Treball Final de Grau

Robot Car kit: Adaptació a IoT

Grau en Enginyeria de Sistemes TIC
Curs 18/19

Autor: Baldelomar Garcia, Abel Santiago
Director: Escobet Canal, Antoni
Data: 5 de Juliol de 2019
Localitat: Manresa, Barcelona

Resum

El Robot Car Kit és un automòbil robòtic intel·ligent de la casa Elegoo amb quatre rodes motrius, un mòdul de seguiment de línia, un sensor d'ultrasons dirigible, un mòdul Bluetooth i un mòdul de control remot per infrarojos. Així mateix, aquesta plataforma està basada en el microcontrolador Arduino Uno.

La finalitat principal del projecte és adapta la plataforma al microcontrolador Arduino MKR 1010 WiFi. Aquesta placa està equipada amb un potent processador ARM de 32 bits i permet la connexió a una xarxa WiFi mitjançat el seu mòdul ESP32. L'objectiu recau en el fet d'adaptar la nova placa a certs dispositius perifèrics que disposa la plataforma, afegir uns codificadors als motors (encoders) per tal de poder tancar el llaç de control dels motors, i alhora, oferir una solució de comunicació versàtil entre el Robot Car Kit i una aplicació o sistema extern mitjançat una xarxa WiFi. Els dispositius perifèrics en questió són el mòdul de seguiment de línia, el sensor d'ultrasons dirigible i les rodes motrius.

En resum, aquest nou model del Robot Car Kit, es podrà controlar amb algun dispositiu connectat a la xarxa wifi (PC, telèfon mòbil, tauleta, ...), i en conseqüència, poder dissenyar aplicacions o sistemes orientats a la IoT.

Abstract

The Robot Car Kit is an intelligent robotic car of the brand Elegoo that has four driving wheels, a line tracking module, a dirigible ultrasound sensor, a bluetooth module and a module of infrared remote control. In addition, this platform is based on Arduino One microcontroller board.

The main objective of this project is the adaptation of the platform to the Arduino MKR WiFi 1010 microcontroller board. This board is equipped with a powerful ARM 32-bit processor and allows connection to a WiFi network through its ESP32 module. The objective lies in the fact that the new board adapts to certain peripheral devices that the platform has, add some encoders for the motors in order to close the motors control loop, an at the same time, provide a versatile communication solution between the Robot Car Kit and an application or an external system through a WiFi network. The peripheral devices in question are the line tracking module, the dirigible ultrasound sensor and the drive wheels.

In summary, this new system of Robot Car Kit will be able to be controlled by a connected device/system to a WiFi network (PC, telèfon mòbil, tauleta, ...), and therefore, be able to design application and systems oriented to IoT.

Índex

1 Introducció **7**
1.1 Objectius . . . 7
1.2 Estat incial del projecte . . . 9

2 Capa de comunicació **13**
2.1 Missatges HTTP . . . 14
2.2 Llibreria Servidor . . . 17
2.2.1 Visió general . . . 17
2.2.2 Documentació llibreria TCPSERVER . . . 18
2.2.3 Exemples d'ús de la llibreria TCPSERVER . . . 26
2.3 Llibreria Client . . . 30
2.3.1 Visió general . . . 30
2.3.2 Documentació llibreria TCPCLIENT . . . 30
2.3.3 Exemples d'ús de la llibreria TCPCLIENT . . . 38

3 Mòduls del Robot Car Kit **42**
3.1 Mòdul de les Rodes Motrius . . . 42
3.1.1 Funcionament de l'electrònica implicada . . . 44
3.1.2 Obtenció de la velocitat RPM de les rodes . . . 50
3.1.3 Assignació senyal PWM a RPM . . . 53
3.1.4 Control del posicionamet de la Roda . . . 57
3.1.5 Documentació llibreria MOTOR . . . 58
3.2 Mòdul d'Ultrasò Dirigible . . . 72
3.2.1 Servo motor SG90 . . . 72
3.2.2 Ultrasò HC-SR04 . . . 74
3.2.3 Documentació llibreria ULTRASOUND . . . 76
3.3 Mòdul de Sensor de Línia . . . 77
3.3.1 Sensor HCARDU0005 . . . 79
3.3.2 Documentació llibreria LINETRACKING . . . 80
3.4 Connexió Mòduls Robot Car Kit amb Arduino MKR WiFi 1010 . . . 82

4 Aplicacions de mostra **83**
4.1 Primera aplicació de mostra . . . 83
4.1.1 Servidor Web . . . 85
4.1.2 Capa de comunicació Robot Car Kit . . . 89
4.1.3 Programes Robot Car Kit . . . 92
4.2 Segona aplicació de mostra . . . 100
4.2.1 Màquina d'estats servidor web . . . 101

4.2.2 Màquina d'estats programa rodes motrius 103

5 Conclusions **105**

6 Referències **106**

7 Bibliografia **107**

8 Llista d'acrònims **108**

9 Annexos **109**

9.1 Esquema circuital final del Robot Car Kit i l'Arduino MKR per les aplicacions de mostra 109

Índex de figures

1 Robot Car Kit amb l'Arduino MKR Wifi 1010 8
2 Robot Car Kit amb l'Arduino MKR WiFi 1010 9
3 Arduino MKR WiFi 1010 - PinOut 11
4 Versió HTTP/1.x 14
5 Parts d'una petició i resposta HTTP 15
6 Exemple petició HTTP 16
7 Exemple resposta HTTP 16
8 Exemple d'us TCPSERVER-SOCKET 26
9 Exemple d'us TCPSERVER-SOCKET 27
10 Exemple d'us TCPSERVER-HTTP 28
11 Exemple d'us TCPSERVER-HTTP 29
12 Exemple d'us TCPSERVER-HTTP 29
13 Exemple d'us TCPCLIENT-SOCKET 38
14 Exemple d'us TCPCLIENT-SOCKET 39
15 Exemple d'us TCPCLIENT-HTTP 40
16 Exemple d'us TCPCLIENT-HTTP 41
17 Exemple d'us TCPCLIENT-HTTP 41
18 DC Motor 42
19 Micro DC Motor with Enconder-SJ01 43
20 Driver L298N 43
21 Driver L298N connectat als motors 44
22 Configuració dels senyals EN1 i EN2 45
23 Encoder SJ01 amb dos sensors Hall 46
24 Sensors Hall Actiu 46

25 Sensors Hall No Actiu . . . 46
26 Encoder-Seqüència senyals desfaçats . . . 47
27 Polsos dels canals A i B del Encoder SJ01 . . . 47
28 Encoder-Matriu Gray . . . 48
29 Connexió motor-enconder amb l'Arduino . . . 49
30 Freqüències AnalogWrite . . . 53
31 Syntax AnalogWrite . . . 54
32 PWM AnalogWrite . . . 54
33 Assignació valors PWM-RPM . . . 55
34 Valors RPM acceptats per la llibreria MOTOR . . . 56
35 Valors RPM acceptats per la llibreria MOTOR . . . 62
36 Valors RPM acceptats per la llibreria MOTOR . . . 69
37 Ultrasò dirigible . . . 72
38 Servo SG90 . . . 72
39 Servo Taula d'angles-cicle de treball senyal PWM . . . 73
40 Exemple Servo.h . . . 73
41 Principals aspectes tècnics HC-SR04 . . . 74
42 Timing dels senyals trigger i echo . . . 74
43 Mòdul de Sensor de Línia . . . 77
44 Mòdul de Sensor de Línia-localització en el Robot . . . 77
45 Algorisme seguidor de línia simple . . . 78
46 Potenciòmetre del sensor HCARDU0005 . . . 79
47 Disposició mòdul sensor de línia . . . 80
48 BOB 11978 convertidor lògic de nivell . . . 82
49 Aplicació 1: xarxa de comunicació . . . 84
50 Sistema MVC . . . 85
51 Aplicació 1: main-webPage . . . 86
52 Aplicació 1: vista Programa 1-webPage . . . 87
53 Aplicació 1: vista Programa 2-webPage . . . 87
54 Aplicació 1: vista Programa 3-webPage . . . 87
55 Arquitectura WebSocket . . . 88
56 FSM TCPClient . . . 90
57 FSM TCPClient-llegenda . . . 90
58 Aplicació 1: vista Programa 1-webPage . . . 92
59 FSM Programa1 . . . 94
60 FSM Programa1-llegenda . . . 94
61 Aplicació 1: vista Programa 2-webPage . . . 95
62 FSM Programa 2 . . . 96
63 FSM Programa 2-llegenda . . . 96
64 Aplicació 1: vista Programa 3-webPage . . . 97

65 Pla cartesià Programa 3 98
66 FSM Programa 3 99
67 FSM Programa 3-llegenda 99
68 Aplicació 2 xarxa de comunicació 100
69 Aplicació 2 Màquina d'estats Servidor Web 101
70 Aplicació 2 Màquina d'estats Servidor Web 101
71 FSM Programa1 104
72 FSM Programa1-llegenda 104
73 Esquema Circuital de connexions 109

1 Introducció

La Internet of Things (IoT), o Internet de les Coses, representa avui en dia una revolució tecnològica que ha transformat i transformarà en major grau la nostra forma d'interactuar amb el món. Les aplicacions derivades de la IoT, tant de forma directa o no, han donat lloc a diversos camps d'estudi com ara smart cities, smart home, smart energy, smart health, industry 4.0, connected vehicle i moltes altres vessants que han permès una millor en l'eficiència i gestió a l'hora de dissenyar un sistema. Així doncs, és vital avançar en l'estudi del disseny de sistemes orientats a la IoT, o en l'adaptació de sistemes no dissenyats originalment amb aquest fi. Són per aquests motius, pels quals aquest treball té com a propòsit l'adaptació del sistema Robot Car Kit[comf] (automòbil robòtic intel·ligent) a la IoT mitjançat l'ús del controlador Arduino MKR 1010 WiFi[comb].

1.1 Objectius

Entre el gran ventall de conceptes que comprèn i defineixen la IoT, es troba, la definició d'una xarxa d'objectes físics, màquines o dispositius electrònics que utilitzen sensors i interfases de programació d'aplicacions (API) amb l'objectiu de permetre la comunicació i intercanvi de dades per Internet o de forma remota. Això, permet l'obtenció i gestió de dades mitjançats sensors fins al big data, anàlisi predictius, desenvolupament de sistemes enfocats a machine learning o artificial inteligence (AI), entre moltes altres aplicacions. D'aquesta forma, i tenint en compte els conceptes previs, s'estableixen els objectius del projecte. El primer objectiu d'aquest treball és la creació d'una capa de comunicació per l'Arduino MKR WiFi 1010 basat en les comunicacions socket[coml] pel Protocol de Control de Transmisió (TCP/IP)[comj] i missatges de Hypertext Transfer Protocol (HTTP)[mdnb], les quals permeten la creació d'APIs enfocats a la comunicació entre el Robot Car Kit i una xarxa WiFi dintre dels protocols fundamentals de comunicació en Internet. El segon objectiu és el desenvolupament de llibreries o APIs del mòdul de seguiment de línia, del mòdul d'ultrasons dirigible (fent ús d'un servomotor) i les rodes motrius (incorporats amb un encoder), amb el fi de poder facilitar la implementació de funcionalitats d'interès com ara, el control de la velocitat de les rodes o el seu posicionament, l'obtenció de dades dels sensors, l'obtenció de distàncies per ultrasò o l'aplicació d'un controlador Proporcional, Integratiu i Derivatiu (PID) pel seguidor de línia, entre altres. Finalment, com a tercer objectiu, es té la creació de sistemes (programes) o aplicacions de mostra sobre la funcionalitat dels dos objectius previs.

Per resumir, en les següents seccions a la introducció es desenvoluparan els objectius prèviament explicats. En primer lloc, es tractarà la secció **Capa de comunicació**, on es descriurà l'objectiu d'oferir una via de comunicació entre els mòduls del Robot Car Kit i una xarxa WiFi amb l'Arduino MKR WiFi 1010, a més es descriuen les llibreries TCPSERVER i TCPCLIENT que permeten programar l'Arduino MKR WiFi 1010 en mode servidor o client[comi], tant socket com web (capacitat per gestionar missatges HTTP). En segon lloc, es tractarà la secció **Mòduls del Robot Cat Kit**, en el qual es detallarà el funcionament dels mòduls de les rodes motrius, del sensor d'ultrasò dirigible i el seguidor de línia, a més s'explicarà les llibreries desenvolupades i corresponents als mòduls, es a dir, la llibreria MOTOR, la llibreria ULTRASOUND i la llibreria LINETRACKING. Per finalitzar, es tractarà la secció **Aplicacions de mostra**. Les aplicacions de mostra es componen de dos sistemes o aplicacions que permeten observar en funcionament les llibreries desenvolupades en aquest projecte. Els dos sistemes tenen l'objectiu de comunicar el Robot Car Kit mitjançat la capa de comunicació oferida per les llibreries TCPSERVER i TCPCLIENT. La primera aplicació es tracta del desenvolupament d'una aplicació web amb la qual poder controlar diversos programes implementats sobre el Robot Car Kit, en aquest cas, la capa de comunicació serà implementada amb la llibreria TCPCLIENT. Pel que fa a la segona aplicació, es tracta del desenvolupament d'un servidor web en l'Arduino MKR WiFi 1010 amb la llibreria TCPSERVER, a partir del qual es controlarà un programa sobre les rodes motrius del Robot Car Kit.

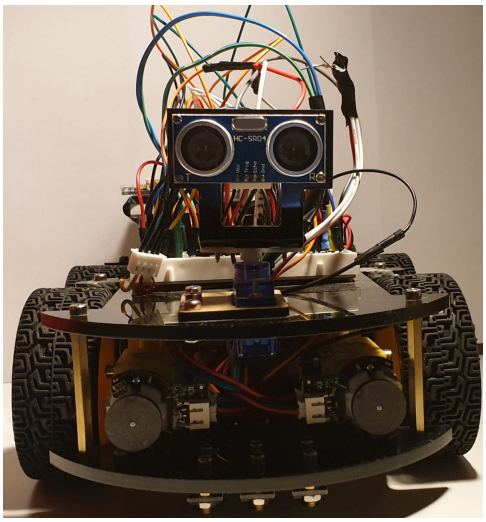

Figura 1: Robot Car Kit amb l'Arduino MKR WiFi 1010[1]

[1]Font: imatge pròpia

1.2 Estat incial del projecte

A continuació s'explicarà l'estat inicial o punt de partida del projecte en relació a la part hardware i la part software.

Pel que fa a la part hardware, el projecte està compost per quatre blocs, els quals són el Robot Car Kit, l'Arduino MKR WiFi 1010, el convertidor de nivell lògic BOB-11978 i un ordinador. Seguidament es detallarà quin paper tenen en el projecte.

