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 official: Raspberry Pi OS – Raspberry Pi

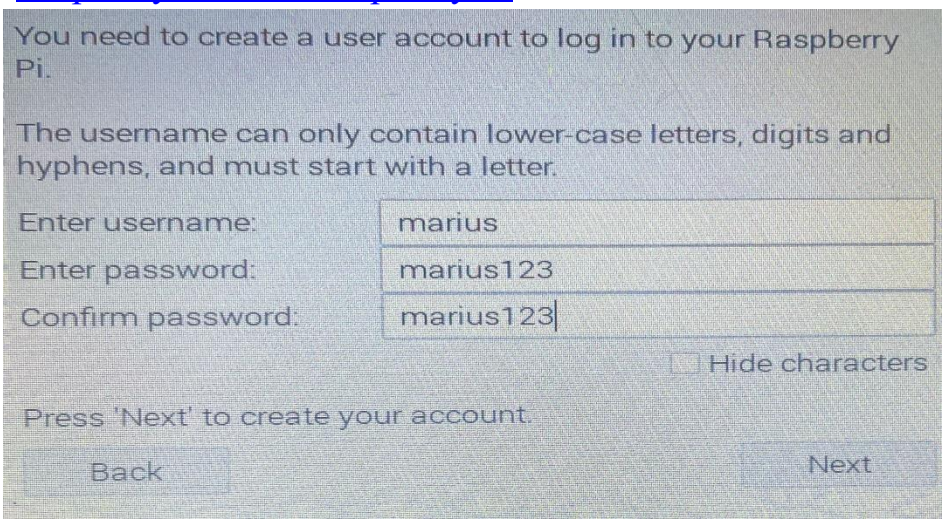

Fig 1. Noul username si parola pentru Raspberry PI

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

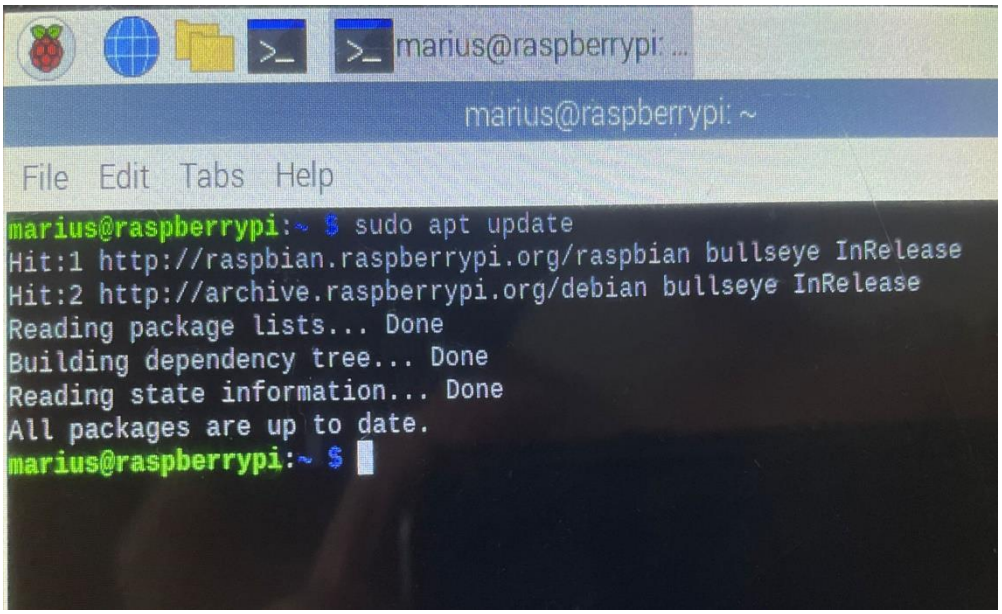

Fig 2. Comanda executata cu success

3. **Pasul 3:** Instalam broke-ul mosquito 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 requ
  libfuse2
Use 'sudo apt autoremove' to remove it.
The following additional packages will be installed:
```

Fig 3. Instalare mqtt broker mosquitto

4. **Pasul 4:** verifcam 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 fro
clients running on this machine.
1720002932: Create a configuration file which defines a listener to allow remo
 access.
