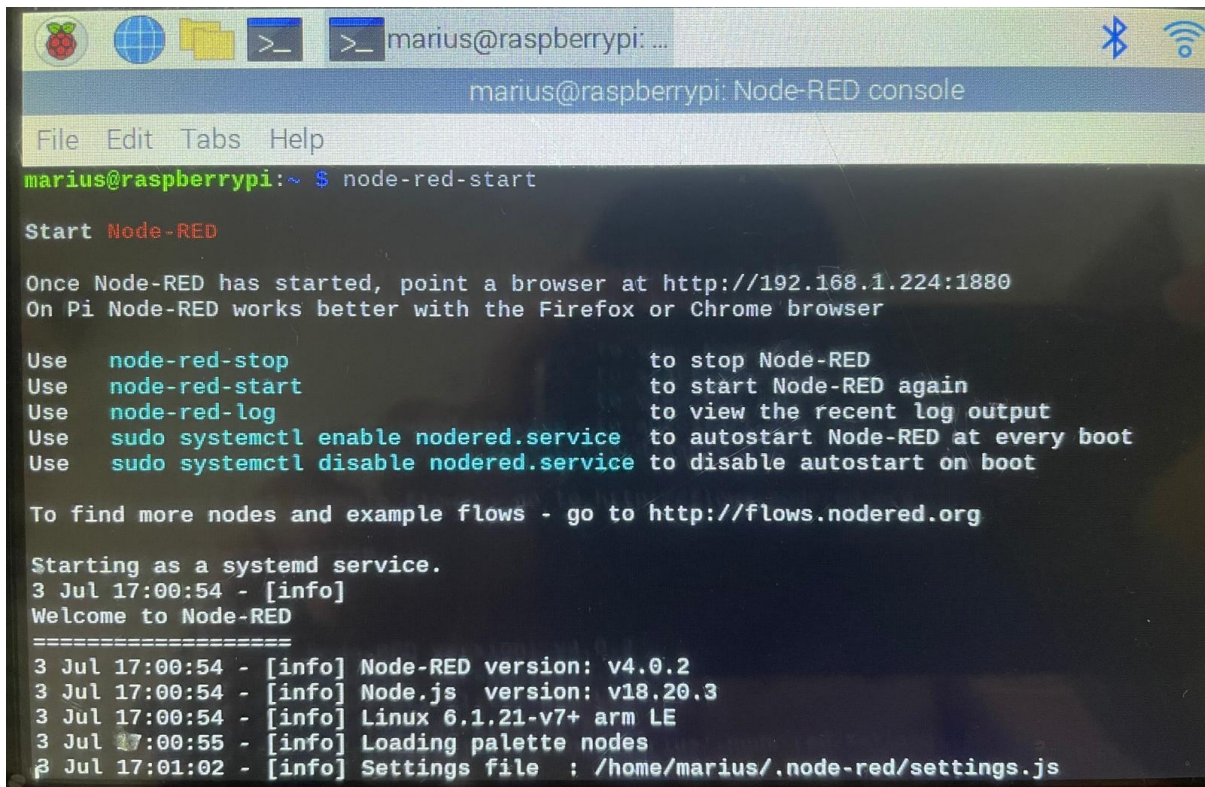
User Security
=====
✓ Do you want to setup user security? · Yes
✓ Username · marius
✓ Password · *****
✓ User permissions · full access
✓ Add another user? · No

Projects
=====
The Projects feature allows you to version control your flow using a local git repository.
✓ Do you want to enable the Projects feature? · Yes
✓ What project workflow do you want to use? · auto - changes are automatically committed

Editor settings
=====
? Select a theme for the editor. To use any theme other than "default" you will need to install
```

Fig 12. Configurare NodeRed

### 3. Pasul 3: Pornim Node Red folosind comanda **node-red-start**



```
marius@raspberrypi:~ $ node-red-start

Start Node-RED

Once Node-RED has started, point a browser at http://192.168.1.224:1880
On Pi Node-RED works better with the Firefox or Chrome browser

Use node-red-stop to stop Node-RED
Use node-red-start to start Node-RED again
Use node-red-log to view the recent log output
Use sudo systemctl enable nodered.service to autostart Node-RED at every boot
Use sudo systemctl disable nodered.service to disable autostart on boot