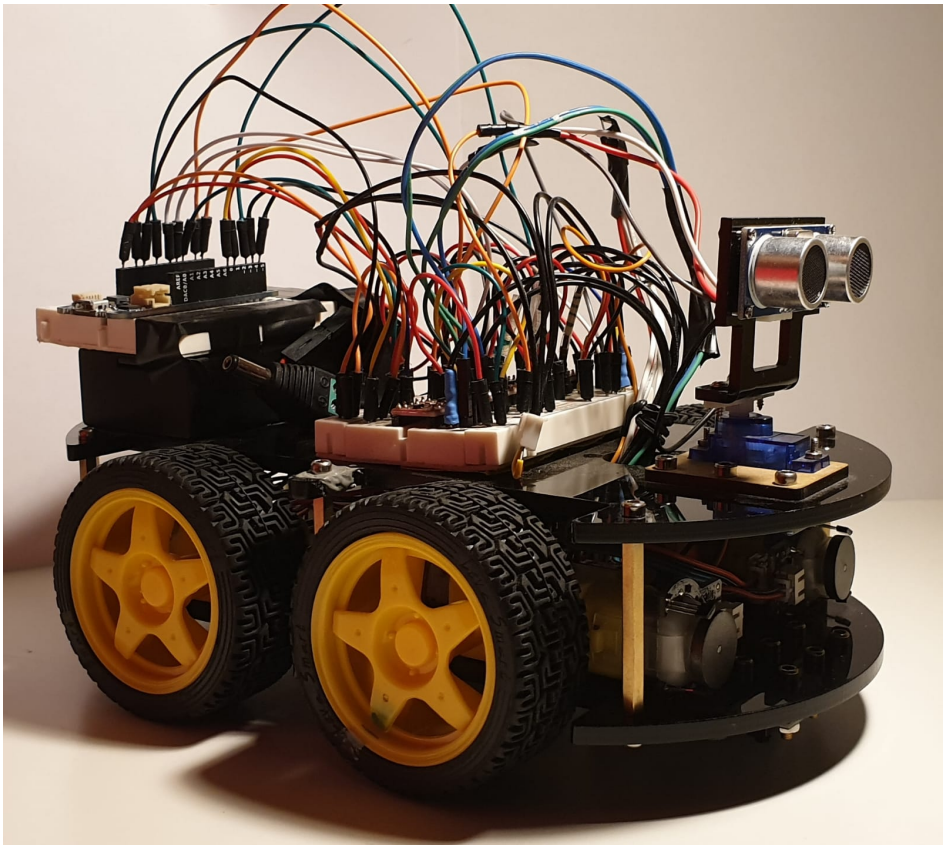

Figura 2: Robot Car Kit amb l'Arduino MKR Wifi 1010[2]

[2]Font: imatge pròpia

El Robot Car Kit és un automòbil robòtic intel·ligent de la casa Elegoo amb quatre rodes motrius, un mòdul de seguiment de línia, un sensor d'ultrasons dirigible, un mòdul Bluetooth i un control remot d'infrarojos. Aquesta plataforma està basada en la placa Arduino Uno, la qual serà substituïda per l'Arduino MKR WiFi 1010. En aquest projecte només es treballarà amb les rodes motrius, el mòdul de seguiment de línia i el sensor d'ultrasons dirigible, ja que, la comunicació del Robot Car Kit amb qualsevol aplicació exterior recau en la incorporació del mòdul WiFi integrat en la placa Arduino MKR WiFi 1010. En relació a les rodes motrius, el mòdul està compost per dos motors simples de corrent contínua amb engranatges incorporats, dos micro motors de corrent continua amb caixa d'engranatge i encoders de quadratura incorporats a l'eix del motor, el qual juntament amb el controlador de motors L298N permeten la creació d'un llaç de control sobre les rodes amb el fi de tenir el control sobre les velocitats i posicionaments dels motors mitjançats senyals Pulse Width Modulation (PWM). Cal especificar, que el Robot Car kit inicialment tenia dos motors simples de corrent continua en la seva part frontal, aquests s'hi han substituït pels dos micro motors amb encoders amb el fi de tancar el llaç de control sobre les rodes motrius. El mòdul de seguiment de línia està compost per tres sensors KY-033, els quals són sensors seguidors de línia d'infrarojos basats en el TCRT5000 amb un potenciòmetre per ajustar la sensibilitat. Bàsicament, el sensor KY-033 capta la recepció de llum emès per un díode infraroig, on la sensibilitat de la recepció de llum s'ha d'ajustar amb un potenciòmetre. Finalment, el mòdul d'ultrasons dirigible està compost pel sensor d'ultrasons HC-SR04 i el servo motor SG90. Per una banda, el sensor HC-SR04 permet la mesura de distàncies bassat en la mesura del temps de l'enviament i la recepció d'un pols sonor quan rebota contra un objecte. Per altra banda, el servo motor SG90 permet girar el sensor d'ultransons dins d'un rang de 0 graus a 180 graus.

En quant l'Arduino MKR WiFi 1010 és una placa equipada amb el mòdul ESP32 fabricat per U-BLOX i està orientat al desenvolupament i prototipatge d'aplicacions de IoT basats en WiFi. Principalment, està compost per tres blocs, el microcontrolador SAMD21 Cortex-M0 que ofereix una potència computacional de 32 bits d'arquitectura ARM, el mòdul Wifi ESP32 amb 2.4 GHz de baixa potència i el Cryptochip d'autentificació ECC508 per una comunicació segura. A més, el MKR WIFI 1010 funciona a 3.3V, el qual comporta l'ús de convertidors de nivell pel bon funcionament en les comunicacions de tipus TTL amb la majoria de sensors del Robot Car Kit. En referència al convertidor de nivell lògic, es tracta del model BOB-11978, que a partir de l'ús de transistors MOSFET de canal N permet la conversió de voltatges de 3.3 a 5 volts i a la inversa.

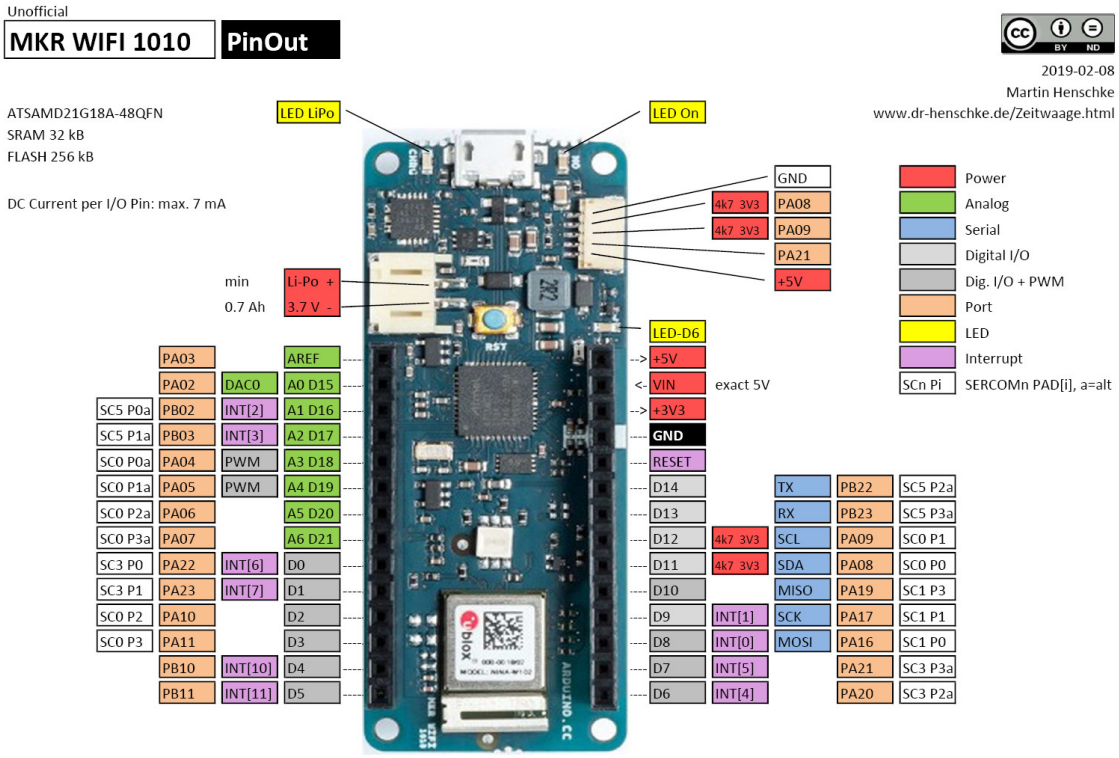

Figura 3: Arduino MKR WiFi 1010 - PinOut[3]

[3]Font: `www.dr-henschke.de/Zeitwaage.html`

En última instància tenim l'ordinador, el qual està pensant per actuar com client o servidor en una connexió socket (TCP/IP) o com servidor web o client web en comunicacions per missatges HTTP. Cal destacar, que en cas d'actuar com servidor web aquest també pot incloure un sistema de base de dades segons els requeriments de l'aplicació.

Respecte al desenvolupament i programació del software del projecte relacionat amb el Robot Car Kit i l'Arduino MKR WiFi 1010, es a dir, les llibreries del mòdul de les rodes motrius, el mòdul de seguiment de lína, el mòdul d'ultrasò dirigible i el mòdul WiFi (capa de comunicació), es farà amb l'ús del software d'Arduino (IDE) i les seves llibreries pertinents sobre el microcontrolador Arduino MKR WiFi 1010. En concret, s'utilitzarà la llibreria d'Arduino WiFiNINA[comd] per la implementació de les funcionalitats de l'anomenada capa de comunicació d'aquest projecte. La llibreria WiFiNINA permet la configuració de l'Arduino MKR WiFi 1010 com servidor acceptant connexions entrants o com a client reaitzant connexions a un servidor extern. Entre les funcionalitats que ofereix tenim la WiFi Class, la Server class, la Client class i la UDP class. La primera permet la connexió a un punt d'accés WiFi. La segona, permet la creació de servidors que poden enviar i rebre informació als clients connectats. La tercera, la creació de clients que poden connectar-se a servidors i enviar i rebre informació; i l'última, l'abilitació d'enviar i rebre missatjes pel protocol UDP. A més, la llibreria suporta l'encriptació personal de tipus WEP y WAP2.

En relació amb l'ordinador, s'utilitzarà el Framework Laravel juntament amb la llibreria Ratchet (llibreria per Websockets) per la creació d'un servidor web apte per aplicacions a temps real. També, es farà ús de la base de dades MySQL per la implementació d'una de les aplicacions de mostra i el programa PostMan o un Navegador web per realitzar peticions HTTP com client. Per finalitzar, s'utilitzarà el llenguatge de programació PYTHON per la creació d'un servidor de tipus socket (TCP/IP) per la implementació d'una de les aplicacions de mostra.

2 Capa de comunicació

Tenint en compte, que probablement un dels aspectes claus en una aplicació o sistema basat en IoT recau en el fet d'establir una comunicació amb una xarxa WiFi, i conseqüentment amb Internet, fa que un dels principals objectius d'aquest projecte sigui oferir una solució flexible en la comunicació existent entre els components que conformen el Robot Car Kit i una xarxa WiFi, o bé un sistema que tingui accés al mateix.

Arran d'aquest fet, el desenvolupament de la capa de comunicació d'aquest projecte s'ha focalitzat de cara a poder establir una comunicació de tipus sockets o mitjançats missatges de Hypertext Transfer Protocol (HTTP) sobre l'Arduino MKR WiFi 1010, el qual incorpora un mòdul WiFi ESP32. Les connexions sockets basades en el protocol TCP/IP és un dels models àmpliament utilitzats dintre del sistema de capes d'Internet, també conegut com a capa de xarxa, i té la finalitat d'establir una comunicació fiable entre processos d'extrem a extrem pel que fa a la capa de transport. És per aquest motiu, que d'establiment d'una comunicació de xarxa basat en el model TCP/IP comporta a ser una solució molt versàtil i flexible orientada a futures aplicacions de IoT. Tot i això, una gran part d'aplicacions IoT són dissenyats per poder interactuar o establir comunicacions amb la World Wide Web (WWW) pels seus grans beneficis, com l'accés directe a l'aplicació a partir d'un navegador. Per aquesta raó, és crucial poder englobar dintre la capa de comunicació del projecte el Protocol de Transferència d'Hipertext (HTTP), el qual és el protocol de comunicació que permet les transferències d'informació en la World Wide Web. D'aquesta forma, la possibilitat d'atendre o dur a terme peticions HTTP permet el desenvolupament Interfases de Programació d'Aplicacions o APIs, en concret d'APIs web, tant en la part de fronted o backend de l'aplicació a desenvolupar.

Cal esmentar, que el protocol TCP/IP no estableix una jerarquia mestre/esclau en les comunicacions, no obstant això, les aplicacions basades en comunicacions TCP/IP i missatges HTTP utilitzen el model client-servidor. On el client realitza peticions sobre uns recursos o serveis al servidor, el qual és l'encarregat d'oferir una resposta a les peticions dels diversos clients. En conseqüència, l'anomenada capa de comunicació d'aquest projecte estarà format per dues llibreries, una orientada a desenvolupar el role de client i l'altre el role de servidor, les quals podran establir els dos tipus de comunicacions anteriorment explicats, es a dir, comunicacions per sockets (TCP/IP) i mitjançades peticions HTTP.

2.1 Missatges HTTP

Tot seguit, es presentarà la informació rellevant sobre els missatges HTTP per la comprensió de les llibreries TCPSERVER i TCPCLIENT. Per més informació podeu accedir a la documentació sobre missatges HTTP de Mozilla [mdnb].

Els missatges HTTP són un mètode utilitzat per intercanviar dades entre servidors i clients. Els missatges HTTP es classifiquen o es diferencien en dos tipus, **peticions o respostes** (*requests o responses*). Les peticions són enviades pel client al servidor, per demanar l'inici d'una acció. Pel que fa a les respostes, són les respostes oferides pel servidor a l'inici de les accions demanades.

En essència, els missatges HTTP estan compostos de text pla, codificat en ASCII, i normalment comprenen múltiples línies. La versió HTTP tractada en aquest projecte es tracta de la versió HTTP/1.1.

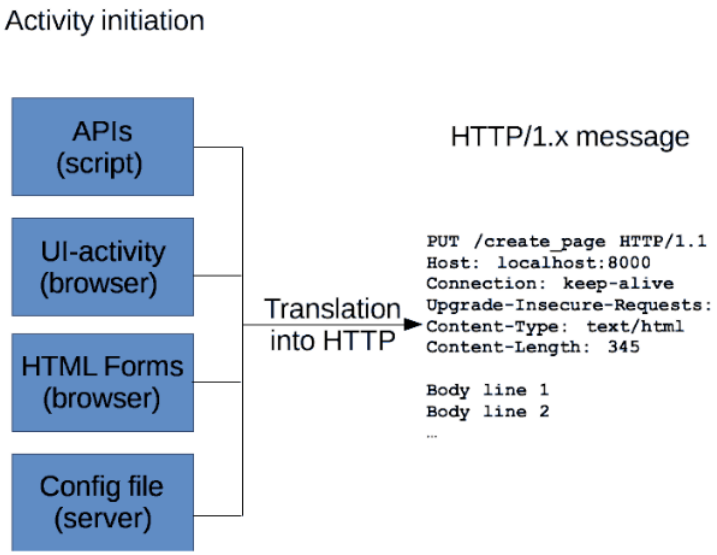

Figura 4: S'observa la forma d'un petició HTTP de versió 1.1 [4]

[4]Font: `https://mdn.mozillademos.org/files/13825/HTTPMsg2.png`

L'estructura d'una petició o resposta HTTP està format pels següents elements de forma endreçada:

1. **Línia d'inici (start-line):** Descriu la petició a ser implementada o el seu estat, sigui d'èxit o fracàs. Aquesta línia d'inici, és sempre una única línia.

2. **Capçaleres HTTP (headers):** Indiquen la petició o descriuen el cos (body) del missatge. Aquest camp/secció és opcional en la petició/resposta HTTP.

3. **Línia buida (empty-line):** Una línia buida o salt de línia, la qual indica que tota la meta-informació ha estat enviat.

4. **Cos del missatge (body):** Es el lloc on ubicar les dades associades amb la petició/resposta, sigui , contingut HTML, arxius o text pla. La presència del cos i la seva mida ha de ser indicat en la línia d'inici.

