

Plateforme véhiculaire embarquée multitâche basée sur les protocoles de communication I2C (Maître/Multi-Esclaves) et MQTT (Nuage/Noeud)

BADRY ZAKARIA

Filière : Genie informatique

Kénitra

email: zakaria.badry@uit.ac.ma

MEZIANE ZERHOUNI HASSAN

Filière Genie informatique

Kénitra

email: hassan.mezianezerhouni@uit.ac.ma

CHAGRI ANASS

Filière Genie informatique

Kénitra

email : anass.chagri@uit.ac.ma

hamza mouhawire

Filière : Genie informatique

Kénitra email:

hamza.mouhawire@uit.ac.ma

I. INTRODUCTION

Avec l'essor des véhicules intelligents et connectés, il devient essentiel de concevoir des plateformes embarquées capables de gérer efficacement les données et les actions liées à la surveillance et au contrôle du véhicule. Le présent rapport décrit la conception et l'implémentation d'une plateforme véhiculaire embarquée multitâche basée sur les protocoles de communication I2C (Maître/Multi-Esclaves) et MQTT (Nuage/Noeud). Cette plateforme est composée d'un maître, un Raspberry Pi, et trois esclaves Arduino, chacun responsable de tâches bien spécifiques .

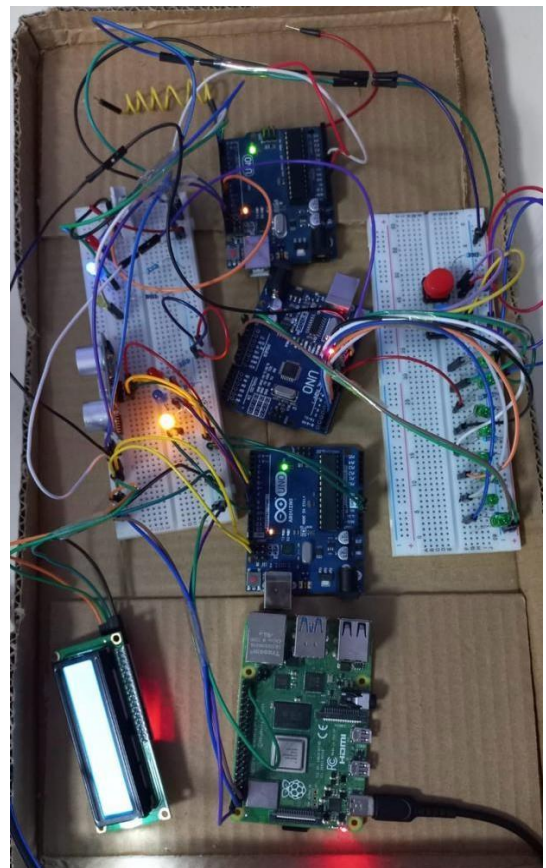
II. OBJECTIF

L'objectif principal de ce projet est de créer une plateforme capable de collecter et de traiter des données provenant de différents capteurs embarqués . Les capteurs sont capables de fournir des informations pertinentes , son environnement, . La plateforme est chargée de recevoir, de stocker, et de traiter ces données de manière efficace .

III. L'ARCHITECTURE DE LA PLATEFORME

La structure fondamentale de cette plateforme repose sur une hiérarchie maître-esclave, où le Raspberry Pi assume le rôle de maître tandis que trois Arduino jouent le rôle d'esclaves. Les échanges d'informations entre le maître et les esclaves sont orchestrés au moyen du protocole I2C, garantissant une communication efficace et bidirectionnelle. Parallèlement, la connectivité entre le nuage (cloud) et le nœud (la plateforme Raspberry Pi et ses esclaves Arduino) est établie grâce au protocole MQTT, permettant une communication fluide et instantanée, enrichissant ainsi la

plateforme de fonctionnalités de surveillance et de contrôle avancées.