1720002932: For more details see https://mosquitto.org/documentation/authentic
ion-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 verifcata cu succes

!!! Observam ca serverul lucreaza pe **portul 1883**, pe acesta il folosim cand dorim sa ne conectam ca sa ii accesam serviciile.

5. **Pasul 5:** Activam serviciul mosquito mqtt pentru a porni odata cu Raspberry Pi utilizand comanda in consola: `sudo systemctl enable mosquitto.service`

```
marius@raspberrypi:~ $ sudo systemctl enable mosquitto.service
Synchronizing state of mosquitto.service with SysV service script with /lib/sy
emd/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`

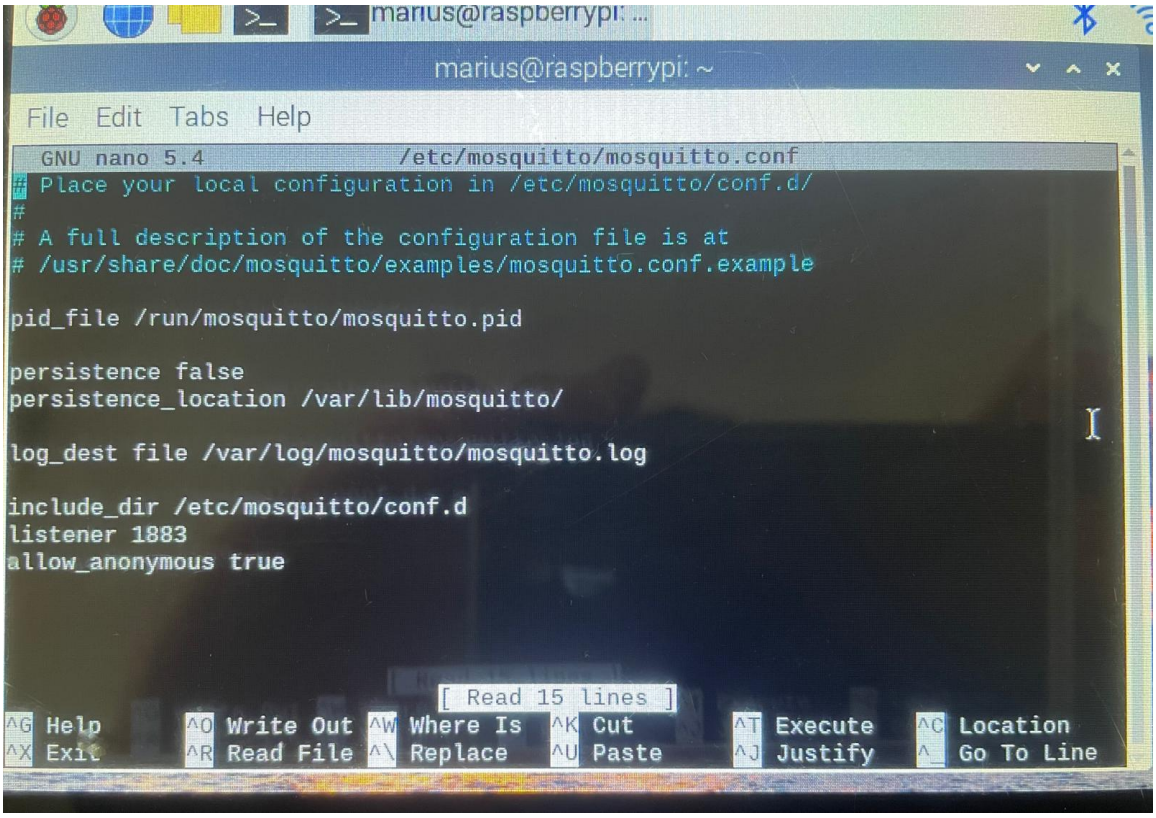

Fig 6. Fisierul mosquitto.conf cu noile valori

Aici am modificat urmatoarele:

a. Am sters: `per_listener_settings` si `password_file`.