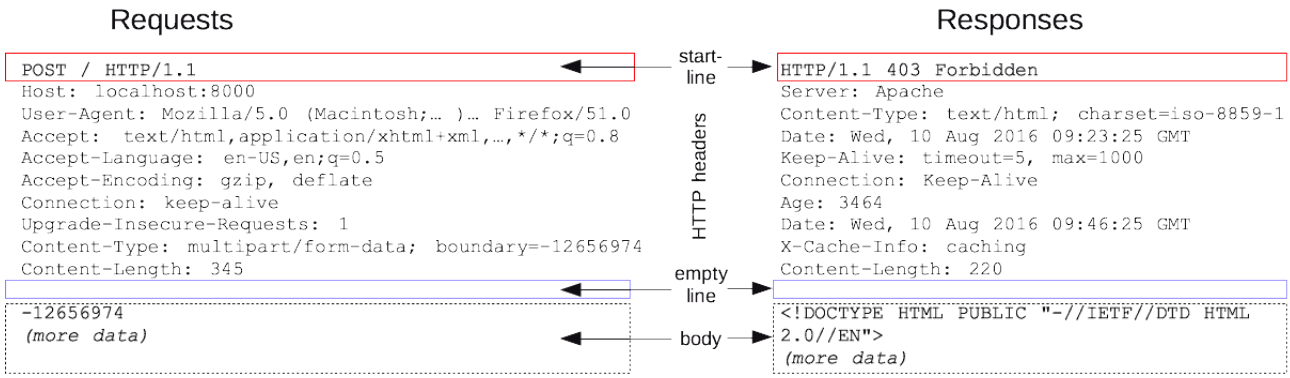

Figura 5: Parts d'una petició i resposta HTTP[5]

[5]Font: `https://mdn.mozillademos.org/files/13827/HTTPMsgStructure2.png`

Tot seguit es mostra un exemple de petició i resposta HTTP:

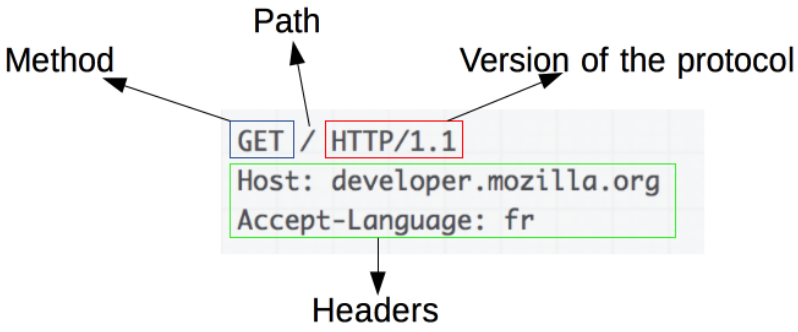

Figura 6: Exemple petició HTTP[6]

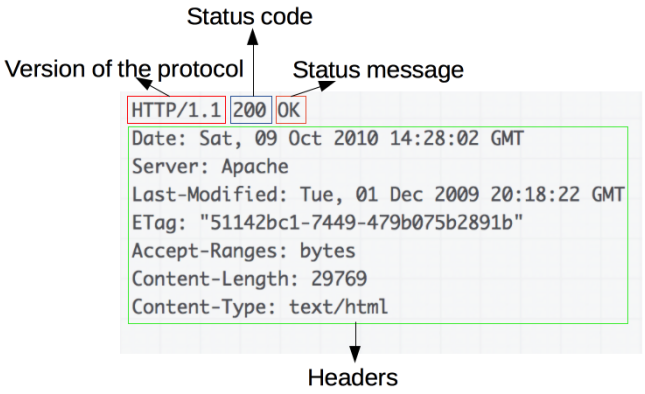

Figura 7: Exemple resposta HTTP[7]

Query String

Igual que el path, la query string és una part de l'Uniform Resource Locator (URL). Aquest comença després del signe d'interrogació i el podem trobar junt amb el path en la petició HTTP. Un exemple de query string seria:

/..Path?MyParam1=ValueParam1 & MyParam2=ValueParam2

[6]Font: `https://mdn.mozillademos.org/files/13687/HTTP_Request.png`
[7]Font: `https://mdn.mozillademos.org/files/13691/HTTP_Response.png`

2.2 Llibreria Servidor

2.2.1 Visió general

Dintre de l'arquitectura client-servidor, aquesta llibreria desenvolupa el role de servidor, per tal de proveir recursos o serveis pels clients demandants estableix un sistema de comunicació basat en la transmissió i recepció de missatges del tipus cadena de caràcters o strings. Així doncs, la llibreria s'anomena **TCPSERVER** i té l'objectiu d'oferir una interfície de funcions simples a l'hora d'establir una comunicació pel protocol TCP, possibilitant un desenvolupament de codi menys complex i directe per la majoria d'aplicacions.

El motiu pel qual, la llibreria TCPSERVER ha estat desenvolupada seguint el protocol TCP és per poder assolir connexions sockets de tipus TCP/IP i connexions per sessions HTTP. És necessari recordar, que les sessions HTTP és la seqüència de peticions i respostes HTTP, on prèviament el client ha d'establir una connexió pel protocol TCP amb el servidor desitjat.

Finalment, cal mencionar, que la llibreria TCPSERVER ha estat desenvolupada fent ús de la llibreria WiFiNINA[comd] d'Arduino, la qual ofereix un ventall més ampli de funcionalitats a nivell de manipulació de caràcters i configuracions de connexions. És per aquest motiu, que la llibreria TCPSERVER està més delimitada en funcionalitats respecte a la llibreria WiFiNINA, ja que, aquesta està orientada a una connexió pe protocol TCP i la transacció de missatges de tipus string, comuna en la majoria d'aplicacions.

2.2.2 Documentació llibreria TCPSERVER

Les següents funcions estan orientades a la connexió del mòdul WiFi amb un punt d'accés WiFi:

- **void connectAccesPoint(char ssid[], char password[])**
  - **Descripció:** Aquesta funció permet la connexió del mòdul ESP32 Wi-Fi de l'Arduino MKR WiFi 1010 a un punt d'accés WiFi encriptat amb WPA/WAP2.
  - **Paràmetres:**
    ssid: El paràmetre és un apuntador (pointer) de string que conté el SSID (nom) del punt d'accés WiFi.

    password: El paràmetre és un apuntador (pointer) de string que conté la contrasenya del punt d'accés WiFi.
  - **Return:** No retorna cap valor.
- **void closeConnecAccesPoint(void)**
  - **Descripció:** Aquesta funció permet el fi de connexió del mòdul ESP32 WiFi de l'Arduino MKR WiFi 1010 amb el punt d'accés WiFi prèviament connectat.
  - **Paràmetres:** No rep cap paràmetre.
  - **Return:** No retorna cap valor.

Les següents funcions estan orientades a la posada en marxa del servidor, la possibilitat de conèixer si s'ha produït una connexió amb algun client i la finalització de la connexió del client amb el servidor:

- **#define PORT**
  - **Descripció:** És un define disponible en *tcpserver.h* per poder configurar el port de connexió del servidor. Per defecte el servidor té assignat el port 80 en aquest define.
- **void initServer(void)**
  - **Descripció:** Aquesta funció posar en marxa el servidor per poder començar a escoltar connexions entrants de clients.
  - **Paràmetres:** No rep cap paràmetre.
  - **Return:** No retorna cap valor.
- **void ipAddressServer(int *ipServer)**
  - **Descripció:** Aquesta funció retorna la IP assignada al servidor pel Punt d'Accés connectat.
  - **Paràmetres:**
    ipServer: És un apuntador d'array d'enters, on la longitud de l'array ha de ser 4 i serà on es desarà l'IP del servidor.
  - **Return:** No retorna cap valor.
- **int portServer(void)**
  - **Descripció:** Aquesta funció retorna el port assignat al servidor.
  - **Paràmetres:** No rep cap paràmetre.
  - **Return:** Retorna un enter, el qual indica el port assignat al servidor.

- **bool clientAvailable(void)**
  - **Descripció:** La funció permet conèixer si hi ha un client connectat amb dades (bytes) per ser llegides.
  - **Paràmetres:** No rep cap paràmetre.
  - **Return:** Retorna true quan hi ha un client connectat amb dades disponibles per ser llegides, en altre cas retorna false.

- **void clientClose(void)**
  - **Descripció:** La funció permet el fi de connexió del client actual connectat al servidor. Cal especificar, que la connexió entre el client i el servidor és persistent, i per tant, es necessari tancar la connexió del client actual abans d'atendre al pròxim.
  - **Paràmetres:** No rep cap paràmetre.
  - **Return:** No retorna cap valor.

Les següents funcions estan orientades a la recepció i transmissió de strings, per tal de suplir els recursos o serveis sol·licitats per part del client al servidor dintre d'una connexió socket de tipus TCP/IP:

- **int readClientMessage(char buffer[], int limitBuffer)**
  - **Descripció:** La funció permet la lectura de tots els bytes disponibles (caràcters de tipus char) d'un client connectat, es a dir, el missatge entrant. Aquest missatge es desarà en un *buffer* passat pel camp de paràmetres.
  - **Paràmetres:**

    buffer: El paràmetre és un apuntador d'array de caràcters, i serà on es desarà del missatge rebut.

    limitBuffer: El paràmetre és de tipus enter i indica el nombre màxim de bytes que poden ser llegits i desats en el *buffer*. Si el nombre de caràcters que conforma el missatge és superior a *limitBuffer*, provocarà l'existència de caràcters pendents per ser llegits.
  - **Return:** Retorna un enter que indica el nombre de bytes (chars) que s'han llegit i desats.
- **int readClientLine(char buffer[], int limitBuffer)**
  - **Descripció:** La funció permet la lectura de tots els bytes disponibles (caràcters de tipus char) d'un client connectat, es a dir, el missatge entrant. En cas de trobar el caràcter associat a salt de línia ('\n'), la lectura terminarà. Aquest missatge es desarà en un *buffer* passat pel camp de paràmetres.
  - **Paràmetres:**

    buffer: El paràmetre és un apuntador d'array de caràcters, i serà on es desarà del missatge rebut.

    limitBuffer: El paràmetre és de tipus enter i indica el nombre màxim de bytes que poden ser llegits i desats en el *buffer*. Si el nombre de caràcters que conforma el missatge és superior a *limitBuffer*, provocarà l'existència de caràcters pendents per ser llegits.
  - **Return:** Retorna un enter que indica el nombre de bytes (chars) que s'han llegit i desats.

- **int sendClientMessage(char buffer[])**
  - **Descripció:** La funció permet l'enviament d'una cadena de caràcters (string), a tots els clients connectats al servidor.
  - **Paràmetres:** El paràmetre és un apuntador d'array de caràcters que conté el missatge a transmetre.
  - **Return:** Retorna el nombre de bytes (chars) que s'han transmès als clients connectats.

Les següents funcions estan orientades a atendre peticions (*requests*) i realitzar respostes HTTP per part del servidor al client, per tal de suplir els recursos o serveis sol·licitats pel client:

- **uint8_t readRequestHTTP(char path[], int limitPath, char *typeRequest)**
  - **Descripció:** La funció permet la captació d'informació enviada per peticions HTTP, mitjançat el mètode GET o POST, realitzades pel client actual connectat. En concret, el *path* de la petició HTTP, els paràmetres o *query string* de la petició HTTP inclòs en el *path* (el fragment precedit pel signe d'interrogació) en cas d'existència, el *body* de la petició en cas d'existència, i finalment el tipus de petició (GET o POST). Cal tenir present, que tant el body com els paràmetres d'una petició HTTP són opcionals.

    Un cop finalitzada la captació d'informació provinent en la petició HTTP, en concret la informació del body o els paràmetres, aquests es desaran en buffers interns de la llibreria amb el fi de poder ser consultades posteriorment per les funcions **getBodyRequestHTTP** i **getParametersRequestHTTP**. Recordar, que per cada nova petició atesa amb la funció **readRequestHTPP**, si la petició comporta nova informació en la secció del body o els paràmetres, els buffers interns es sobreescriuran correspondentment.

- **Paràmetres:**

  path: El paràmetre és un apuntador de tipus string, i serà on es desarà el path de la petició HTTP.

  limitPath: El paràmetre és un enter que limita els caràcters que es desaran en el buffer path.

  typeRequest: El paràmetre és un apuntador de tipus char, en el qual es desarà el char G si la petició ha estat efectuada pel mètode GET, o es desarà el char P si el mètode de la petició ha estat POST.

- **Return:**
  Retorna 0 quan la petició HTTP captada no té informació de paràmetres ni body.
  Retorna 1 quan la petició HTTP captada només conté paràmetres.
  Retorna 2 quan la petició HTTP captada només conté body.
  Retorna 3 quan la petició HTTP captada té body i paràmetres.

- **void getBodyRequestHTTP(char buffer[], int limitBuffer)**

  - **Descripció:** La funció permet el desament de la informació inclosa en el body sobre un *buffer*, passat pel camp de paràmetres de la funció, un cop captada per la funció **readRequestHTTP** sobre la petició HTTP. Aquesta funció ha de ser cridada quan la funció **readRequestHTTP** retorna els valors **2 o 3**.
  - **Paràmetres:**

    buffer: El paràmetre és un apuntador d'array de caràcters, i serà on es desarà del missatge inclòs en el body de la petició HTTP.

    limitBuffer: El paràmetre és de tipus enter i indica el nombre màxim de caràcters que poden ser desats en el *buffer*. Si el nombre de caràcters que conforma el missatge és superior a *limitBuffer*, provocarà l'existència de caràcters pendents per ser llegits en els buffers interns de la llibreria. Cal recordar, que si s'atén una altra petició HTTP amb nova informació en el body, aquesta nova informació sobreescriurà la informació prèvia existent.
  - **Return:** No retorna cap valor.

- **void getParametersRequestHTTP(char buffer[], int limitBuffer)**
  - **Descripció:** La funció permet el desament de la informació corresponent als paràmetres o query string de la petició HTTP sobre un *buffer*, passat pel camp de paràmetres de la funció. Aquesta funció ha de ser cridada quan la funció **readRequestHTTP** retorna els valors **1 o 3**.
  - **Paràmetres:**

    buffer: El paràmetre és un apuntador d'array de caràcters, i serà on es desarà els paràmetres o query string rebuts en la petició HTTP.

    limitBuffer: El paràmetre és de tipus enter i indica el nombre màxim de caràcters que poden ser desats en el *buffer*. Si el nombre de caràcters que conforma el missatge és superior a *limitBuffer*, provocarà l'existència de caràcters pendents per ser llegits en els buffers interns de la llibreria. Cal recordar, que si s'atén una altra petició HTTP amb una nova query string, aquesta nova informació sobreescriurà la informació prèvia existent.
  - **Return:** No retorna cap valor.

- **void sendResponseHTTP(bool statusCode, char body[])**
  - **Descripció:** La funció permet enviar una resposta HTTP per part del servidor al client actual connectat amb informació inclosa en el body. Aquesta funció només permet la configuració de la línia d'inici de la resposta HTTP amb els codis d'estat 200 (indica que la sol·licitud ha tingut exit) i 404 (indica que el servidor no pot trobar el recurs sol·licitat).

    Respecte als *headers* o capçaleres HTTP enviats són els següents:

    Content-type:text/plain
    Content-length: nombre total de caràcters que conforma Body
    Access-Control-Allow-Origin: *

    Els headers *Content-type i Content-length* indiquen característiques sobre la secció del body en la resposta HTTP, i el header *Access-Control-Allow-Origin: **[mdna] indica que la resposta pot ser compartida amb el domini sol·licitant, l'asterisc (*) s'utilitza si es desconeix el domini de procedència dels sol·licitants.

- **Paràmetres:**

  statusCode: El paràmetre és de tipus booleà. Si el valor indicat és *true*, el codi d'estat serà 200 i la línia d'inici serà *HTTP/1.1 200 OK*, en cas d'indicar *false*, el codi d'estat serà 404 i la línia d'inici serà *HTTP/1.1 404 Not Found*.

  body: El paràmetre és un apuntador d'array de caràcters que conté el missatge que s'incorporarà en el body de la resposta HTTP.

- **Return:** No retorna cap valor.

- **void configResponseHTTP(char startLine[], char headers[], char body[])**

  - **Descripció:** La funció permet enviar una resposta HTTP de forma configurable sobre les tres seccions que la conformen, es a dir, passar pel camp de paràmetres la línia d'inici, les capçaleres i el cos de la resposta HTTP.

    Recordar, que cada *header* ha d'acabar amb retorn de carro i salt de línia. Per exemple, els *headers*:
    Content-type:text/plain
    Content-length: X
    Access-Control-Allow-Origin: *

    Hauren de ser passats com **'Content-type:text/plain\r \n Content-length: X\r \n Access-Control-Allow-Origin: * \r \n'**. Tant la línia d'inici com el cos no requereixen l'afegiment del retorn de carro i salt de línia.

  - **Paràmetres:**

    startLine: El paràmetre és un apuntador d'array de caràcters que ha de contenir la línia d'inici en la resposta HTTP. Per exemple, **'HTTP/1.1 200 OK'**.

    headers: El paràmetre és un apuntador d'array de caràcters que ha de contenir els headers de la resposta HTTP.

    body: El paràmetre és un apuntador d'array de caràcters que ha de contenir el body de la resposta HTTP.

  - **Return:** No retorna cap valor.