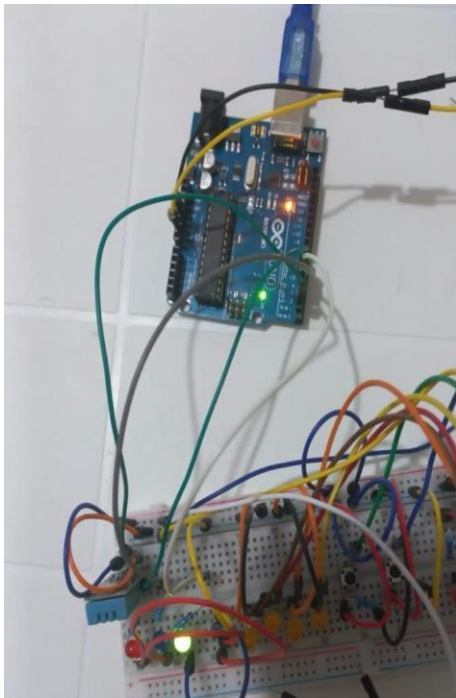
IV. LE MAITRE (RASPBERRY PI)

Le maître Raspberry Pi a plusieurs fonctions dans la plateforme. Il est responsable de :

- Gérer la communication I2C avec les esclaves Arduino, en leur envoyant des requêtes ou des commandes, et en recevant leurs données.
- Gérer la communication MQTT avec le nuage, en publiant les données des esclaves sur des topics spécifiques, et en s'abonnant à des topics pour recevoir des instructions ou des alertes du nuage.

-

Le premier esclave est équipé d'un capteur DHT11 permettant de mesurer la température et l'humidité. Ces données sont transmises au maître via le protocole I2C.



The screenshot shows the Arduino IDE interface. The main editor contains the following code:

```

42 digitalWrite(LED_F3, HIGH);
43 }
44 else{
45   digitalWrite(LED_F3, LOW);
46   digitalWrite(LED_F3, HIGH);
47   digitalWrite(LED_F3, LOW);
48   digitalWrite(LED_F3, HIGH);
49 }
50 delay(1000); // wait for a second
51