b. Am modificat `persitence true` in `persistence false` pentru a elimina stocarea in baza de date interna si a nu a supraincarca serverul, evitam crash-ul.

c. 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:** Verficam 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:~ $
```

Creeam 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 !"`

```
dan@raspberrypi: ~
File  Edit  Tabs  Help
Client (null) sending PINGREQ
Client (null) received PINGRESP
Client (null) sending PINGREQ
Client (null) received PINGRESP
Client (null) sending PINGREQ
Client (null) received PINGRESP
Client (null) sending PINGREQ
Client (null) received PINGRESP
Client (null) received PUBLISH (d0, q0, r0, m0, 'testTopic', ... (6 bytes))
Test !
Client (null) received PUBLISH (d0, q0, r0, m0, 'testTopic', ... (6 bytes))
Test !
```

```
dan@raspberrypi: ~
File  Edit  Tabs  Help
dan@raspberrypi:~ $ mosquitto_pub -d -t testTopic -m "Test !"
Client (null) sending CONNECT
Client (null) received CONNACK (0)
Client (null) sending PUBLISH (d0, q0, r0, m1, 'testTopic', ... (6 bytes))
Client (null) sending DISCONNECT
dan@raspberrypi:~ $ mosquitto_pub -d -t testTopic -m "Test !"
Client (null) sending CONNECT
Client (null) received CONNACK (0)
Client (null) sending PUBLISH (d0, q0, r0, m1, 'testTopic', ... (6 bytes))
Client (null) sending DISCONNECT
dan@raspberrypi:~ $
```

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)`