To find more nodes and example flows - go to http://flows.nodered.org

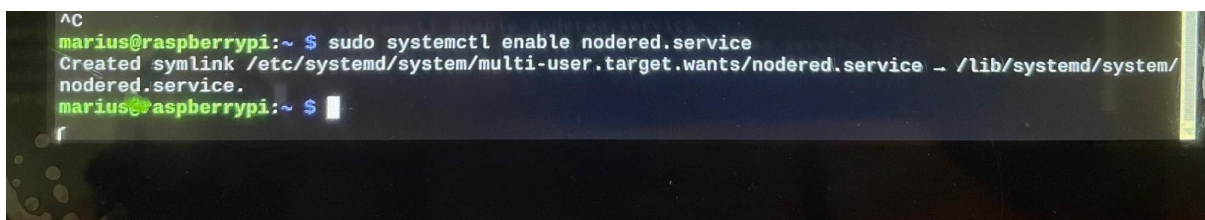
Starting as a systemd service.
3 Jul 17:00:54 - [info]
Welcome to Node-RED
=====
3 Jul 17:00:54 - [info] Node-RED version: v4.0.2
3 Jul 17:00:54 - [info] Node.js version: v18.20.3
3 Jul 17:00:54 - [info] Linux 6.1.21-v7+ arm LE
3 Jul 17:00:55 - [info] Loading palette nodes
3 Jul 17:01:02 - [info] Settings file : /home/marius/.node-red/settings.js
```

Fig 13. Pornire Node Red cu succes

!!! Folosim comenzile din figura pentru a configura NodeRed in continuare.

Conectarea la NodeRed se face folosind adresa indicata si la Dashboard ne conectam atasand la final dupa 1880 un **/ui**.

### 4. Pasul 4: Activam pornirea NodeRed sa fie concomitenta cu cea a Raspberry Pi utilizand comanda: **sudo systemctl enable nodered.service**

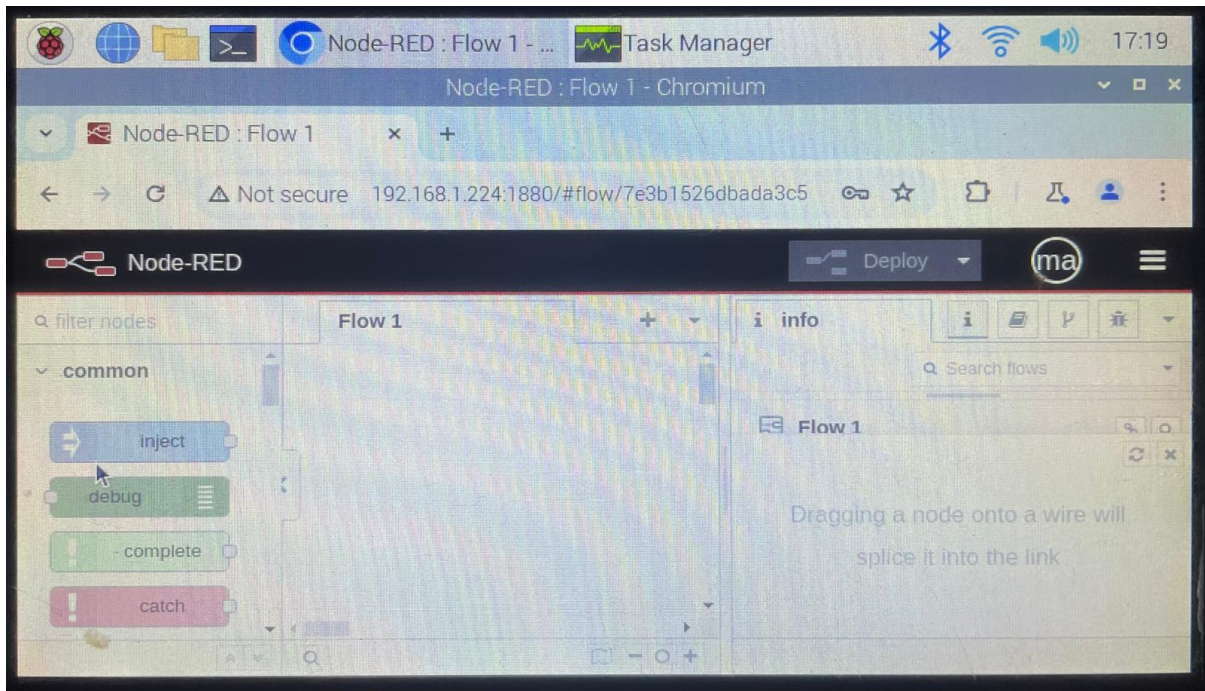