```

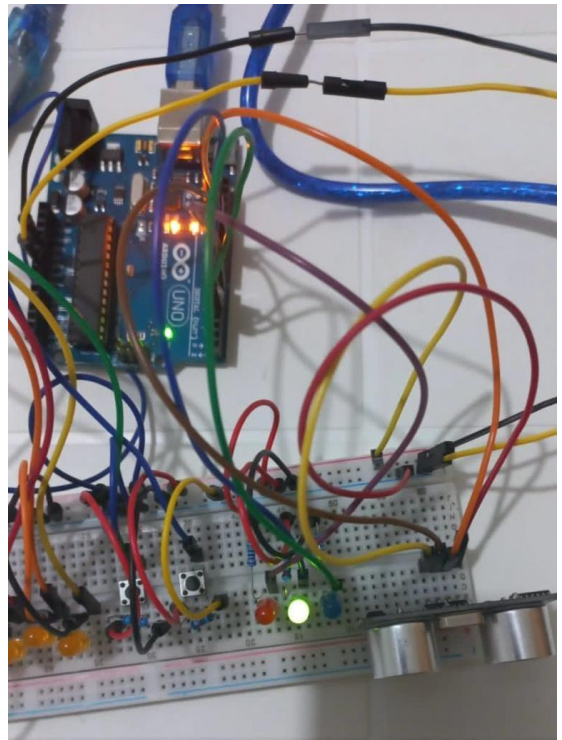
The Serial Monitor at the bottom shows the output of the program, displaying the state of LED_F3 (ON/OFF) every 100ms:

```

Transmitting 25.00 %
Transmitting 25.50 %
Transmitting 26.00 %
Transmitting 26.50 %
Transmitting 27.00 %
Transmitting 27.50 %
Transmitting 28.00 %
Transmitting 28.50 %
Transmitting 29.00 %
Transmitting 29.50 %
Transmitting 30.00 %
Transmitting 30.50 %
Transmitting 31.00 %
Transmitting 31.50 %
Transmitting 32.00 %
Transmitting 32.50 %
Transmitting 33.00 %
Transmitting 33.50 %
Transmitting 34.00 %
Transmitting 34.50 %
Transmitting 35.00 %
Transmitting 35.50 %
Transmitting 36.00 %
Transmitting 36.50 %
Transmitting 37.00 %
Transmitting 37.50 %
Transmitting 38.00 %
Transmitting 38.50 %
Transmitting 39.00 %
Transmitting 39.50 %
Transmitting 40.00 %
Transmitting 40.50 %
Transmitting 41.00 %
Transmitting 41.50 %
Transmitting 42.00 %
Transmitting 42.50 %
Transmitting 43.00 %
Transmitting 43.50 %
Transmitting 44.00 %
Transmitting 44.50 %
Transmitting 45.00 %
Transmitting 45.50 %
Transmitting 46.00 %
Transmitting 46.50 %
Transmitting 47.00 %
Transmitting 47.50 %
Transmitting 48.00 %
Transmitting 48.50 %
Transmitting 49.00 %
Transmitting 49.50 %
Transmitting 50.00 %
Transmitting 50.50 %
Transmitting 51.00 %
Transmitting 51.50 %
Transmitting 52.00 %
Transmitting 52.50 %
Transmitting 53.00 %
Transmitting 53.50 %
Transmitting 54.00 %
Transmitting 54.50 %
Transmitting 55.00 %
Transmitting 55.50 %
Transmitting 56.00 %
Transmitting 56.50 %
Transmitting 57.00 %
Transmitting 57.50 %
Transmitting 58.00 %
Transmitting 58.50 %
Transmitting 59.00 %
Transmitting 59.50 %
Transmitting 60.00 %
Transmitting 60.50 %
Transmitting 61.00 %
Transmitting 61.50 %
Transmitting 62.00 %
Transmitting 62.50 %
Transmitting 63.00 %
Transmitting 63.50 %
Transmitting 64.00 %
Transmitting 64.50 %
Transmitting 65.00 %
Transmitting 65.50 %
Transmitting 66.00 %
Transmitting 66.50 %
Transmitting 67.00 %
Transmitting 67.50 %
Transmitting 68.00 %
Transmitting 68.50 %
Transmitting 69.00 %
Transmitting 69.50 %
Transmitting 70.00 %
Transmitting 70.50 %
Transmitting 71.00 %
Transmitting 71.50 %
Transmitting 72.00 %
Transmitting 72.50 %
Transmitting 73.00 %
Transmitting 73.50 %
Transmitting 74.00 %
Transmitting 74.50 %
Transmitting 75.00 %
Transmitting 75.50 %
Transmitting 76.00 %
Transmitting 76.50 %
Transmitting 77.00 %
Transmitting 77.50 %
Transmitting 78.00 %
Transmitting 78.50 %
Transmitting 79.00 %
Transmitting 79.50 %
Transmitting 80.00 %
Transmitting 80.50 %
Transmitting 81.00 %
Transmitting 81.50 %
Transmitting 82.00 %
Transmitting 82.50 %
Transmitting 83.00 %
Transmitting 83.50 %
Transmitting 84.00 %
Transmitting 84.50 %
Transmitting 85.00 %
Transmitting 85.50 %
Transmitting 86.00 %
Transmitting 86.50 %
Transmitting 87.00 %
Transmitting 87.50 %
Transmitting 88.00 %
Transmitting 88.50 %
Transmitting 89.00 %
Transmitting 89.50 %
Transmitting 90.00 %
Transmitting 90.50 %
Transmitting 91.00 %
Transmitting 91.50 %
Transmitting 92.00 %
Transmitting 92.50 %
Transmitting 93.00 %
Transmitting 93.50 %
Transmitting 94.00 %
Transmitting 94.50 %
Transmitting 95.00 %
Transmitting 95.50 %
Transmitting 96.00 %
Transmitting 96.50 %
Transmitting 97.00 %
Transmitting 97.50 %
Transmitting 98.00 %
Transmitting 98.50 %
Transmitting 99.00 %
Transmitting 99.50 %
Transmitting 100.00 %

```

Le deuxième esclave est équipé d'un capteur ultrasonique pour mesurer la distance et la vitesse du véhicule. Il contrôle également trois lampes de signalisation (bleue pour loin, verte pour moyennement proche , rouge pour tres proche) en fonction de la distance mesurée.