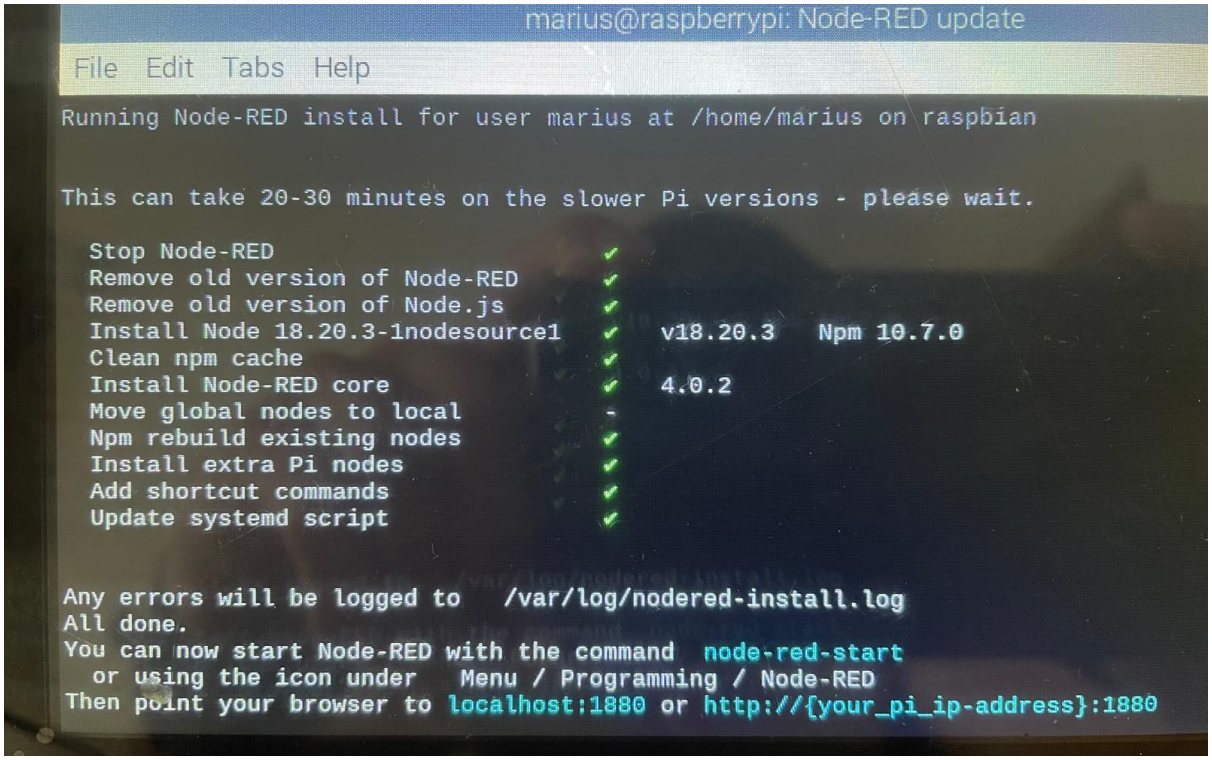

Fig 10. Instalare NodeRed cu succes

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

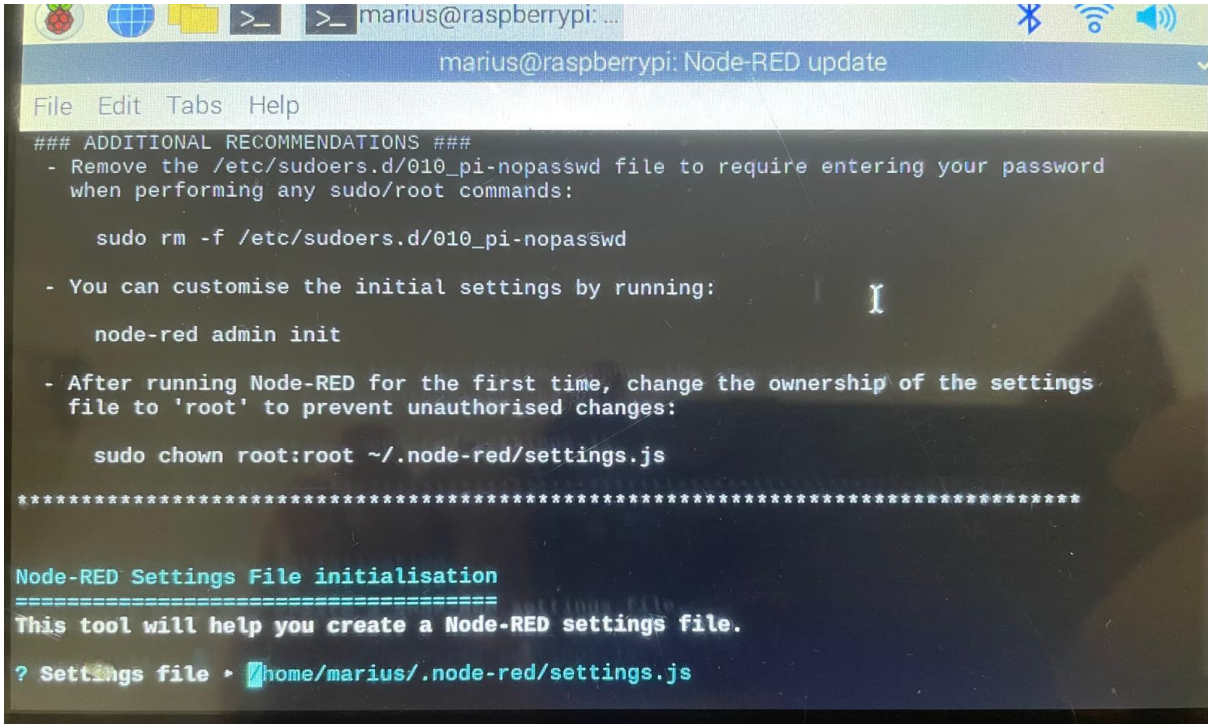

Fig 11. Setari aditionale pentru NodeRed

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

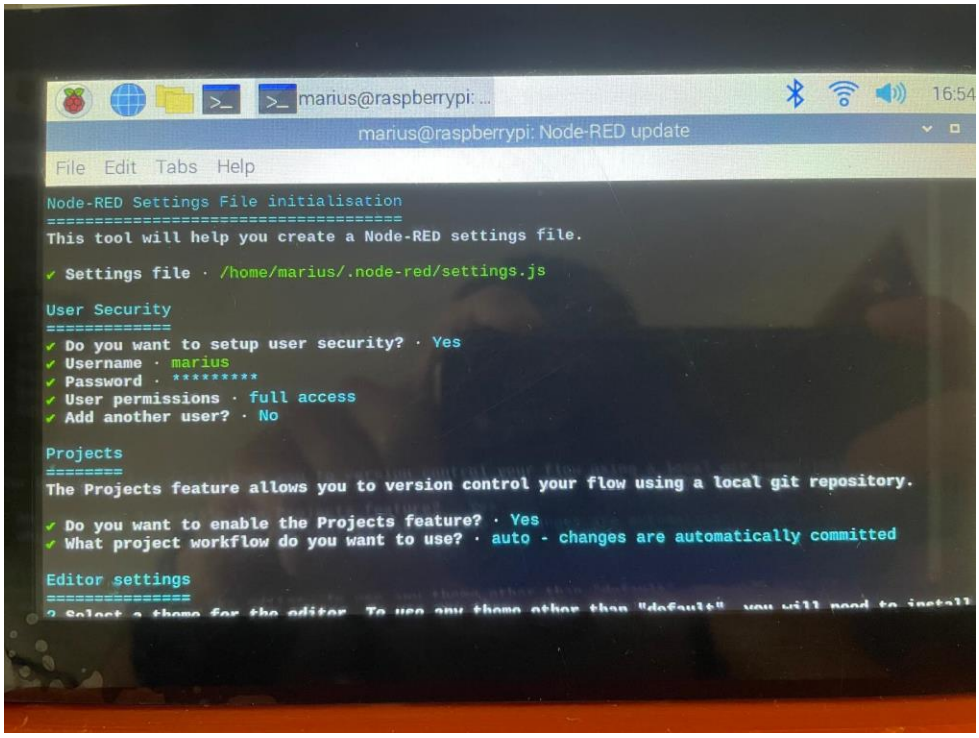

Fig 12. Configurare NodeRed

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

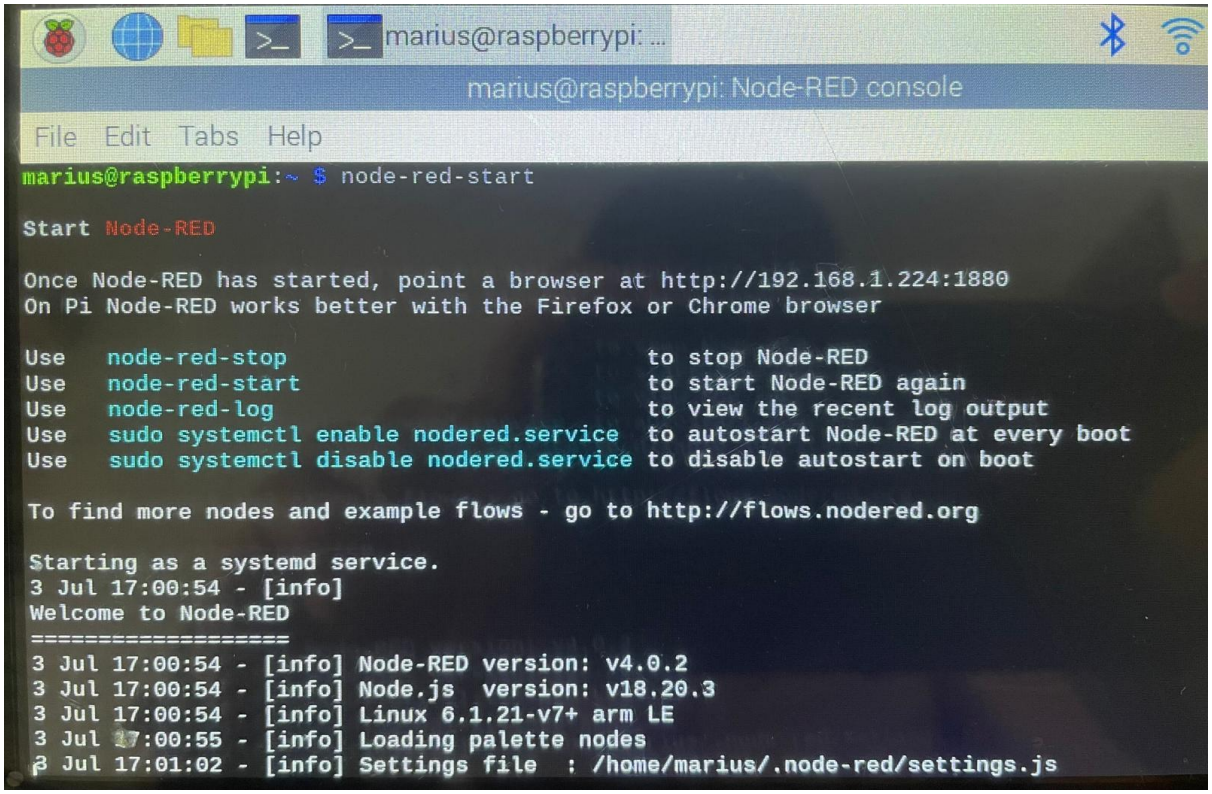

Fig 13. Pornire Node Red cu succes