2.2.3 Exemples d'ús de la llibreria TCPSERVER

- Connectar-se a un Punt d'Accés Wifi i rebre i respondre a un missatge d'un client socket:

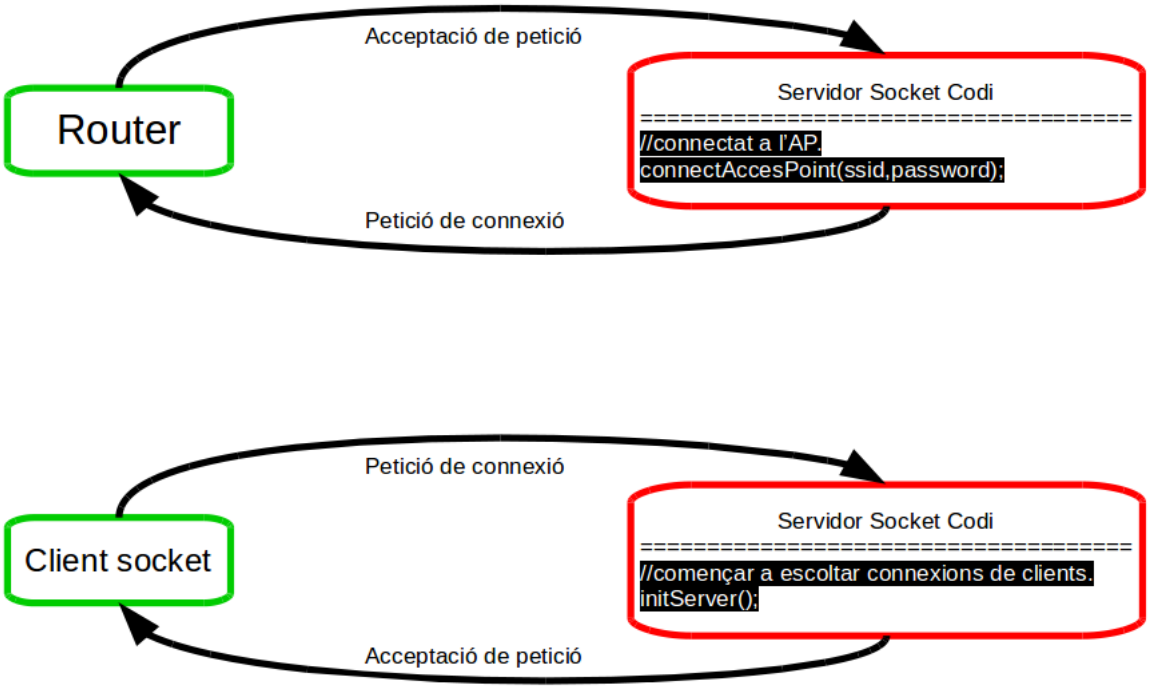

Figura 8: Exemple d'us TCPSERVER-SOCKET[8]

[8]Font: imatge pròpia

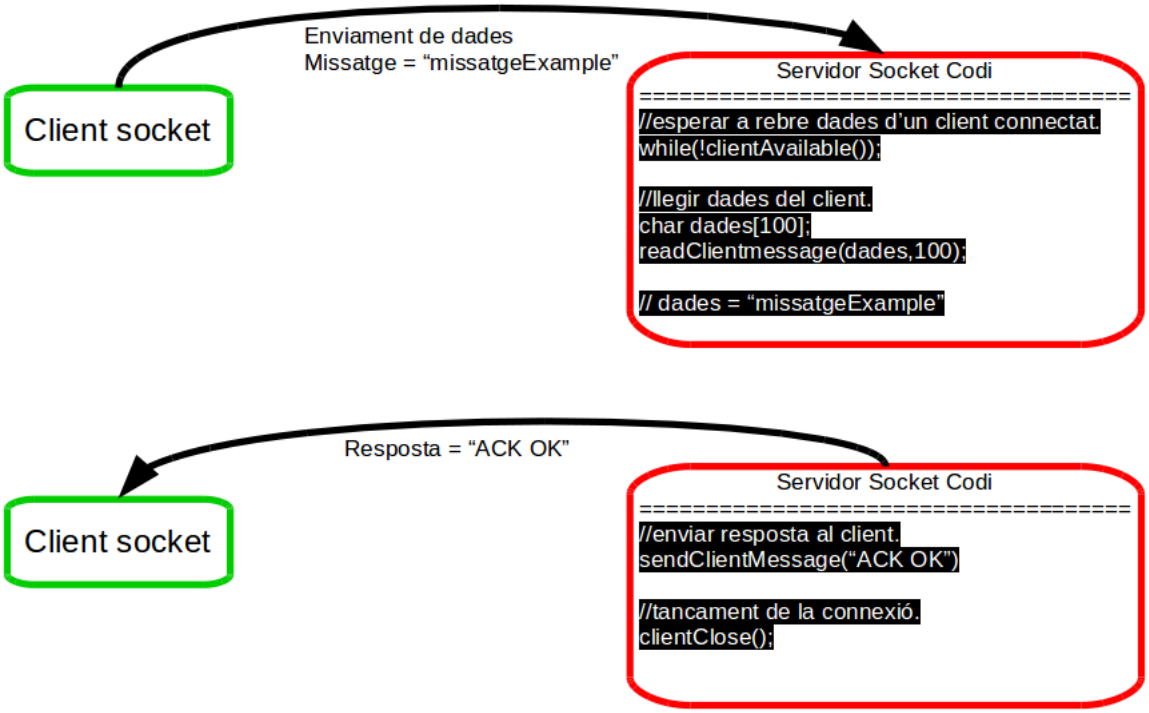

Figura 9: Exemple d'us TCPSERVER-SOCKET[9]

[9]Font: imatge pròpia

- Connectar-se a un Punt d'Accés Wifi i rebre i respondre a un missatge HTTP d'un client, amb informació en el body i en cap query string:

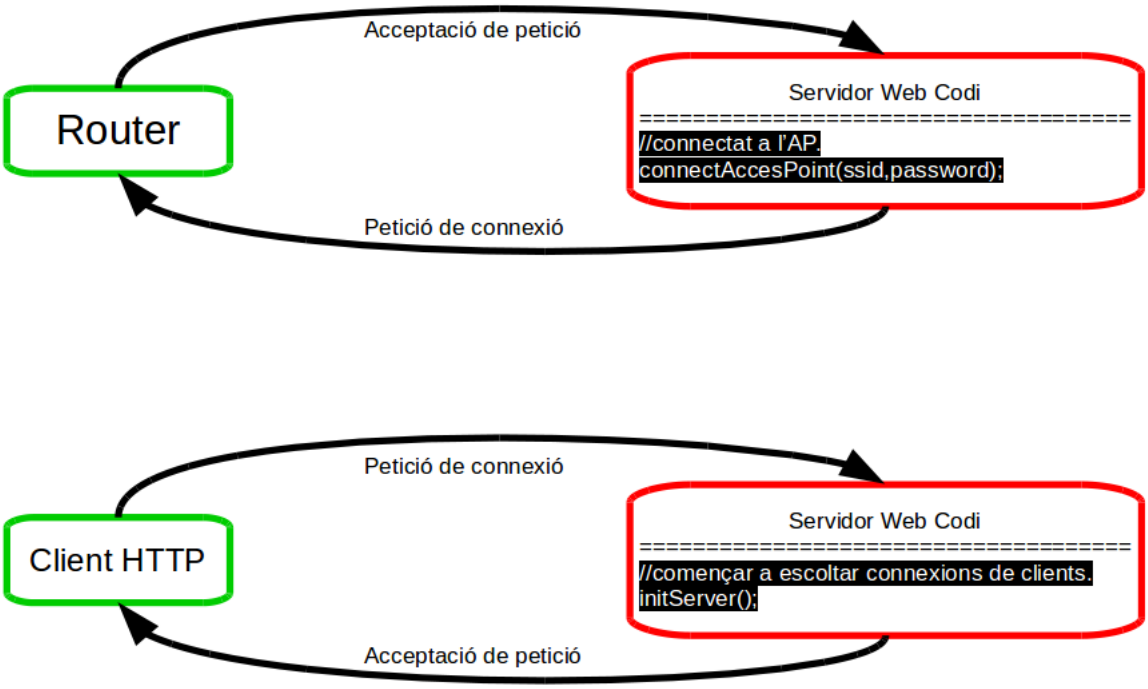

Figura 10: Exemple d'us TCPSERVER-HTTP[10]

[10]Font: imatge pròpia

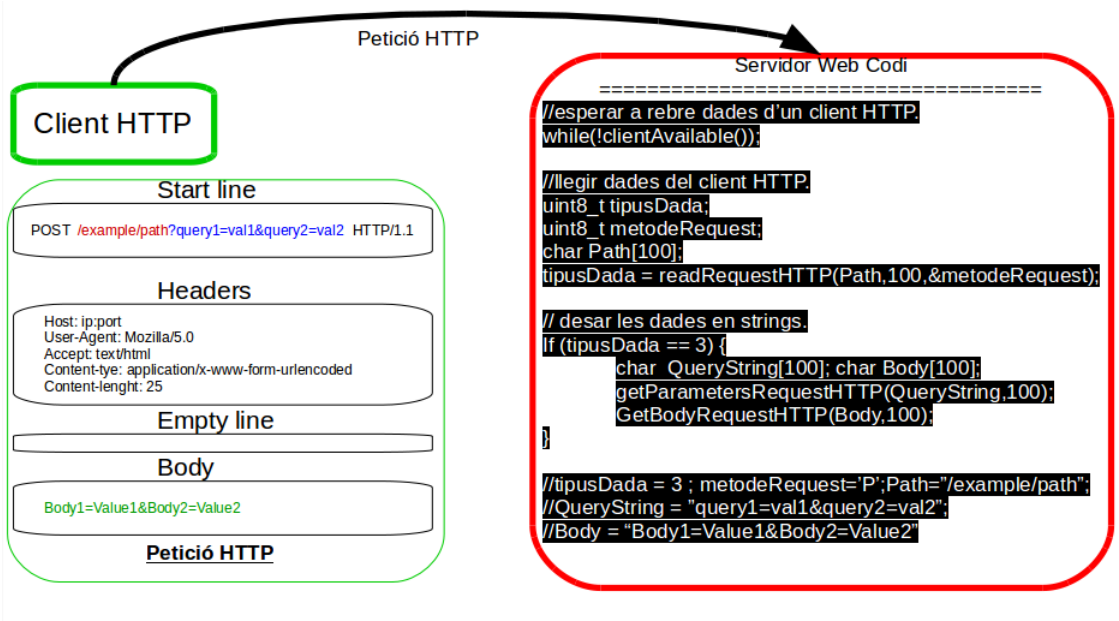

Figura 11: Exemple d'us TCPSERVER-HTTP[11]

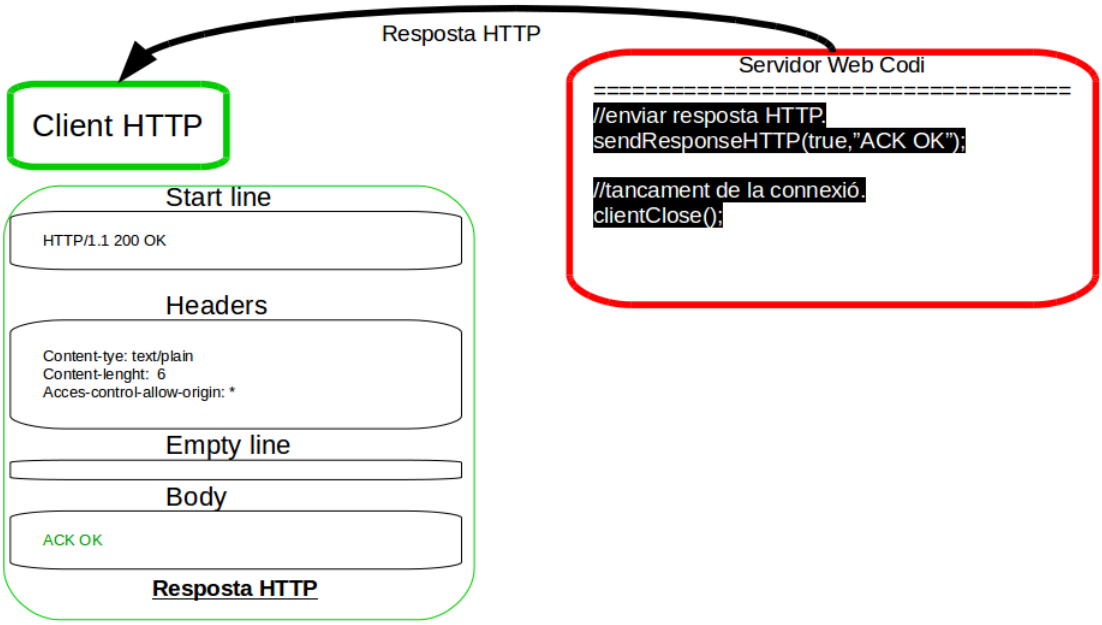

Figura 12: Exemple d'us TCPSERVER-HTTP

[11]Font: imatge pròpia

2.3 Llibreria Client

2.3.1 Visió general

Tal com s'ha explicat prèviament, la llibreria client representa el paper de client demanant serveis o recursos al servidor desitjat. De la mateixa forma que el servidor, aquest estableix un sistema de comunicació basat en la transmissió i recepció de missatges del tipus string, per l'atenció dels recursos sol·licitats.

La llibreria client és anàloga a la llibreria TCPSERVER i s'anomena de forma similar, **TCPCLIENT**, i té els mateixos objectius i condicionaments de com ha estat implementada. En resum, possibilita connexions socket de tipus TCP/IP i per sessions HTTP (pel protocol TCP) d'una forma simplificada.

2.3.2 Documentació llibreria TCPCLIENT

Les següents funcions estan orientades a la connexió del mòdul WiFi amb un punt d'accés WiFi:

- **void connectAccesPoint(char ssid[], char password[])**
  - **Descripció:** Aquesta funció permet la connexió del mòdul ESP32 Wi-Fi de l'Arduino MKR WiFi 1010 a un punt d'accés WiFi encriptat amb WPA/WAP2.
  - **Paràmetres:**
    ssid: El paràmetre és un apuntador de cadena de caràcters que conté el SSID (nom) del punt d'accés WiFi.

    password: El paràmetre és un apuntador de cadena de caràcters que conté la contrasenya del punt d'accés WiFi.
  - **Return:** No retorna cap valor.

- **void ipAddressClient(int *ipClient)**
  - **Descripció:** Aquesta funció retorna la IP assignada al client pel Punt d'Accés connectat.
  - **Paràmetres:**
    ipClient: És un apuntador d'array d'enters, on la longitud de l'array ha de ser 4 i serà on es desarà l'IP del client.
  - **Return:** No retorna cap valor.
- **void closeConnecAccesPoint(void)**
  - **Descripció:** Aquesta funció permet el fi de connexió del mòdul ESP32 WiFi de l'Arduino MKR WiFi 1010 amb el punt d'accés WiFi prèviament connectat.
  - **Paràmetres:** No rep cap paràmetre.
  - **Return:** No retorna cap valor.

Les següents funcions estan orientades a l'establiment de connexió del client a un servidor pel protocol TCP/IP:

- **bool connectTCPClient(byte ip[], uint16_t port)**
  - **Descripció:** La funció permet la connexió a un servidor mitjançat la seva adreça IP i el port de connexió.
  - **Paràmetres:**

    ip: El paràmetre és un apuntador a un array de bytes. L'array de bytes haurà de contenir l'IP del servidor amb el qual es vol establir la connexió. Així mateix, l'array *ip* haurà de ser de mida 4. Per exemple, l'IP 192.168.0.166 hauria de ser creat de la següent forma:

    byte ip[4] =192,168,0,166;

    port: El paràmetre és el port de connexió i és de tipus unsigned de 16 bits, ja que, comprèn tots els ports de connexió existents.
  - **Return:** El valor retornat és un booleà. Retorna true quan la connexió ha estat exitosa i false en cas contrari.
- **void closeConnectTCPClient(void)**
  - **Descripció:** La funció permet la desconnexió del client cap al servidor.
  - **Paràmetres:** No rep cap paràmetre.
  - **Return:** No retorna cap valor.
- **bool connectedTCPClient(void)**
  - **Descripció:** La funció permet conèixer si un client està connectat o no al servidor. Notar, que es considera connectat un client si aquest encara té dades per llegir del servidor, encara que, la connexió s'hi hagi tancat.
  - **Paràmetres:** No rep cap paràmetre.
  - **Return:** Retorna true si el client està connectat i false en cas contrari.

Les següents funcions estan orientades a la recepció i transmissió de strings, per tal de sol·licitar recursos o serveis per part del client al servidor dintre d'una connexió socket de tipus TCP/IP:

- **int availableServerMessage(void)**
  - **Descripció:** La funció permet conèixer nombre total de bytes que el servidor ha enviat (missatge enviat pel servidor), i que encara no s'han llegit.
  - **Paràmetres:** No rep cap paràmetre.
  - **Return:** Retorna el nombre total de bytes/chars que encara no s'han llegit.
- **int readServerMessage(char buffer[], int limitBuffer)**
  - **Descripció:** La funció permet la lectura de tots els bytes disponibles (caràcters de tipus char) que han estat enviats pel servidor actual connectat, es a dir, el missatge entrant. Aquest missatge es desarà en un *buffer* passat pel camp de paràmetres.
  - **Paràmetres:**

    buffer: El paràmetre és un apuntador d'array de caràcters, i serà on es desarà del missatge rebut.

    limitBuffer: El paràmetre és de tipus enter i indica el nombre màxim de bytes que poden ser llegits i desats en el *buffer*. Si el nombre de caràcters que conforma el missatge és superior a *limitBuffer*, provocarà l'existència de caràcters pendents per ser llegits.
  - **Return:** Retorna un enter que indica el nombre de bytes (chars) que s'han llegit i desats.
- **int readServerLine(char buffer[], int limitBuffer)**
  - **Descripció:** La funció permet la lectura de tots els bytes disponibles (caràcters de tipus char) que han estat enviats pel servidor actual connectat, es a dir, el missatge entrant. En cas de trobar el caràcter associat a salt de línia ('\n'), la lectura terminarà. Aquest missatge es desarà en un *buffer* passat pel camp de paràmetres.