- Execution ULTRASON :

```

1 #define BLUE_PIN 4 // BLUE LED pin
2 #define GREEN_PIN 5 // Green LED pin
3 #define RED_PIN 6 // Red LED pin
4 #define TRIG_PIN 11 // Ultrasonic sensor trigger pin
5 #define ECHO_PIN 12 // Ultrasonic sensor echo pin
6
7 #define DISTANCE_MIN 20 // Minimum distance threshold in centimeters
8 #define DISTANCE_MAX 20 // Maximum distance threshold in centimeters
9
10 void setup() {
11   pinMode(BLUE_PIN, OUTPUT);
12   pinMode(GREEN_PIN, OUTPUT);
13   pinMode(RED_PIN, OUTPUT);
14   pinMode(TRIG_PIN, OUTPUT);
15   pinMode(ECHO_PIN, INPUT);
16
17   Serial.begin(9600);
18 }
19
20 void loop() {
21   long duration;
22   float distance;
23
24   digitalWrite(TRIG_PIN, LOW);
25   delayMicroseconds(2);
26   digitalWrite(TRIG_PIN, HIGH);
27   delayMicroseconds(10);
28   digitalWrite(TRIG_PIN, LOW);
29   duration = pulseIn(ECHO_PIN, HIGH);
30   distance = duration * 0.034 / 2;
31
32   if (distance < DISTANCE_MIN) {
33     digitalWrite(BLUE_PIN, HIGH);
34     digitalWrite(GREEN_PIN, LOW);
35     digitalWrite(RED_PIN, LOW);
36   } else if (distance < DISTANCE_MAX) {
37     digitalWrite(BLUE_PIN, LOW);
38     digitalWrite(GREEN_PIN, HIGH);
39     digitalWrite(RED_PIN, LOW);
40   } else {
41     digitalWrite(BLUE_PIN, LOW);
42     digitalWrite(GREEN_PIN, LOW);
43     digitalWrite(RED_PIN, HIGH);
44   }
45
46   Serial.print("Distance: ");
47   Serial.println(distance);
48 }

```

```

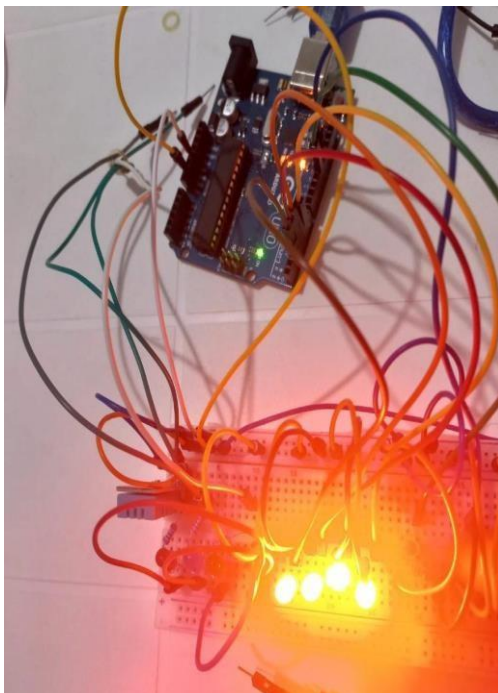
1 #include <Wire.h>
2
3 #define SLAVE_ADDRESS 0x08 // Define slave address
4 const int BUTTON_RIGHT = 12, BUTTON_LEFT = 13;
5 char states;
6 const int LEDS[] = {3, 4, 5, 6};
7
8 void requestEvent() {
9   Wire.write(states); // Send the state of the buttons to the master
10 }
11
12 void setup() {
13   for (int i = 3; i < 7; i++) {
14     pinMode(LEDS[i], OUTPUT);
15   }
16   pinMode(BUTTON_RIGHT, INPUT_PULLUP);
17   pinMode(BUTTON_LEFT, INPUT_PULLUP);
18 }
19
20 void loop() {
21   if (digitalRead(BUTTON_RIGHT) == LOW) {
22     states |= 1;
23   }
24   if (digitalRead(BUTTON_LEFT) == LOW) {
25     states |= 2;
26   }
27   requestEvent();
28 }