!!! 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`

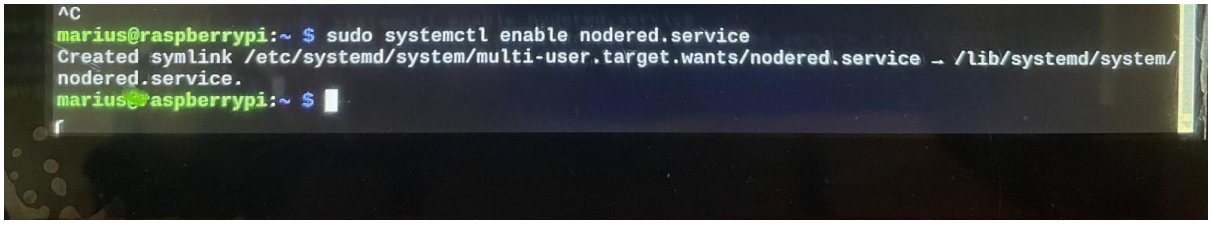

Fig 14. Activare NodeRed la pornire RaspPI

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

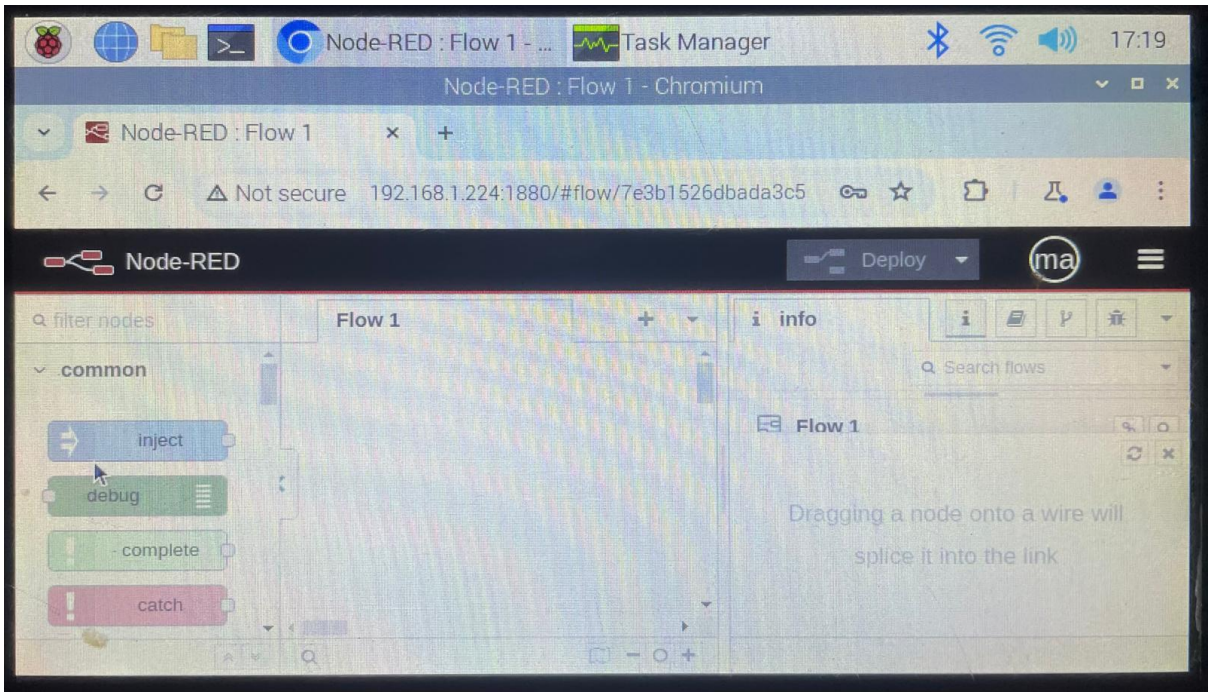

Fig 11. Node Red in browser intern RaspPi

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 obeserva cum functioneaza codul pe esp vizualizand consola:

Fig 12. Functionare comunicare ESP32 – server MQTT