  - **Paràmetres:**

    buffer: El paràmetre és un apuntador d'array de caràcters, i serà on es desarà del missatge rebut.

    limitBuffer: El paràmetre és de tipus enter i indica el nombre màxim de bytes que poden ser llegits i desats en el *buffer*. Si el nombre de caràcters que conforma el missatge és superior a *limitBuffer*, provocarà l'existència de caràcters pendents per ser llegits.
  - **Return:** Retorna un enter que indica el nombre de bytes (chars) que s'han llegit i desats.

- **int sendServerMessage(char request[])**
  - **Descripció:** La funció permet l'enviament d'una cadena de caràcters al servidor actualment connectat.
  - **Paràmetres:** El paràmetre és un apuntador d'array de caràcters que conté el missatge a transmetre.
  - **Return:** Retorna el nombre de bytes (chars) que s'han transmès al servidor.

Les següents funcions estan orientades a realitzar peticions i obtenir respostes HTTP per part del client al servidor, per tal d'obtenir respostes els recursos o serveis sol·licitats pel client:

- **void requestServerHTTP(char type,char path[], char message[])**
  - **Descripció:** La funció permet realitzar peticions HTTP pels mètodes GET o POST en una direcció (path), indicat en el camp de paràmetres de la funció, i sobre el servidor actualment connectat.

    Adicionalment, també és pot afegir una *query string* en la petició HTTP o un *body* codificat segons el header *application/x-www-form-urlencoded*, es a dir, en tuples clau-valor separats per & , amb un = entre la clau i el valor. La codificació pel header *application/x-www-form-urlencoded* està orientat a informació provinent de formularis HTML. Aquesta informació addicional ha de ser passada pel paràmetre **message**. En cas de realitzar la petició HTTP sense una query string ni body el pàrametre **message** ha de ser igual a una cadena de caracters buit ("").
  - **Paràmetres:**

    type: És de tipus char i indica el mètode de la petició HTTP. Per indicar el mètode GET el seu valor ha de ser el caràcter 'G'. Per indicar el mètode POST el seu valor ha de ser el caràcter 'P'.

    path: És de tipus apuntador de string i conté el path al qual va dirigit la petició HTTP.

    message: És de tipus apuntador de string i conté la query string o body que s'hi ha d'efegir a la petició HTTP. Aquest paràmetre és opcional, en cas de no voler afegir ni query string ni body, el seu valor ha de ser un string buit ("").
  - **Return:** No retorna res.

- **void configrequestServerHTTP(char startLine[], char headers[], char body[])**
  - **Descripció:** La funció permet enviar una petició HTTP de forma configurable sobre les tres seccions que la conformen, es a dir, passar pel camp de paràmetres la línia d'inici, les capçaleres i el cos de la petició HTTP.

    Recordar, que cada *header* ha d'acabar amb retorn de carro i salt de línia. Per exemple, els *headers*:
    Content-type:text/plain
    Content-length: X
    Access-Control-Allow-Origin: *

    Hauran de ser passats com **'Content-type:text/plain\r \n Content-length: X\r \n Access-Control-Allow-Origin: * \r \n'**. Tant la línia d'inici com el cos no requereixen l'afegiment del retorn de carro i salt de línia.
  - **Paràmetres:**

    startLine: El paràmetre és un apuntador d'array de caràcters que ha de contenir la línia d'inici en la petició HTTP. Per exemple, **'POST /PATH... HTTP/1.1'**.
    headers: El paràmetre és un apuntador d'array de caràcters que ha de contenir els headers de la petició HTTP.
    body: El paràmetre és un apuntador d'array de caràcters que ha de contenir el body de la petició HTTP.
  - **Return:** No retorna cap valor.

- **void responseServerHTTP(char response[], int lenResponse)**
  - **Descripció:** La funció permet la lectura del body quan una resposta es enviada pel servidor. En concret llegeix i desa tot els caràcters que conforma missatge integrat en el body de la petició HTTP.
  - **Paràmetres:**

    response: El paràmetre és un apuntador d'array de caràcters, i serà on es desarà el missatge contingut en el body de la resposta HTTP.

    lenResponse: El paràmetre és de tipus enter i indica el nombre màxim de bytes que poden ser llegits i desats en el array indicat per *response*.
  - **Return:** No retorna cap valor.

2.3.3 Exemples d'ús de la llibreria TCPCLIENT

- Connectar-se a un Punt d'Accés Wifi i enviar i rebre un missatge d'un servidor socket:

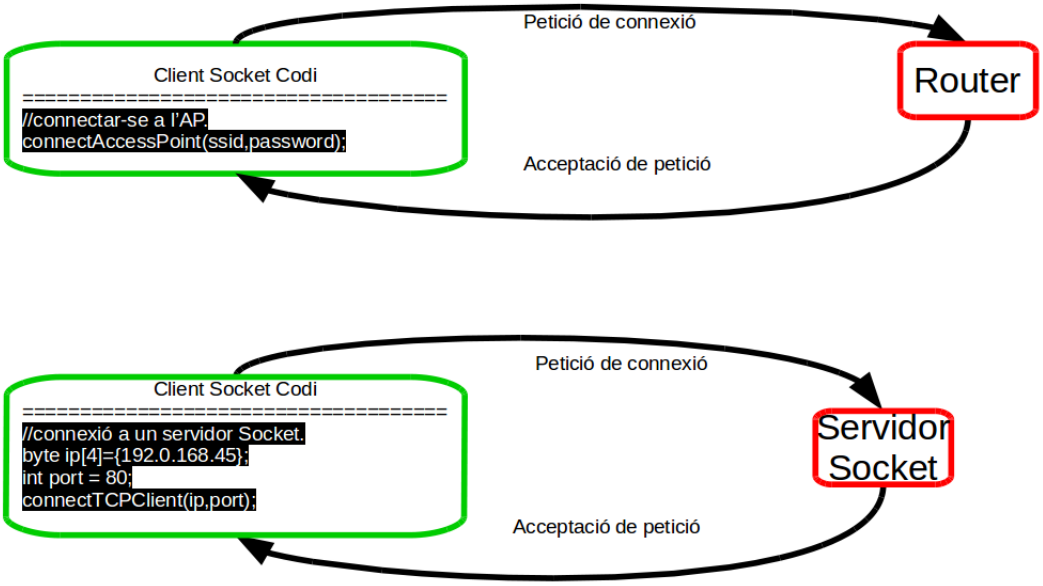

Figura 13: Exemple d'us TCPCLIENT-SOCKET[12]

[12]Font: imatge pròpia

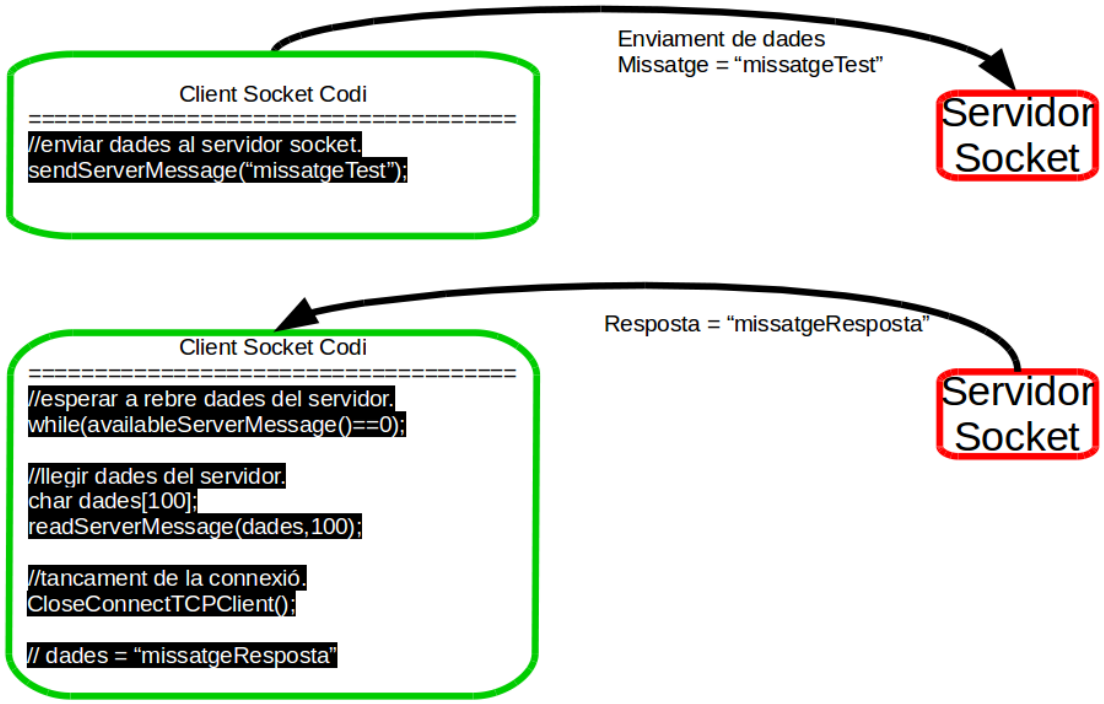

Figura 14: Exemple d'us TCPCLIENT-SOCKET[13]

[13]Font: imatge pròpia

- Connectar-se a un Punt d'Accés Wifi i rebre i respondre a un missatge HTTP d'un client, amb informació en el body i en cap query string:

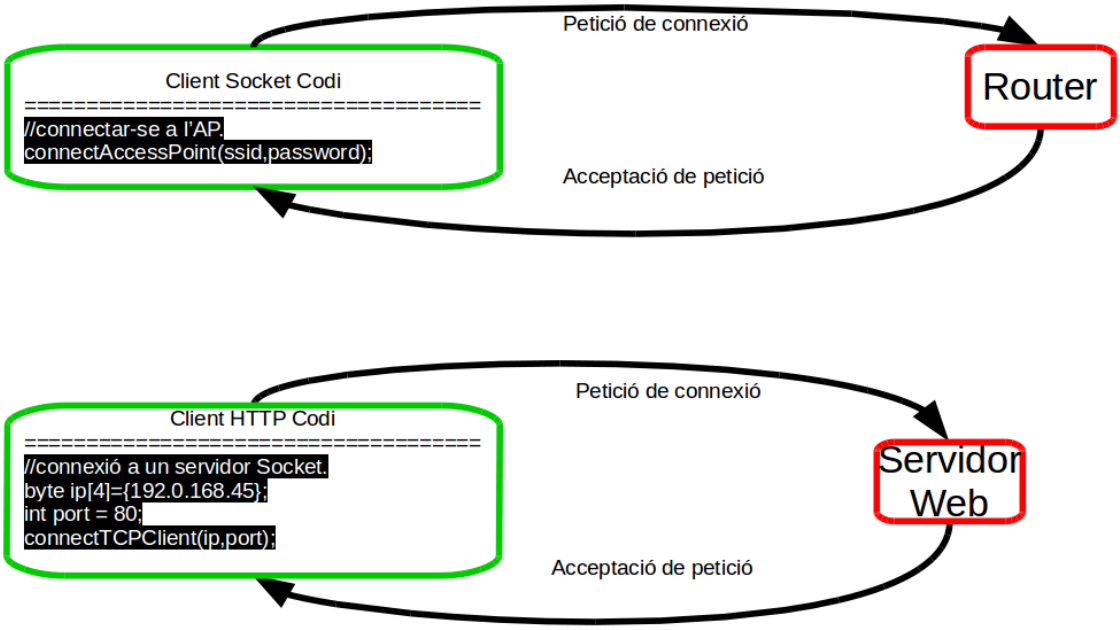

Figura 15: Exemple d'us TCPCLIENT-HTTP[14]

[14]Font: imatge pròpia

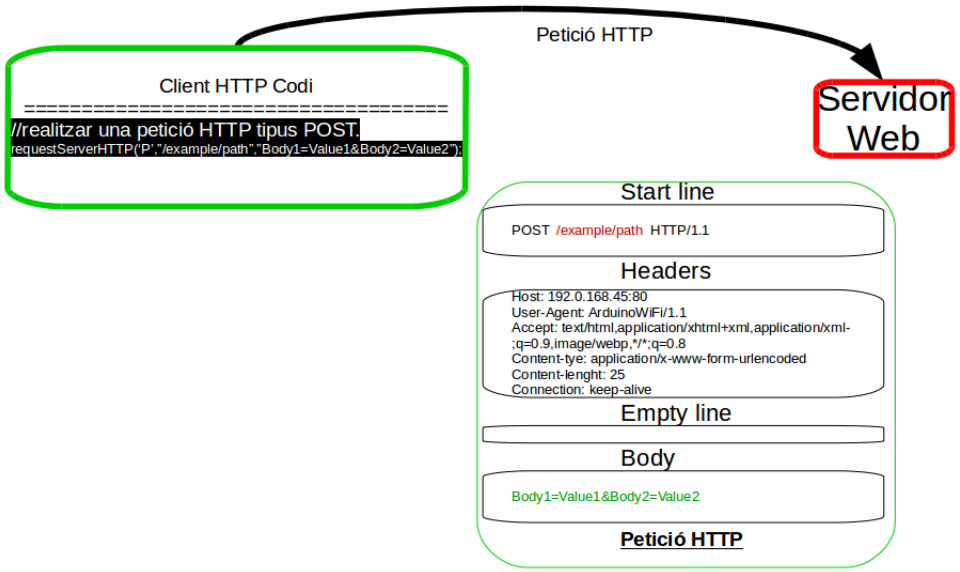

Figura 16: Exemple d'us TCPCLIENT-HTTP[15]

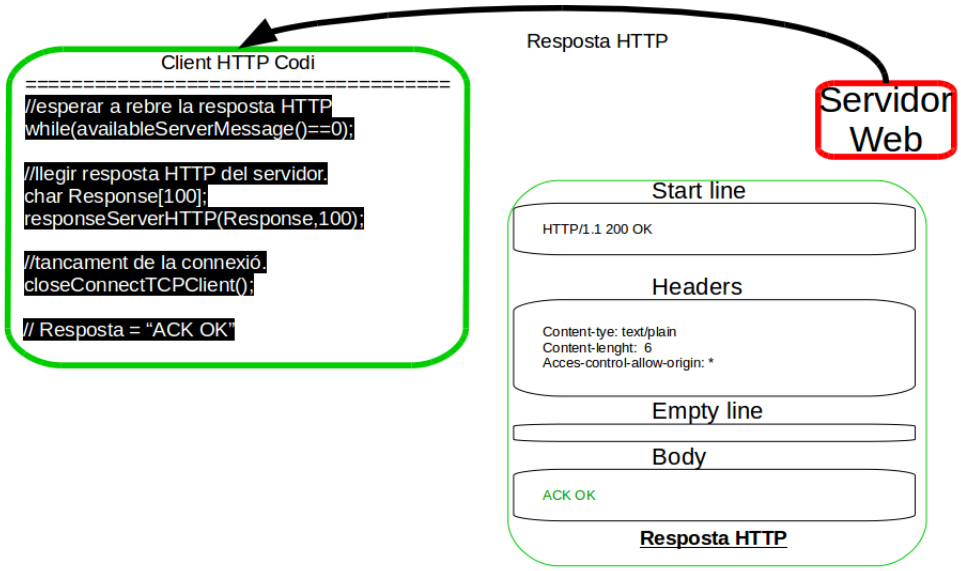

Figura 17: Exemple d'us TCPCLIENT-HTTP

[15]Font: imatge pròpia

3 Mòduls del Robot Car Kit

En aquesta secció es tractarà el desenvolupament dels mòduls que conformen el Robot Car Kit, es a dir, el mòdul de les rodes motrius, el mòdul d'ultrasò dirigible i el de seguiment de línia. Principalment, es desenvoluparen llibreries per cada mòdul fent ús de l'IDE d'Arduino sobre el microcontrolador MKR WiFi 1010, amb el fi de facilitar el seu control.