```

- Execution camera :

VII. L'ESCLAVE 3 ET CAMERA

Le troisième esclave assume la responsabilité de la gestion de la direction . Cette direction est visuellement signalée par des indicateurs lumineux qui défilent de manière dynamique, s'ajustant en temps réel en fonction de la direction sélectionnée par l'utilisateur à l'aide d'un bouton dédié.il y a aussi une camera qui capture en continue des photos et eux aussi sont envoyés au maître .



- Execution SIGNAL :

```

1 import cv2
2 import socket
3 import time
4
5 # Define the server IP and port
6 SERVER_IP = '192.168.1.100'
7 SERVER_PORT = 8080
8
9 # Create a socket
10 client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
11
12 # Connect to the server
13 client_socket.connect((SERVER_IP, SERVER_PORT))
14
15 # Send the captured photo to the server
16 with open('photo.jpg', 'rb') as file:
17     data = file.read()
18     client_socket.send(data)
19
20 # Close the socket
21 client_socket.close()

```

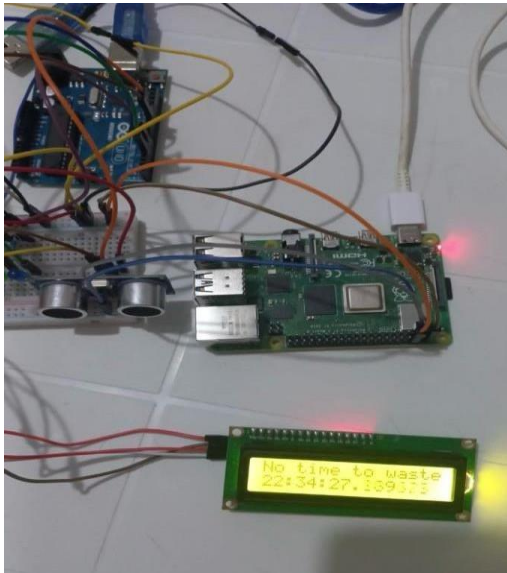
```

1 import cv2
2 import socket
3
4 # Define the server IP and port
5 SERVER_IP = '192.168.1.100'
6 SERVER_PORT = 8080
7
8 # Create a socket
9 server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
10
11 # Bind the socket to the IP and port
12 server_socket.bind((SERVER_IP, SERVER_PORT))
13
14 # Listen for incoming connections
15 server_socket.listen(5)
16
17 # Accept an incoming connection
18 (client_socket, address) = server_socket.accept()
19
20 # Receive data from the client
21 data = client_socket.recv(1024)
22
23 # Decode the data
24 image = cv2.imdecode(data, cv2.IMREAD_COLOR)
25
26 # Display the image
27 cv2.imshow('Received Image', image)
28 cv2.waitKey(0)
29 cv2.destroyAllWindows()
30
31 # Close the socket
32 client_socket.close()

```

VIII. GESTION DE LA COMMUNICATION I2C

Le maître (Raspberry Pi) communique avec les esclaves (Arduino) via le protocole I2C, assurant une transmission de données bidirectionnelle. Des adresses I2C uniques sont attribuées à chaque esclave pour permettre une communication sélective.