```
marius@raspberrypi:~ $ sudo systemctl enable nodered.service
Created symlink /etc/systemd/system/multi-user.target.wants/nodered.service - /lib/systemd/system/nodered.service.
marius@raspberrypi:~ $
```

Fig 14. Activare NodeRed la pornire RaspPI

!!! La final aplicam setarile cu un **sudo reboot**.





**Fig 11.** Node Red in browser intern RaspPi

Testare Node Red in browser RaspPi.



### 3. Conectare ESP32 la server MQTT si vizualizare in NodeRed

**1.Pasul 1:** Cod exemplu pentru **ESP32**. Ne conectam la serverul **MQTT** folosind adresa IP, folosind **adresa IP gasita** cu comanda **hostname -I** in consola **Raspberry PI**.

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

// Replace the SSID/Password details as per your wifi router
const char* ssid = "empty";
const char* password = "empty";

// Replace your MQTT Broker IP address here:
const char* mqtt_server = "192.168.0.80"; /* hostname -I pe RPI */

WiFiClient espClient;
PubSubClient client(espClient);

#define ledPin 2 //built-in LED of ESP32

long lastMsg = 0;

void setup()
{
  pinMode(ledPin, OUTPUT);
  Serial.begin(115200);

  setup_wifi();
  client.setServer(mqtt_server, 1883); // 1883 is the default port
  for MQTT server
  client.setCallback(callback);
}

void loop()
{
  if (!client.connected())
  {
    connect_mqttServer();
  }

  client.loop();
}
```

```

    long now = millis();
    if (now - lastMsg > 4000)
    {
        lastMsg = now;
        client.publish("esp32/sensor1", "88"); // topic name (to which
this ESP32 publishes its data). 88 is the dummy value.
        Serial.print("[MQTT_TRANSMIT] Sent value to the server:
");Serial.println(" 88");
    }
}

void connect_mqttServer()
{
    // Loop until we're reconnected
    while (!client.connected())
    {
        // First check if connected to wifi
        if(WiFi.status() != WL_CONNECTED)
        {
            // If not connected, then first connect to wifi
            setup_wifi();
        }

        // Now attempt to connect to MQTT server
        Serial.print("[MQTT_CONNECTION] Attempting MQTT connection...");
        // Attempt to connect
        if (client.connect("ESP32_client1"))
        {
            // Attempt successful
            Serial.println("connected");
            // Subscribe to topics here
            client.subscribe("rpi/broadcast");
            // client.subscribe("rpi/xyz"); //subscribe more topics here
        }
        else
        {
            // Attempt not successful
            Serial.print("failed, rc=");
            Serial.print(client.state());
            Serial.println(" trying again in 2 seconds");
            blink_led(1, 1250); // Blink LED once (for 1250ms ON time)
        }
    }
}

```



```

        // Wait 2 seconds before retrying
        delay(2000);
    }
}

void callback(char* topic, byte* message, unsigned int length)
{
    Serial.print("[MQTT_RECEIVE] Message arrived on topic: ");
    Serial.print(topic);
    Serial.print(". Message: ");
    String messageTemp;

    for (int i = 0; i < length; i++)
    {
        Serial.print((char)message[i]);
        messageTemp += (char)message[i];
    }
    Serial.println();

    // Check if a message is received on the topic "rpi/broadcast"
    if (String(topic) == "rpi/broadcast")
    {
        if(messageTemp == "10")
        {
            Serial.println("Action: blink LED");
            blink_led(1, 1250); // Blink LED once (for 1250ms ON time)
        }
    }

    // Similarly add more if statements to check for other subscribed
    // topics
}

void setup_wifi()
{
    delay(10);
    // We start by connecting to a WiFi network
    Serial.println();
    Serial.print("[WIFI_CONNECTION] Connecting to ");
    Serial.println(ssid);

```

```
WiFi.begin(ssid, password);

while (WiFi.status() != WL_CONNECTED)
{
    delay(500);
    Serial.print(".");
}

Serial.println("");
Serial.print("[WIFI_CONNECTION]  WiFi connected:");
Serial.print("IP address: ");
Serial.println(WiFi.localIP());
}

void blink_led(int times, int duration)
{
    for (int i = 0; i < times; i++)
    {
        digitalWrite(ledPin, HIGH); // Turn the LED on
        delay(duration);
        digitalWrite(ledPin, LOW);  // Turn the LED off
        delay(duration);
    }
}
```