3.1 Mòdul de les Rodes Motrius

Aquest mòdul està destinat al control de les rodes motrius del Robot Car Kit mitjançat un conjunt de components electrònics. Principalment, la llibreria corresponent aquest mòdul s'anomena **MOTOR** i està enfocada al control del sentit de gir i la velocitat en RPM sobre les rodes motrius a partir del llaç de control electrònic disponible, amb el fi d'implementar funcionalitats d'interès envers la mobilitat del Robot Car Kit. En les següents seccions s'explicaren el llaç de control i les estratègies seguides per la implementació de la llibreria MOTOR.

Els components electrònics implicats sobre el llaç o sistema de control són:

1. Dos motors simples de corrent continua (DC) amb caixa d'engranatges incorporats, ubicats en la part posterior del Robot Car Kit.

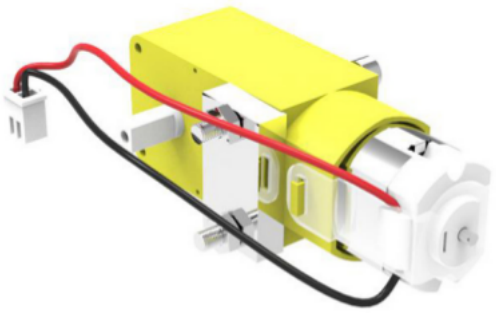

Figura 18: DC Motor[16]

[16]Font: documentació del kit elegoo Robot Car Kit

2. Dos micro motors de corrent continua (DC) amb caixa d'engranatges i enconders de quadratura incorporats a l'eix del motor, ubicats en la part davantera del Robot Car Kit, en concret el *Micro DC Motor with Enconder-SJ01 SKU FIT0450*[come]. Cal especificar, que el Robot Car kit inicialment tenia dos motors simples de corrent continua en la seva part frontal, aquests s'hi han substituït pels dos micro motors amb encoders amb el fi de tancar el llaç de control sobre les rodes motrius.

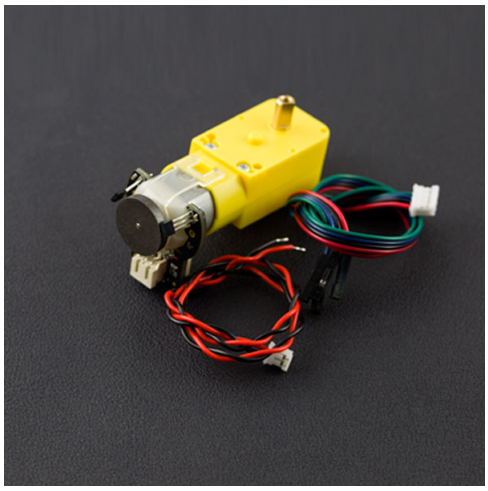

Figura 19: Micro DC Motor with Enconder-SJ01[17]

3. Un mòdul controlador de motors, en concret el *driver L298N*.

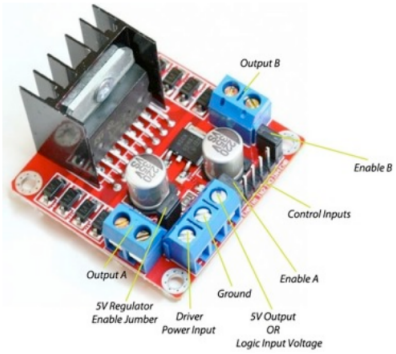

Figura 20: Driver L298N[18]

[17]Font: `https://wiki.dfrobot.com/Micro_DC_Motor_with_Encoder-SJ01_SKU__FIT0450`
[18]Font: documentació del kit elegoo Robot Car Kit

3.1.1 Funcionament de l'electrònica implicada

Essencialment, els elements electrònics que permeten el control sobre les rodes són el *driver L298N* i *Micro DC Motor with Enconder-SJ01 SKU FIT0450*, els quals conformen el sistema o llaç de control de les rodes, a banda del microcontrolador Arduino MKR WiFi 1010.

Tot seguit, s'explicarà en detall el funcionament del driver L298N, el qual permet el control de quatre motors de corrent contínua a partir de dos canals. Cada canal de control admet fins a un nombre màxim de dos DC motors, on la configuració programada per a cada canal s'aplica de forma paral·lela als dos motors connectats. Posant el cas del sentit de gir de rotació dels motors, els dos motors connectats al mateix canal giraran en el mateix sentit i direcció de forma simultània. Aquest fet, comporta una delimitació envers un control independent dels quatre motors.

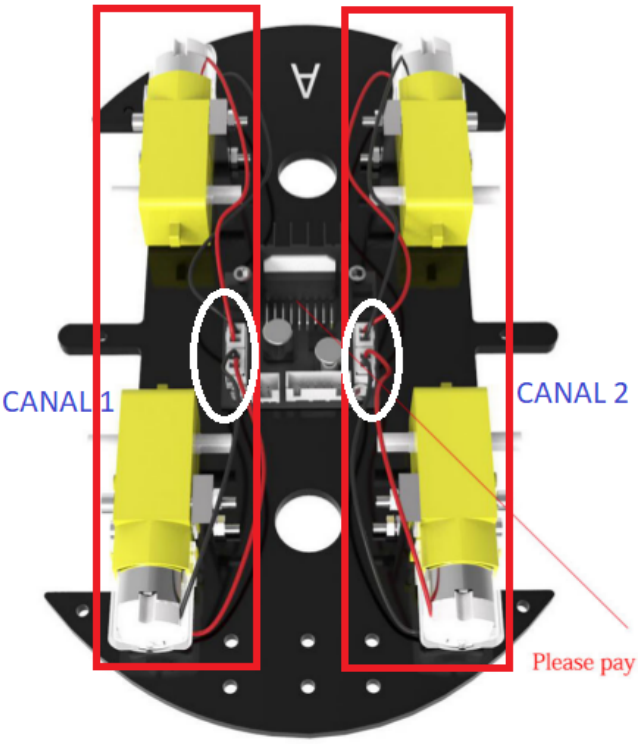

Figura 21: Driver L298N connectat als motors a partir dels dos canals disponibles del driver[20]

[20]Font: documentació del kit elegoo Robot Car Kit

Les funcionalitats de control que permet el driver per cada canal són la velocitat en revolucions per minut (RPM) i el sentit de gir de rotació. La velocitat RPM és controlada mitjançats senyals *Pulse Width Modulation (PWM)*[Hir] sobre els pins d'entrada ENA i ENB, segons el canal, en el driver L298N. Els senyals PWM es generen des de l'Arduino MKR WiFi 1010 amb la funció *analogWrite*[coma] d'Arduino. El sentit de gir de rotació es controlat a partir del subministrament de senyals digitals (alst o baixos) en els pins d'entrada EN1 i EN2 pel canal corresponent al pin ENA, i als pins EN3 i EN4 pel canal corresponent al pin ENB del driver L298N. Aquests pins d'entrada en el driver L298N s'utilitzaran per a implementar la llibreria MOTOR.

Els pins ENA, ENB, EN1, EN2, EN3 i EN4 seran pins de sortida en L'Arduino MKR WiFi 1010.

ENA	IN1	IN2	DC MOTOR STATUS
O	X	X	STOP
1	0	O	BRAKING
1	0	1	FORWARD
1	1	0	BACKWARD
1	1	1	BARKING

Figura 22: Taula de configuració dels senyals EN1 i EN2, aquesta configuració és anàloga per EN3 i EN4[22]

Tal com s'ha esmentat prèviament, el driver L298N permet el control de les velocitats en RPM sobre els motors, a partir de senyals PWM, però s'observa una mancança sobre el driver L298N per conèixer els RPMs dels motors un cop aplicat els senyals PWM en els pins ENA i ENB. Es a dir, des del driver L298N s'apliquen senyals PWM que fan variar les velocitats RPM dels motors, però es desconeix quines velocitats assoleixen aquests pels senyals PWM subministrats. Arran d'aquest fet, s'introdueix els encoders de quadratura (encoder SJ01) amb el fi de tancar el llaç de control i poder conèixer quines velocitats RPM assoleixen els motors.

A continuació, per una millor comprensió s'explicarà el funcionament dels encoders de quadratura implicats i seguidament les característiques tècniques de l'encoder SJ01 acoplat sobre l'eix del motor que faciliten l'obtenció de les velocitats RPM i el posicionament sobre les rodes motrius.

[22]Font: documentació del kit elegoo Robot Car Kit

L'encoder SJ01, de quadratura, té dos sensors d'efecte Hall, els quals generen polsos digitals gràcies a un disc magnètic giratori, en el nostre cas muntat en l'eix del motor[rob].

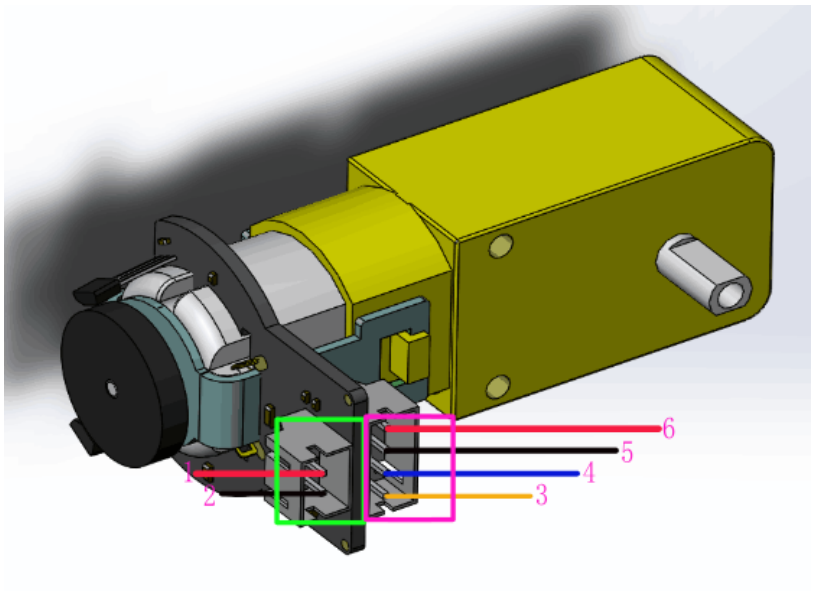

Figura 23: Encoder SJ01 amb dos sensors Hall[23]

La seqüència de polsos digitals obtinguts, normalment provinents per dos canals (A i B), ens permet conèixer la direcció i el nombre de desplaçaments (voltes) que s'han produït en l'encoder.

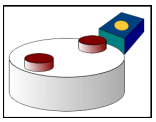

Figura 24: Sensors Hall Actiu

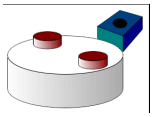

Figura 25: Sensors Hall No Actiu

[23]Font: `https://raw.githubusercontent.com/DFRobot/DFRobotMediaWikiImage/master/Image/FIT0450.png`

Aquesta seqüència està en Codi Gray de dos bits. Principalment, aquesta codificació ens permet conèixer la direcció de gir que pren l'encoder, a partir del desfasament (de 90 graus, d'aquí el nom encoder de quadratura) existent en les dues seqüències provinents de l'encoder i la seva interpretació fent ús d'una matriu que incorpora la codificació Gray. En les següents imatges s'observa el desfasament existent i la matriu Gray.

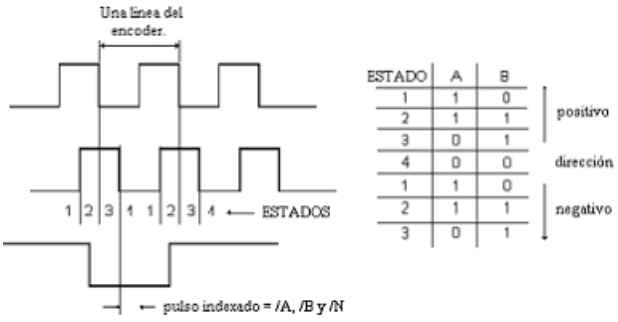

Figura 26: Seqüència senyals desfaçats canals A i B[24]

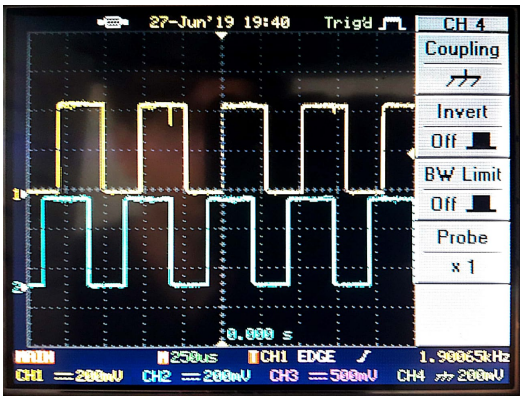

Figura 27: Polsos dels canals A i B del Encoder SJ01[25]

[24]Font: `http://androminarobot.blogspot.com/2016/08/encoder-de-cuadratura-y-arduino.html`

[25]Font: imatge pròpia

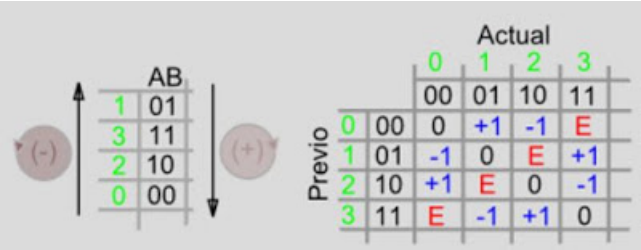

Figura 28: Matriu Gray pels canals A i B[26]

Prenent, com a bit més significatiu el senyal digital provinent del canal A i com a bit menys significatiu el senyal del canal B, es pot observar en la taula de veritat de la **figura 26** el sentit de gir del motor quan es compara la posició actual amb la posició anterior. De la mateixa forma, si comparem la posició actual amb la posició prèvia com a coordenades en la matriu Gray podem conèixer si el sentit de gir és antihorari en el cas d'obtenir un *-1*, si el sentit de gir és el de l'horari en el cas d'obtenir un *+1*, si no hi hagut modificació del sentit de gir en el cas d'un *0* i si s'hi ha produït un error en l'encoder en el cas d'una *E*. Tot així, de cara a poder conèixer el sentit de gir dels motors/rodes es farà a partir del driver L298N, el qual facilita el seu control mitjançats els pins EN1, EN2, EN3 i EN4, evitant el mostreig dels senyals provinents de l'encoder en el microcontrolador Arduino MKR WiFi 1010 amb el fi d'aplicar la codificació Gray.

[26]Font: `http://androminarobot.blogspot.com/2016/08/encoder-de-cuadratura-y-arduino.html`

Respecte a les característiques tècniques del *Micro DC Motor with Enconder-SJ01 SKU FIT0450* que faciliten l'obtenció de les velocitats RPM i el posicionament sobre les rodes motrius trobem la seva relació de transmissió i els dos canals provinents dels sensors de l'encoder per la captació de les revolucions de l'eix del motor. En primer lloc, la seva relació de transmissió/reducció és i=120:1, es a dir, es requereix 120 revolucions/voltes de l'eix del motor (on es trobar ubicat l'encoder) per realitzar 1 revolució/volta sobre l'eix de la roda. En segon lloc, l'encoder integrat a l'eix del motor proporciona dos canals, A i B, per la captació dels senyals digitals (quadrats o polsos) provinents dels sensors de l'encoder. Cada canal té una resolució de 8 polsos/cicles per cada revolució de l'eix del motor, el qual fa una resolució de 16 polsos per cada revolució de l'eix del motor.

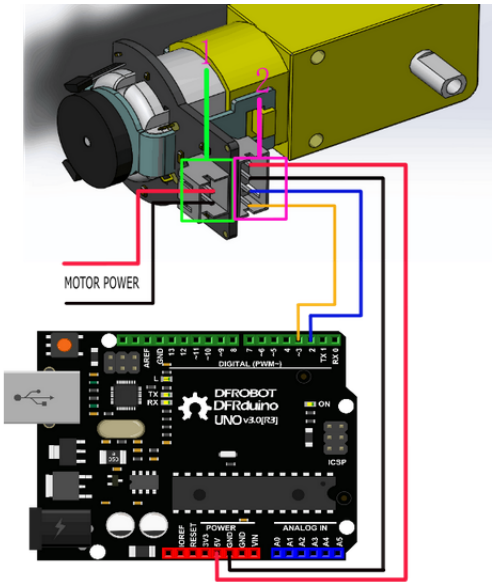

Figura 29: Connexió motor-enconder amb l'Arduino, en el nostre cas seria el MKR WiFi 1010[28]

[28]Font: `https://raw.githubusercontent.com/DFRobot/DFRobotMediaWikiImage/master/Image/FIT0450+UNO.png`

A partir dels polsos captats pels canals A i B de l'encoder i l'ús d'interrupcions al microcontrolador Arduino MKR WiFi 1010, es torna factible l'obtenció de la velocitat RPM assolida en l'eix de la roda i també el posicionament d'aquesta. Notar, que els senyals provinents dels canals A o B de l'encoder seran entrades en l'Arduino MKR WiFi 1010.