- Détection lcd i2c

```

zakaria@raspberrypi:~$ login as: zakaria
zakaria@192.168.250.70's password:
Linux raspberrypi 6.1.0-rpi4-rpi-v6 #1 SMP PREEMPT Debian 1:6.1.54-1+rpt2 (2023-11-05) aarch64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

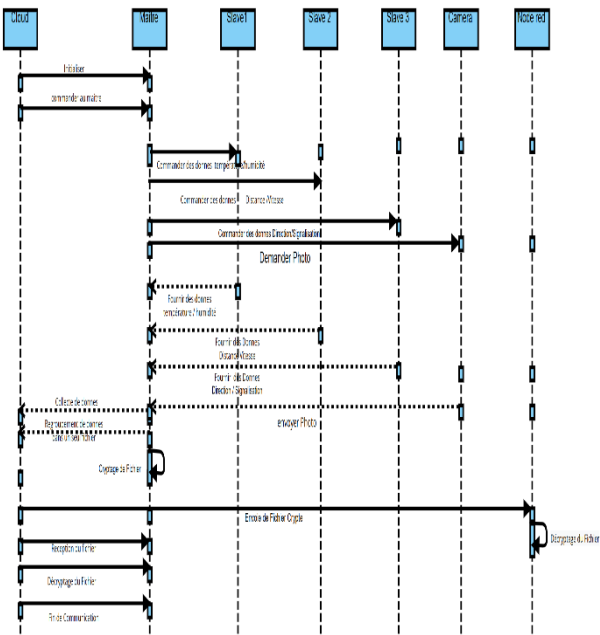
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
last login: Tue Nov 14 22:54:29 2023
zakaria@raspberrypi:~$ i2cdetect -y 1
    0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:  -- -- -- -- -- -- -- -- -- -- -- -- -- --
01:  -- -- -- -- -- -- -- -- -- -- -- -- -- --
02:  -- -- -- -- -- -- -- -- -- -- -- -- -- --
03:  -- -- -- -- -- -- -- -- -- -- -- -- -- --
04:  -- -- -- -- -- -- -- -- -- -- -- -- -- --
05:  -- -- -- -- -- -- -- -- -- -- -- -- -- --
06:  -- -- -- -- -- -- -- -- -- -- -- -- -- --
07:  -- -- -- -- -- -- -- -- -- -- -- -- -- --
08:  -- -- -- -- -- -- -- -- -- -- -- -- -- --
zakaria@raspberrypi:~$
  
```

```

pcpy X
sok > pcpy > ...
1 import socket
2
3 # Configuration du client
4 HOST = '192.168.250.70' # Adresse IP de la Raspberry Pi
5 PORT = 12345
6
7 def send_request():
8     with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as client_socket:
9
10 zakaria@raspberrypi:~/code
11 (venv) zakaria@raspberrypi:~/code $ cat all data.txt
12 2023-11-15 22:37:52 - Direction: 48, Distance: 255, Température: 255
13 2023-11-15 22:37:52 - Direction: 48, Distance: 255, Température: 255
  
```

La plateforme utilise MQTT ou dans notre cas Socket pour la communication Cloud-Maitre. Le nuage envoie des ordres au maître via le broker MQTT, lui demandant de lui fournir les données des esclaves. Le maître reçoit les ordres , et les exécute en demandant les données aux esclaves via I2C, il collecte les données des esclaves en simultané, les compresse en un seul fichier, et le crypte . Le maître utilise ensuite un socket pour envoyer le fichier crypté au cloud via le broker MQTT. Le nuage reçoit le fichier crypté , le décrypte , et le décompresse pour extraire les données. Puis le cloud communique les données à NODE-RED pour la création d'un tableau de bord interactif affichant les données de manière conviviale.