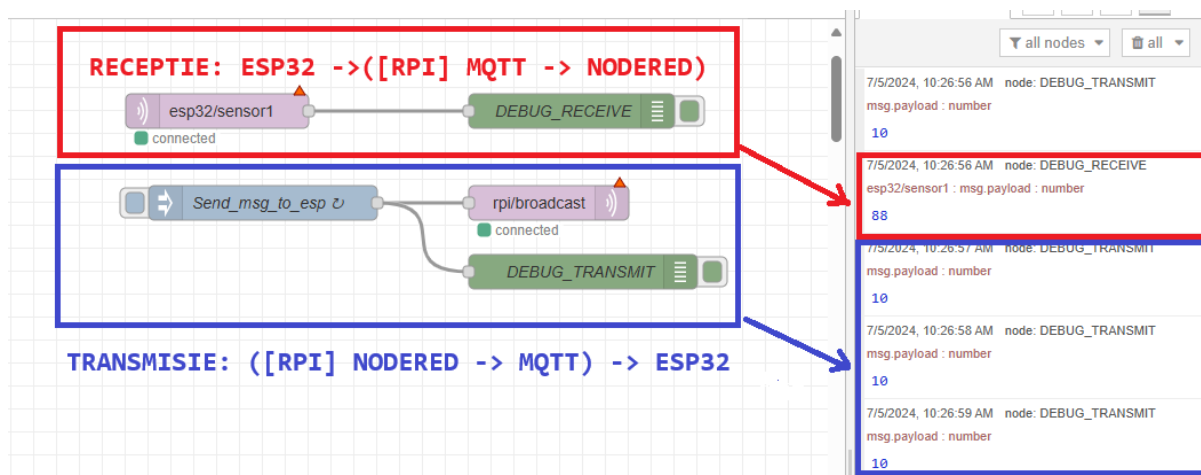
In continuare vom observa cum functioneaza codul pe esp vizualizand consola:



```
10:10:56.223 -> ets Jul 29 2019 12:21:46
10:10:56.261 ->
10:10:56.261 -> rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
10:10:56.261 -> configisp: 0, SPIWP:0xee
10:10:56.261 -> clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
10:10:56.261 -> mode:DIO, clock div:1
10:10:56.261 -> load:0x3ffff000,len:1344
10:10:56.261 -> load:0x40078000,len:13964
10:10:56.261 -> load:0x40080400,len:3600
10:10:56.261 -> entry 0x400805f0
10:10:56.560 ->
10:10:56.560 -> [WIFI_CONNECTION] Connecting to CIDDC
10:10:57.182 -> ..
10:10:57.664 -> [WIFI_CONNECTION] WiFi connected:IP address: 192.168.0.25
10:10:57.664 -> [MQTT_CONNECTION] Attempting MQTT connection...connected
10:10:58.570 -> [MQTT_RECEIVE] Message arrived on topic: rpi/broadcast. Message: 10
10:10:58.613 -> Action: blink LED
10:11:01.080 -> [MQTT_TRANSMIT] Sent value to the server: 88
10:11:01.080 -> [MQTT_RECEIVE] Message arrived on topic: rpi/broadcast. Message: 10
10:11:01.126 -> Action: blink LED
10:11:03.604 -> [MQTT_RECEIVE] Message arrived on topic: rpi/broadcast. Message: 10
10:11:03.604 -> Action: blink LED
10:11:06.117 -> [MQTT_TRANSMIT] Sent value to the server: 88
10:11:06.117 -> [MQTT_RECEIVE] Message arrived on topic: rpi/broadcast. Message: 10
10:11:06.117 -> Action: blink LED
10:11:08.600 -> [MQTT_RECEIVE] Message arrived on topic: rpi/broadcast. Message: 10
10:11:08.600 -> Action: blink LED
10:11:11.083 -> [MQTT_TRANSMIT] Sent value to the server: 88
10:11:11.083 -> [MQTT_RECEIVE] Message arrived on topic: rpi/broadcast. Message: 10
10:11:11.129 -> Action: blink LED
10:11:13.578 -> [MQTT_RECEIVE] Message arrived on topic: rpi/broadcast. Message: 10
10:11:13.623 -> Action: blink LED
10:11:16.080 -> [MQTT_TRANSMIT] Sent value to the server: 88
10:11:16.122 -> [MQTT_RECEIVE] Message arrived on topic: rpi/broadcast. Message: 10
10:11:16.122 -> Action: blink LED
10:11:18.606 -> [MQTT_RECEIVE] Message arrived on topic: rpi/broadcast. Message: 10
10:11:18.606 -> Action: blink LED
10:11:21.129 -> [MQTT_TRANSMIT] Sent value to the server: 88
10:11:21.129 -> [MQTT_RECEIVE] Message arrived on topic: rpi/broadcast. Message: 10
10:11:21.129 -> Action: blink LED
10:11:23.584 -> [MQTT_RECEIVE] Message arrived on topic: rpi/broadcast. Message: 10
10:11:23.625 -> Action: blink LED
```

Fig 12. Functionare comunicare ESP32 – server MQTT

!!! Instalam biblioteca PubSubClient.h de aici: [PubSubClient - Arduino Reference](#).