3.1.2 Obtenció de la velocitat RPM de les rodes

L'obtenció de la velocitat RPM de les rodes motrius es basa en el mostreig dels polsos provinents d'un canal de l'encoder. L'estratègia implementada en la llibreria MOTOR, es tracta en la utilització d'interrupcions amb l'Arduino MKR WiFi 1010, a fi d'implementar un comptador de polsos durant un temps de mostreig determinat.

D'aquesta forma, coneixent les següents dades:

1. Relació de transmissió/reducció del motor de **i=120:1**, es a dir, 120 revolucions o voltes de l'eix del motor equival a 1 volta de l'eix de la roda.

2. Resolució de cada canal (A i B) de l'encoder, es a dir, 8 polsos per 1 revolució o volta de l'eix del motor.

3. Nombre de polsos mostrejats per un dels canals de l'encoder (A o B) durant un temps determinat (temps de mostreig). Especificar, que només es farà ús d'un canal de l'encoder, a fi d'optimitzar recursos sobre l'Arduino MKR WiFi 1010. El fet d'utilitzar els dos canals per cada encoder, tenint dos motors amb encoders amb resolució de 8 polsos/voltaEix i relació de reducció de 120 voltesEix/voltaRoda, resultaria en l'existència de 3840 polsos per cada volta de roda, es a dir, $3840 = 2 \cdot 2 \cdot 8 \cdot 120$. Amb el fi de mostrejar els polsos vinents dels dos encoders amb interrupcions, provocaria 3840 interrupcions per volta del motor.

Amb les dades anteriors, es poden aplicar les següents fórmules pel càlcul de les revolucions per minut assolides per les rodes:

- Càlcul de les voltes assolides per les rodes durant un temps de mostreig:

  $$VoltesRodes = PolsosMostrejats \cdot \frac{1VoltaEixMotor}{8Polsos} \cdot \frac{1VoltaEixRoda}{120VoltaEixMotor}$$

- Si el temps de mostreig és en segons, haurem de passar-ho a minuts:

  $$TempsMostreig \cdot \frac{1minut}{60Segons} = \frac{TempsMostreig}{60} \text{ minuts}$$

- Finalment obtenim la velocitat en RPM amb el nombre de polsos mostrejats i el temps de mostreig de la següent forma:

  $$RodaRPM = PolsosMostrejats \cdot \frac{1VoltaEixMotor}{8Polsos} \cdot \frac{1VoltaEixRoda}{120VoltaEixMotor} \cdot \frac{60}{TempsMostreigSegons}$$

Recordar, que els senyals PWM subministrats sobre els pins ENA i ENB del driver L298N, s'utilitzen per poder generà diferents velocitats RPM en els motors connectats i provenen de l'Arduino MKR WiFi 1010 mitjançat la funció *analogWrite* d'Arduino. On dos dels quatre motors del Robot Car Kit, incorporen l'encoder SJ01, amb el qual es podrà calcular quines velocitats RPM assoleixen a partir dels senyals PWM subministrats.

Alternativament, coneixent la freqüència del senyal digital provinent d'un canal de l'encoder SJ01 i donat un temps de mostreig, es pot calcular la velocitat en RPM de la roda acoblada al motor. Principalment, amb la freqüència del senyal es possible conèixer el nombre de polsos que seran mostrejats i d'aquesta forma aplicar els càlculs anteriors.

- Càlcul del nombre de polsos provinents d'un canal de l'encoder donat un temps de mostreig i la freqüència dels senyals dels polsos:

  $$PolsosMostrejats = \frac{TempsMostreig}{FrequenciaPolsos^{-1}}$$

  On la $frequencia^{-1}$ és el període del senyal, tenint en compte que tant el temps de mostreig i el període tenen la mateixa unitat de temps.

- Tenint en compte que el període del senyal i el temps de mostreig estiguin en segons, obtenim la velocitat com:

$$RodaRPM = \frac{TempsMotreig}{Frequencia^{-1}} \cdot \frac{1}{8} \cdot \frac{1}{120} \cdot \frac{60}{TempsMostreig}$$

$$RodaRPM = Frequencia \cdot \frac{1}{8} \cdot \frac{1}{120} \cdot 60$$

D'aquesta forma es pot assignar a cada senyal PWM una velocitat RPM de la roda, només amb la freqüència que el senyal PWM provoca en un canal de l'encoder.

3.1.3 Assignació senyal PWM a RPM

Com s'ha explicat prèviament, els senyals PWM subministrats sobre els pins ENA i ENB del driver L298N, s'utilitzen per poder generà diferents velocitats RPM en els motors connectats i provenen de l'Arduino MKR WiFi 1010 mitjançat la funció *analogWrite* d'Arduino. On dos dels quatre motors del Robot Car Kit, incorporen l'encoder SJ01, amb el qual es podrà calcular quines velocitats RPM assoleixen a partir dels senyals PWM subministrats. També cal recordar, que es pot assignar a cada senyal PWM una velocitat RPM de la roda, només amb la freqüència que el senyal PWM provoca en un canal de l'encoder.

Així mateix, un dels objectius de la llibreria MOTOR es poder treballar directament amb valors en unitats RPM en el control de les velocitats de les rodes motrius. A causa d'aquest fet, cal conèixer quin rang de senyals PWM ofereix la funció **analogWrite**[coma] d'Arduino, i conseqüentment calcular quines velocitats RPM s'obtenen.

En resum, la funció **analogWrite** escriu un valor (senyals PWM) sobre un pin de l'Arduino MKR WiFi 1010. Aquest senyal PWM, té una freqüència de treball de 732 Hz.

BOARD	PWM PINS	PWM FREQUENCY
Uno, Nano, Mini	3, 5, 6, 9, 10, 11	490 Hz (pins 5 and 6: 980 Hz)
Mega	2 - 13, 44 - 46	490 Hz (pins 4 and 13: 980 Hz)
Leonardo, Micro, Yún	3, 5, 6, 9, 10, 11, 13	490 Hz (pins 3 and 11: 980 Hz)
Uno WiFi Rev.2	3, 5, 6, 9, 10	976 Hz
MKR boards *	0 - 8, 10, A3 (18), A4 (19)	732 Hz
MKR1000 WiFi *	0 - 8, 10, 11, A3 (18), A4 (19)	732 Hz
Zero *	3 - 13, A0 (14), A1 (15)	732 Hz
Due **	2-13	1000 Hz
101	3, 5, 6, 9	pins 3 and 9: 490 Hz, pins 5 and 6: 980 Hz

Figura 30: Freqüències AnalogWrite[29]

[29]Font: `https://www.arduino.cc/reference/en/language/functions/analog-io/analogwrite/`

La sintàxis de la funció **analogWrite** i el rang de valors permès es mostra a continuació:

Syntax

```
analogWrite(pin, value)
```

Parameters

pin: the Arduino pin to write to. Allowed data types: **int**.
value: the duty cycle: between 0 (always off) and 255 (always on). Allowed data types: **int**.

Returns

Nothing

Figura 31: Syntax AnalogWrite[30]

Tal com s'observa en la figura anterior, el rang de valors permès pels senyals PWM[Hir] generats per la funció *analogWrite* és de 0 a 255, i representa el temps de cicle de treball de forma activa. En la següent figura s'observa de forma gràfica aquest fet:

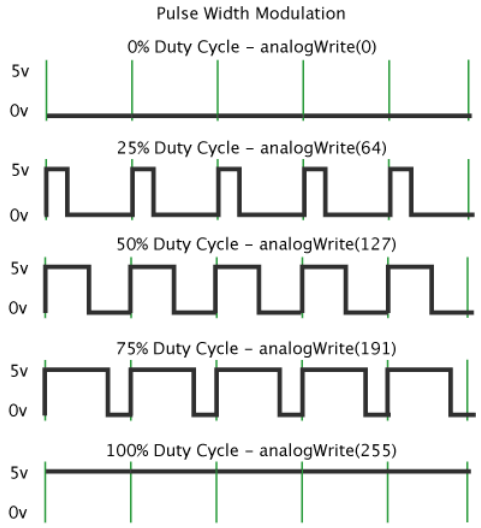

Figura 32: Cicles de treball en els senyals PWM generats per AnalogWrite amb diferents valors[32]

[30]Font: https://www.arduino.cc/reference/en/language/functions/analog-io/analogwrite/

Pel que fa a l'assignació de cada senyal PWM a un valor RPM, es segueix la teoria explicada en la secció 3.1.2, on a partir de l'obtenció dels polsos del senyal d'un canal provinent de l'encoder es calcula el valor RPM associat al senyal PWM subministrat. En la següent figura, s'observa una taula amb les velocitats RPM calculades per la majoria de senyals PWM disponibles de la funció *analogWrite* sobre les rodes motrius.

PWM value	RPM value	PWM value	RPM value
0	0	130	114
5	0	135	115
10	0	140	120
15	0	145	121
20	0	150	125
25	0	155	126
30	0	160	130
35	0	165	130
40	0	170	134
45	0	175	134
50	0	180	137
55	0	185	137
60	27	190	140
65	41	195	140
70	48	200	141
75	56	205	142
80	60	210	142
85	69	215	143
90	75	220	145
95	80	225	146
100	86	230	148
105	89	235	148
110	96	240	149
115	99	245	150
120	105	250	151
125	107	255	157

Figura 33: Assignació valors PWM-RPM[33]

[32]Font: `https://www.arduino.cc/reference/en/language/functions/analog-io/analogwrite/`

[33]Font: imatge pròpia

En la figura anterior, s'observa les velocitats RPM acceptades per les rodes motrius en la llibreria MOTOR, on, cada resultat de velocitat RPM és un valor mitjà per cada senyal PWM.

RPM value	RPM value
0	121
27	125
41	126
48	130
56	134
60	137
69	140
75	141
80	142
86	143
89	145
96	146
99	148
105	149
107	150
114	151
115	157
120	

Figura 34: Valors RPM acceptats per la llibreria MOTOR[34]

[34]Font: imatge pròpia

3.1.4 Control del posicionamet de la Roda

Tot seguit, s'explicarà l'estratègia implementada en la llibreria MOTOR a fi de controlar el posicionament de les rodes motrius, en concret els graus de gir, i conseqüentment el seu desplaçament circular.

L'estratègia es basa en les següents dades, amb les quals es pot conèixer quants polsos es requereixen per cada volta de la roda acoblada al motor, i per tant, per girà 360 graus. Un cop, conegut els polsos es pot implementar un comptador com en la secció 3.1.2, per saber quan parar els motors:

1. Relació de transmissió/reducció del motor de **i=120:1**, es a dir, 120 revolucions o voltes de l'eix del motor equival a 1 volta de l'eix de la roda.

2. Resolució de cada canal (A i B) de l'encoder, es a dir, 8 polsos per 1 revolució o volta de l'eix del motor.

3. Càlcul del nombre de polsos per cada volta de roda:

   $$960 PolsosVoltaroda = \frac{8 Polsos}{1 VoltaEixMotor} \cdot \frac{120 VoltaEixMotor}{1 VoltaRoda}$$

Per tant, és necessari 960 polsos provinents d'un canal de l'encoder per saber si la roda acoblada a l'eix del motor ha donat una volta. Expressat en graus s'obté el següent factor: $\frac{960 Polsos}{360°}$

Amb el factor anterior, és fàcil obtenir quants polsos es requereixen per girar un valor de graus donat o desplaçar-se una distància de gir circular donada, sobre la roda.

Exemple per girar 45° la roda: $120 polsos = \frac{960}{360} \cdot 45$

Exemple per desplaçar-se 1 centímetre en l'arc longitudinal de la circumferència, si prenem la roda com una circumferència:

- Fórmula de la lóngitud d'arc d'una circumferència donat un grau $X°$:
  $Longitud = \frac{2 \cdot \pi \cdot Radi \cdot X°}{360°}$

- Aplicant la fórmula a 1 centímetre de longitud d'arc de circumferència, obtenim els graus que es requereixen per desplaçar-se 1 centímetre:
  $X° = \frac{1 cm \cdot 360°}{2 \cdot \pi \cdot Radi}$
  El Radi hauria d'estar en centímetres.

- Nombre de polsos necessaris per 1 centímetre:
  $Polsos = \frac{960}{360°} \cdot \frac{1 cm \cdot 360°}{2 \cdot \pi \cdot Radi}$

3.1.5 Documentació llibreria MOTOR

Les següents funcions estan destinades a la inicialització de la llibreria MOTOR, especialment, es declaren els pins de connexió entre el driver L298N i els dos encoders SJ01 amb l'Arduino MKR WIFI 1010.

- **void initMotor(int ENA, int ENB, int EN1, int EN2, int EN3, int EN4)**
  - **Descripció:** La funció permet declarar sobre quins pins de l'Arduino MKR WiFi es connectaren els senyals ENA, ENB, EN1, EN2, EN3 i EN4 del driver L298N. Especificar, que aquesta funció s'ha de crida abans d'utilitzar qualsevol altra funció de la llibreria MOTOR.
  - **Paràmetres:**

    ENA: El paràmetre és de tipus enter i ha de correspondre a un pin capacitat per oferir senyals PMW. Aquest pin és utilitzat per controlar la velocitat RPM des del driver L298N sobre els dos motors drets del Robot Car Kit.

    ENB: El paràmetre és de tipus enter i ha de correspondre a un pin capacitat per oferir senyals PMW. Aquest pin és utilitzat per controlar la velocitat RPM des del driver L298N sobre els dos motors de l'esquerra del Robot Car Kit.

    EN1: El paràmetre és de tipus enter i ha de correspondre a un pin digital. Aquest pin és utilitzat per controlar la direcció de gir des del driver L298N sobre els dos motors drets del Robot Car Kit.

    EN2: El paràmetre és de tipus enter i ha de correspondre a un pin digital. Aquest pin és utilitzat per controlar la direcció de gir des del driver L298N sobre els dos motors drets del Robot Car Kit.

    EN3: El paràmetre és de tipus enter i ha de correspondre a un pin digital. Aquest pin és utilitzat per controlar la direcció de gir des del driver L298N sobre els dos motors de l'esquerra del Robot Car Kit.

    EN4: El paràmetre és de tipus enter i ha de correspondre a un pin digital. Aquest pin és utilitzat per controlar la direcció de gir des del driver L298N sobre els dos motors de l'esquerra del Robot Car Kit.

  - **Return:** No retorna cap valor.

- **void initEnconder(int encodRightPin, int encodLeftPin)**

  - **Descripció:** La funció permet declarar sobre quins pins de l'Arduino MKR WiFi es connectaren els senyals provinents dels dos encoders frontals del Robot Car Kit. Recordar, que cada encoder té dos canals, A i B, i només es farà ús d'un canal de cada encoder (en aquest projecte s'ha optat pels canals B), els quals permeten el càlcul de la velocitat RPM sobre les rodes de la dreta i esquerra del Robot Car Kit.
  - **Paràmetres:**

    encodRightPin: El paràmetre és de tipus enter i ha de correspondre a un canal de l'encoder SJ01 del motor dret del Robot Car Kit. En aquest projecte s'ha optat per la utilització dels canals B en els encoders.

    encodLeftPin: El paràmetre és de tipus enter i ha de correspondre a un canal de l'encoder SJ01 del motor esquerrà del Robot Car Kit. En aquest projecte s'ha optat per l'utilització dels canals B en els encoders.

  - **Return:** No retorna cap valor.

Les següents funcions s'utilitzen per controlar la direcció de gir i el control de la velocitat en RPM de les rodes motrius.

- **void setDirectionMotor(char motors, char directionMotor)**

  - **Descripció:** La funció permet la configuració de la direcció de gir de les rodes motrius. Notar, que aquesta funció només configura la direcció de gir de les rodes, però no les posa en marxa.
  - **Paràmetres:**

    motors: El paràmetre és tipus char i s'utilitza per seleccionar per quins motors es configurarà la direcció de gir.
    Quan el paràmetre *motors* val 'R' de *right*, es selecciona els motors drets del Robot Car Kit.