‼‼ Instalam biblioteca PubSubClient.h de aici: PubSubClient - Arduino Reference.

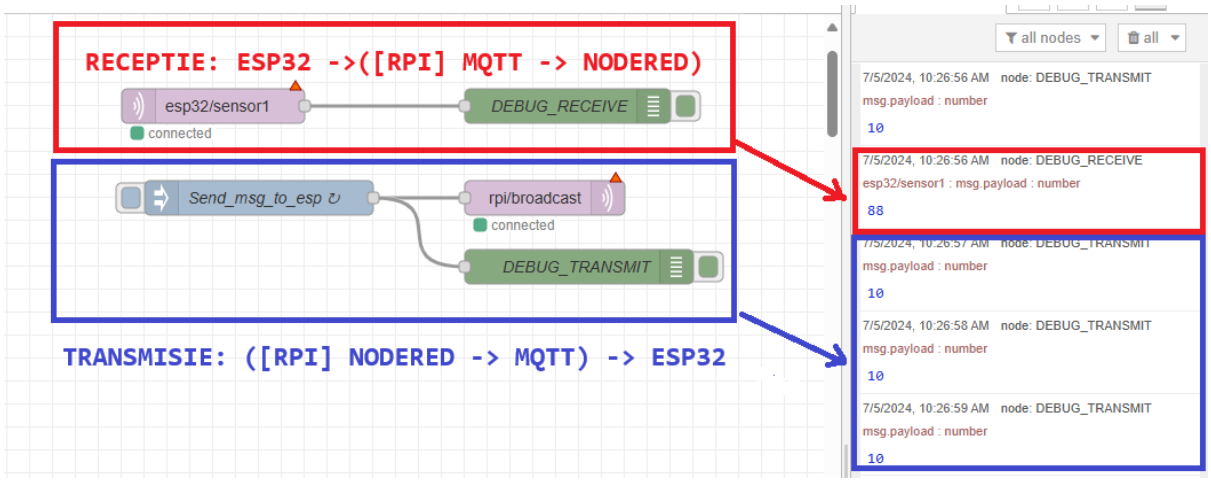

Fig 13. Vizualizare in NodeRed debug

⁉ 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`.

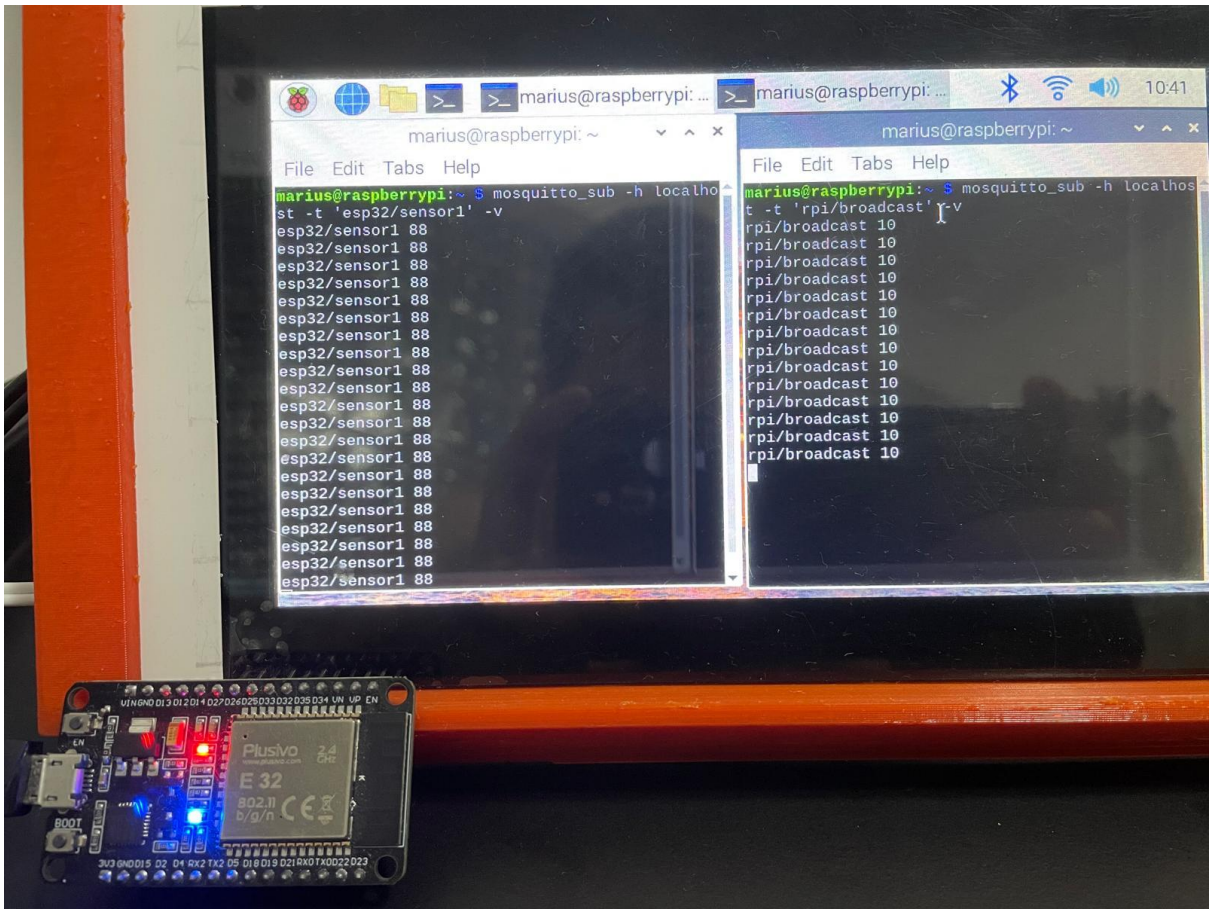

Fig 14. Functionare dispozitive

Folosind comanda in consola:

`mosquitto_sub -h localhost -t 'topic' -v`, am deschis doua terminale si putem observa trasmisia 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'`