IX. Diagramme de séquence



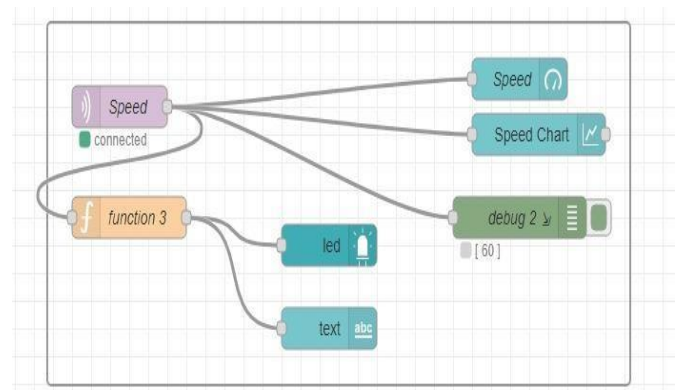
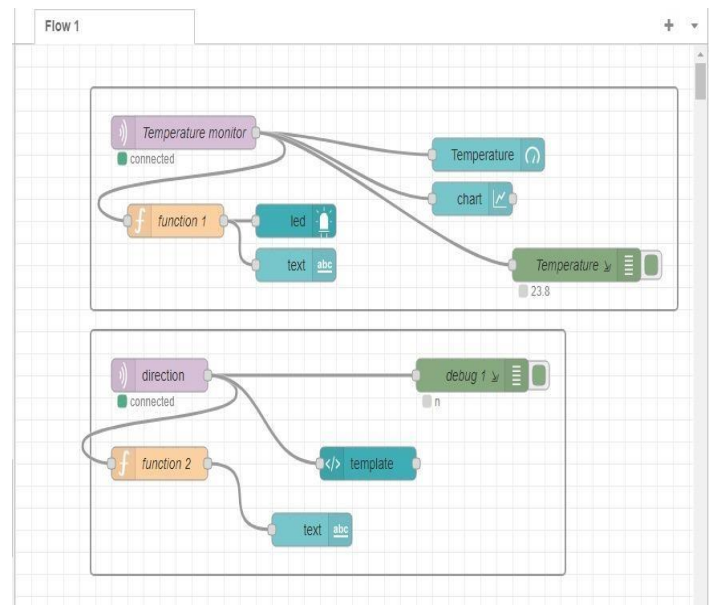
X. MISE EN OEUVRE LOGICIELLE (INTEGRATION DU PROTOCOLE MQTT , SOCKET ET DE NODE-RED)

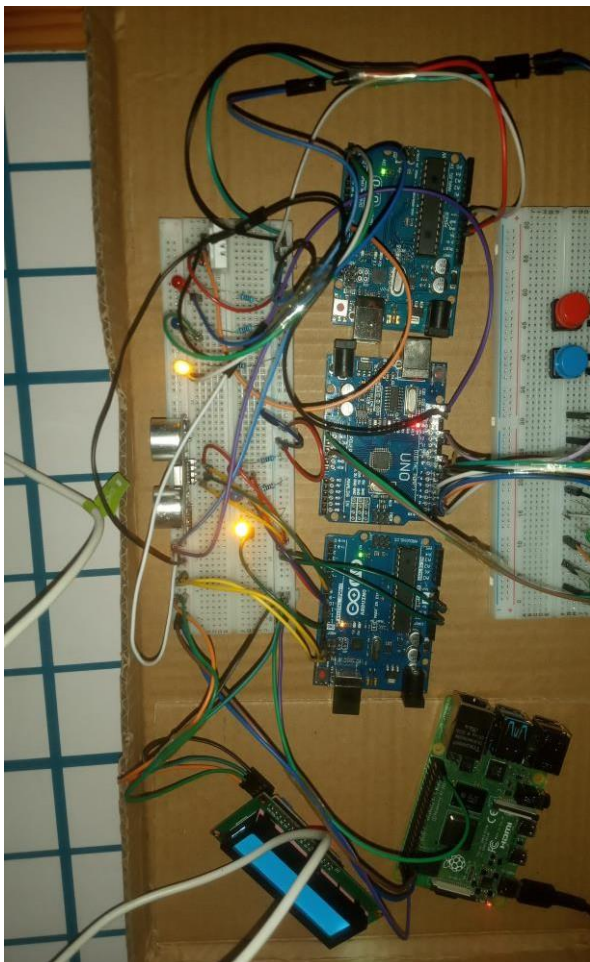
- Conception NODE-RED :

XI. RESULTATS ET TESTS

Les tests effectués ont validé la fiabilité de la plateforme dans des scénarios simulés. L'étalonnage des capteurs, la vérification de la communication I2C, et la réactivité du système ont été confirmés avec succès.

- LES ESCLAVES détectés :





XII. CONCLUSION

La plateforme véhiculaire embarquée multitâche, combinant I2C, MQTT, et Node-RED, offre une solution complète pour la surveillance et le contrôle du véhicule. L'utilisation de Node-RED facilite la visualisation des données, offrant une interface utilisateur intuitive pour une gestion optimale des informations. Cette intégration réussie ouvre la voie à des améliorations futures et à des applications étendues de la plateforme.