**Fig 13.** Vizualizare in NodeRed debug

!!! In NodeRed am configurat doua noduri MQTT transmit/receive am legat un debug window pentru a putea vizualiza informatia si un inject node pentru a trimite valori catre ESP32. Am accesat NodeRed intrand din browser de pe laptop folosind `adresaIPrpi:1880`.



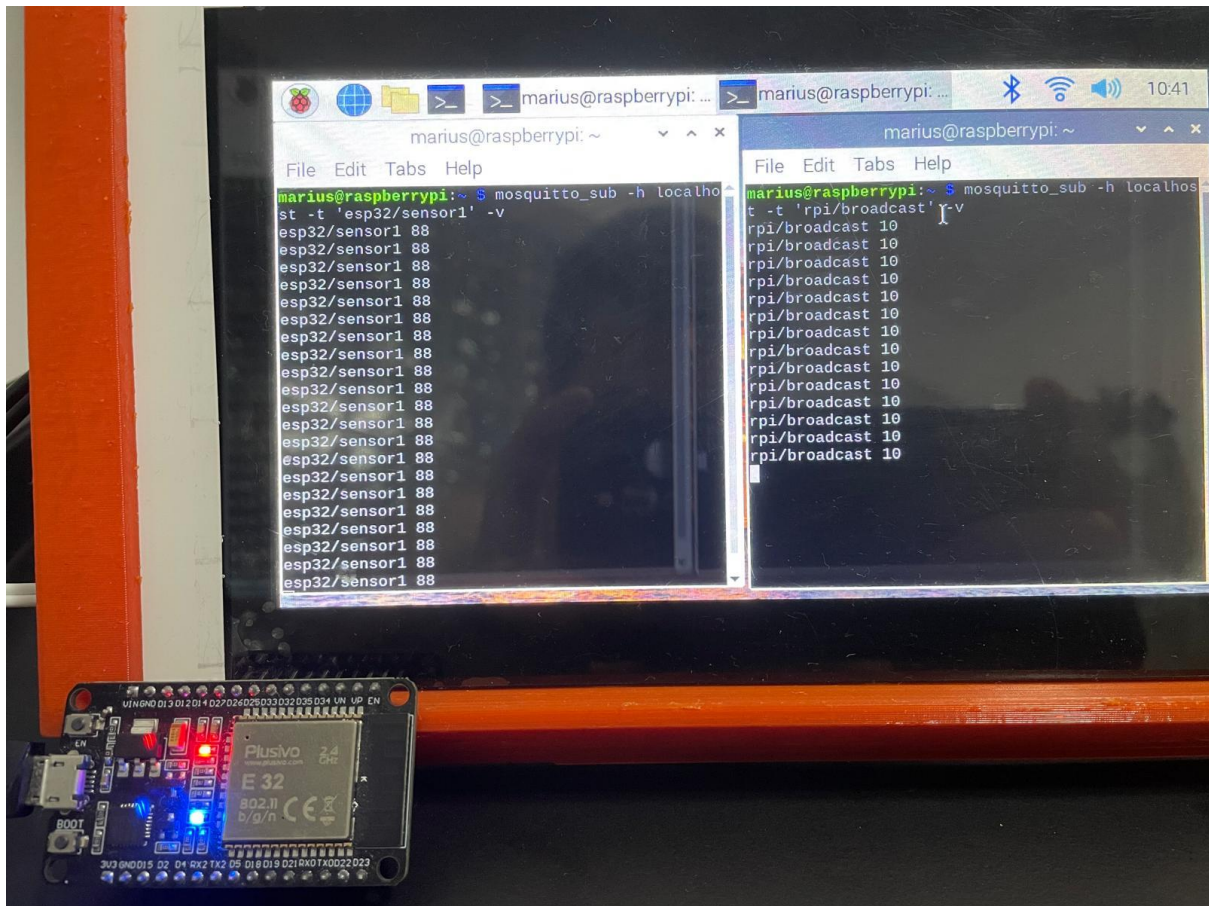


Fig 14. Functionare dispozitive

Folosind comanda in consola:

`mosquitto_sub -h localhost -t 'topic' -v`, am deschis doua terminale si putem observa transmitia si receptia de date intre ESP32 si RaspberryPI si de asemenea putem observa ca LED-ul built in al ESP32 se activeaza daca primeste mesaj cu valoarea "10" de la RaspPi.

Daca vrem sa intervenim si sa publicam ceva in topic putem folosi `mosquitto_pub -h localhost -t 'topic' -m 'value'`