    Quan el paràmetre *motors* val 'L' de *left*, es selecciona els motors de l'esquerra del Robot Car Kit.

Quan el paràmetre *motors* val 'B' de *both*, es selecciona els motors de la dreta i esquerra del Robot Car Kit.

directionMotor: El paràmetre és tipus char i s'utilitza per configurar la direcció de gir dels motors seleccionats.

Quan el paràmetre *directionMotor* val 'F' de *forward*, els motors seleccionats giraran en el sentit horari del rellotge, es a dir, permetrà avançar al Robot Car Kit.

Quan el paràmetre *directionMotor* val 'B' de *backward*, els motors seleccionats giraran en el sentit antihorari del rellotge, es a dir, permetrà retrocedir al Robot Car Kit.

Quan el paràmetre *directionMotor* val 'S' de *stop*, els motors seleccionats pararen, es a dir, permetrà parar al Robot Car Kit. Qualsevol configuració feta prèviament amb la funció *directionMotor* serà esborrada.

Quan el paràmetre *directionMotor* val 'K' representant *break*, els motors seleccionats no giraran, però matindrà qualsevol configuració prèvia realitzada amb la funció *setDirectionMotor*, es a dir, si la configuració ha estat forward aquesta es mantindrà.

– **Return:** No retorna cap valor.

- **void analogWriteMotor(char motors, int pwm)**
  - **Descripció:** Un cop la direcció de gir ha estat configurada amb la funció *setDirectionMotor*, aquesta funció permet posar en marxa els motors seleccionats mitjançat un senyal PWM. El paràmetre *pwm* serà passat de forma directa a la funció *analogWrite* d'Arduino, per tant, es segueixen les seves condicions d'ús.
  - **Paràmetres:**

    motors: El paràmetre és tipus char i s'utilitza per seleccionar quins motors es posaran en marxa.
    Quan el paràmetre *motors* val 'R' de *right*, es selecciona els motors drets del Robot Car Kit.

    Quan el paràmetre *motors* val 'L' de *left*, es selecciona els motors de l'esquerra del Robot Car Kit.

    Quan el paràmetre *motors* val 'B' de *both*, es selecciona els motors de la dreta i esquerra del Robot Car Kit.

    pwm: El paràmetre és de tipus enter sense i indica el senyal PWM que es subministrarà als motors seleccionats. Els valors acceptats són del 0 fins al 255.

- **void runMotor(char motors, uint8_t rpmWheel)**
  - **Descripció:** Un cop la direcció de gir ha estat configurada amb la funció *setDirectionMotor*, aquesta funció permet posar en marxa els motors seleccionats assignant una velocitat RPM a les seves rodes acoblades.
  - **Paràmetres:**

    motors: El paràmetre és tipus char i s'utilitza per seleccionar quins motors es posaran en marxa.
    Quan el paràmetre *motors* val 'R' de *right*, es selecciona els motors drets del Robot Car Kit.

    Quan el paràmetre *motors* val 'L' de *left*, es selecciona els motors de l'esquerra del Robot Car Kit.

Quan el paràmetre *motors* val 'B' de *both*, es selecciona els motors de la dreta i esquerra del Robot Car Kit.

rpmWheel: El paràmetre és de tipus enter sense signe de 8 bits i indica quina velocitat en RPM assolirà l'eix de la roda dels motors seleccionats, es a dir, la velocitat de les rodes acoblades. Els valors acceptats per aquests paràmetres són:

RPM value	RPM value
0	121
27	125
41	126
48	130
56	134
60	137
69	140
75	141
80	142
86	143
89	145
96	146
99	148
105	149
107	150
114	151
115	157
120	

Figura 35: Valors RPM acceptats per la llibreria MOTOR[35]

– **Return:** No retorna cap valor.

[35]Font: imatge pròpia

Les següents funcions estan orientades a poder aconseguir quina velocitat assoleixen les rodes motrius en RPM a partir del rang de valor donat per la funció **analogWrite** en la creació de senyals PWM. En definitiva, permeten conèixer quina velocitat RPM assoleix un senyal PWM donat per la funció **analogWrite**.

- **void getRPM_BMotor(int pwm, float *rpmWheelR, float *rpmWheelL, unsigned long timeSample)**
  - **Descripció:** Un cop la direcció de gir ha estat configurada amb la funció *setDirectionMotor*. La funció permet el càlcul de les velocitats en RPM pels motors (rodes) de la dreta i esquerra del Robot Car Kit donat un senyal PWM durant un temps de mostreig determinat. Especificar, que quan el temps de mostreig s'acaba els motors seleccionats seguiran amb la mateixa configuració feta prèviament amb *setDirectionMotor*, i per tant els motors no es pararen, caldrà crida a la funció *setDirectionMotor* en el mode stop (S) o break (K) per parar-los.
  - **Paràmetres:**

    pwm: El paràmetre és un enter i indica el cicle de treball del senyal PWM generat. El rang de valor permesos són des del 0 fins a 255. Aquest paràmetre serà passat directament a la funció *analogWrite* i per tant segueix les seves normes.

    rpmWheelR: El paràmetre és un apuntador de tipus float i indica on es desarà la velocitat RPM assolida pels motors (rodes) drets del Robot Car Kit, un cop acabat el temps de mostreig.

    rpmWheelL: El paràmetre és un apuntador de tipus float i indica on es desarà la velocitat RPM assolida pels motors (rodes) de l'esquerra del Robot Car Kit, un cop acabat el temps de mostreig.

    timeSample: El paràmetre és de tipus unsigned long i indica el temps de mostreig en milisegons.

  - **Return:** No retorna cap valor.

- **void getRPM_Motor(int pwm, float *rpmWheel, char motor, unsigned long timeSample)**

  - **Descripció:** Un cop la direcció de gir ha estat configurada amb la funció *setDirectionMotor*. La funció permet el càlcul de les velocitats en RPM pels motors (rodes) de la dreta i esquerra del Robot Car Kit donat un senyal PWM durant un temps de mostreig determinat. Especificar, que quan el temps de mostreig s'acaba els motors seleccionats seguiran amb la mateixa configuració feta prèviament amb *setDirectionMotor*, i per tant els motors no es pararen, caldrà crida a la funció *setDirectionMotor* en el mode stop (S) o break (K) per parar-los.
  - **Paràmetres:**

    pwm: El paràmetre és un enter i indica el cicle de treball del senyal PWM generat. El rang de valor permesos són des del 0 fins a 255. Aquest paràmetre serà passat directament a la funció *analogWrite* i per tant segueix les seves normes.

    rpmWheel: El paràmetre és un apuntador de tipus float i indica on es desarà la velocitat RPM assolida pels motors (rodes) seleccionats del Robot Car Kit, un cop acabat el temps de mostreig.

    motor: El paràmetre és de tipus char i indica quins motors (rodes) es seleccionaran pel càlcul de la velocitat en RPM del Robot Car Kit.

    Quan el paràmetre *motor* val 'R' de *right*, es seleccionen els motors (rodes) drets.
    Quan el paràmetre *motor* val 'L' de *left*, es seleccionen els motors (rodes) de l'esquerra.

    timeSample: El paràmetre és de tipus unsigned long i indica el temps de mostreig en milisegons.

  - **Return:** No retorna cap valor.

Les següents funcions estan orientades al posicionamet de les rodes motrius, en concret poder girar les rodes un valor de graus determinats, poder girar el Robot Car Kit certs graus i poder realitzar desplaçaments lineals a partir de l'arc longitudinal de les rodes.

Les accions següents s'executen amb una potència de rotació subministrada sobre els motors comuna per totes les funcions, aquesta potència de rotació pot ser variada amb la funció *powerRotationMotor*. Cal tenir present, que un valor de potència de rotació elevat comporta major inèrcia sobre les rodes motrius, la qual cosa, provocaria major resistència un cop accionat la parada de rotació dels motors, i per tant, imprecisió en els graus de gir de les rodes.

- **void powerRotationMotor(uint8_t power)**
  - **Descripció:** La funció permet variar la potència de rotació emprada en les funcions *rotateWheelMotor*, *rotateCarMotor* i *moveCentimetersMotor*.
  - **Paràmetres:**

    power: El paràmetre és de tipus enter sense signe de 8 bits i indica la potència de rotació emprada en les funcions *rotateWheelMotor*, *rotateCarMotor* i *moveCentimetersMotor*. El rang de valors del paràmetre és de 0 a 255.
  - **Return:** No retorna cap valor.

- **void rotateWheelMotor(char motors, uint16_t degrees)**
  - **Descripció:** Un cop configurat la potència de rotació i la direcció de gir, la funció permet girar certs graus les rodes motrius. Especificar, que quan el procés de rotació acabi, òbviament els motors pararan i tindran la configuració del mode break (K) de la funció *setDirectionMotor*.
  - **Paràmetres:**

    motors: El paràmetre és tipus char i s'utilitza per seleccionar quins motors (rodes) giraran.
    Quan el paràmetre *motors* val 'R' de *right*, es selecciona els motors drets del Robot Car Kit.

    Quan el paràmetre *motors* val 'L' de *left*, es selecciona els motors de l'esquerra del Robot Car Kit.

Quan el paràmetre *motors* val 'B' de *both*, es selecciona els motors de la dreta i esquerra del Robot Car Kit.

degrees: El paràmetre és de tipus enter sense signe de 16 bits i indica els graus sexagesimals que les rodes seleccionades giraran. El rang de valor d'aquest paràmetre és de 0 a 360 graus.

- **Return:** No retorna cap valor.

- **void rotateCarMotor(char motors, uint16_t degrees)**

  - **Descripció:** Un cop configurat la potència de rotació, la funció permet girar certs graus, a la dreta o esquerra, el Robot Car Kit sencer. Especificar, que quan el procés de rotació acabi, òbviament els motors pararan i tindran la configuració del mode break (K) de la funció *setDirectionMotor*.
  - **Paràmetres:**

    direction: El paràmetre és tipus char i s'utilitza per seleccionar si la rotació del Robot Car Kit serà a la dreta o esquerra.

    Quan el paràmetre *direction* val 'R' de *right*, el Robot Car Kit girarà a la dreta.

    Quan el paràmetre *direction* val 'L' de *left*, el Robot Car Kit girarà a l'esquerra.

    degrees: El paràmetre és de tipus enter sense signe de 16 bits i indica els graus sexagesimals que el Robot Car Kit girarà. Els valors acceptats són: 90, 180, 270 i 360.
  - **Return:** No retorna cap valor.

- **void moveCentimetersMotor(char motors, uint8_t centimeters)**
  - **Descripció:** Un cop configurat la potència de rotació i la direcció de gir, la funció permet girar els graus sexagesimals equivalents a un desplaçament lineal. En essència, donat un valor de desplaçament en centímetres, les rodes seleccionades giraran fins a assolir aquest desplaçament. Especificar, que quan el procés de rotació acabi, òbviament els motors pararan i tindran la configuració del mode break (K) de la funció *setDirectionMotor*.
  - **Paràmetres:**

    motors: El paràmetre és tipus char i s'utilitza per seleccionar quins motors (rodes) giraran.
    Quan el paràmetre *motors* val 'R' de *right*, es selecciona els motors drets del Robot Car Kit.

    Quan el paràmetre *motors* val 'L' de *left*, es selecciona els motors de l'esquerra del Robot Car Kit.

    Quan el paràmetre *motors* val 'B' de *both*, es selecciona els motors de la dreta i esquerra del Robot Car Kit.

    centimeters: El paràmetre és de tipus enter sense signe de 8 bits i indica els centímetres que les rodes han de girar. Els valors acceptats són des del 0 fins a 255.
  - **Return:** No retorna cap valor.

Les següents funcions estan orientades al control de la velocitat en RPM de les rodes motrius a partir un controlador PID.

- **void initPIDMotor(char motor, uint8_t setPoint, float Kp, float Ki, float Kd, unsigned long timeSample)**
  - **Descripció:** Un cop la direcció de gir ha estat configurada amb la funció *setDirectionMotor*. La funció permet el control de la velocitat en RPM sobre les rodes motrius seleccionades amb un controlador PID donat un temps d'execució. Aquesta funció ha de ser cridada abans que les funcinons *executePIDMotor* i *endPIDMotor*.
  - **Paràmetres:**

    motor: El paràmetre és tipus char i s'utilitza per seleccionar quins motors (rodes) giraran.
    Quan el paràmetre *motors* val 'R' de *right*, es selecciona els motors drets del Robot Car Kit.

    Quan el paràmetre *motors* val 'L' de *left*, es selecciona els motors de l'esquerra del Robot Car Kit.

    Quan el paràmetre *motors* val 'B' de *both*, es selecciona els motors de la dreta i esquerra del Robot Car Kit.

setPoint: El paràmetre és de tipus enter sense signe de 8 bits i indica el *set point* de velocitat en RPM que s'aplicarà al controlador PID pels motors seleccionats. Els valors acceptats per aquests paràmetres són:

RPM value	RPM value
0	121
27	125
41	126
48	130
56	134
60	137
69	140
75	141
80	142
86	143
89	145
96	146
99	148
105	149
107	150
114	151
115	157
120	

Figura 36: Valors RPM acceptats per la llibreria MOTOR[36]

Kp: El paràmetre és de tipus float i respresenta la constant *proporcional* aplicada al controlador PID.

Ki: El paràmetre és de tipus float i respresenta la constant *integrativa* aplicada al controlador PID.

Kd: El paràmetre és de tipus float i respresenta la constant *derivativa* aplicada al controlador PID.

timeSample: El paràmetre és de tipus unsigned long i indica el temps de mostreig/execució en milisegons del controlador PID.

[36]Font: imatge pròpia

  - **Return:** No retorna cap valor.

- **void endPIDMotor(void)**

  - **Descripció:** La funció permet acabar el procés del control PID. Després d'aquesta acció, els motors seleccionats en el procés de control PID es pararen i tindran la configuració stop de la funció *setDirectionMotor*.
  - **Paràmetres:** No rep cap paràmetre.
  - **Return:** No retorna cap valor.

- **void execute2PIDMotor(int *rightRPM, int *leftRPM)**

  - **Descripció:** La funció permet dur a terme l'execució del procés de control PID inicialitzat amb la funció **initPIDMotor**. Aquesta execució durarà el temps indicat en la seva inicialització, i un cop acabada, retornarà els resultats de les velocitats en RPM aconseguits. **Notar que aquesta funció ha de ser cridada en el cas d'haver seleccionat tant els motors de la dreta com els de l'esquerra, es a dir, el mode BOTH en el paràmetre motor de la funció initPIDMotor**.
  - **Paràmetres:**

    rightRPM: El paràmetre és un apuntador de tipus enter, i serà on es desarà el resultat obtingut de les velocitats en RPM de les rodes de la dreta del Robot Car Kit en el procés de control PID.

    leftRPM: El paràmetre és un apuntador de tipus enter, i serà on es desarà el resultat obtingut de les velocitats en RPM de les rodes de l'esquerra del Robot Car Kit en el procés de control PID.

  - **Return:** No retorna cap valor.

- **void executePIDMotor(int *valueRPM)**

  - **Descripció:** La funció permet dur a terme l'execució del procés de control PID inicialitzat amb la funció **initPIDMotor**. Aquesta execució durarà el temps indicat en la seva inicialització, i un cop acabada, retornarà els resultats de les velocitats en RPM aconseguits. **Notar que aquesta funció ha de ser cridada en el cas d'haver seleccionat o els motors de la dreta o l'esquerra, es a dir, el mode RIGHT o LEFT en el paràmetre motor de la funció initPIDMotor**.
  - **Paràmetres:**

    valueRPM: El paràmetre és un apuntador de tipus enter, i serà on es desarà en el resultat obtingut de les velocitats en RPM de les rodes seleccionades.

  - **Return:** No retorna cap valor.

3.2 Mòdul d'Ultrasò Dirigible

El mòdul d'ultrasò dirigible està compost per dos components, el sensor d'ultrasò HC-SR04 i el servo motor SG90. En les següents seccions s'explicaran el funcionament dels components i les seves llibreries corresponents per la seva interacció.

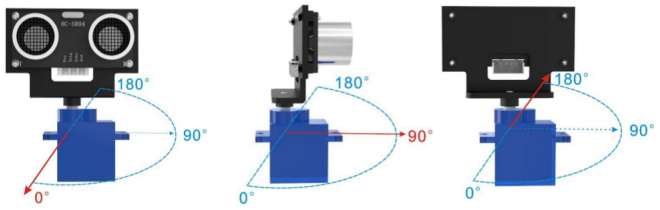

Figura 37: Ultrasò dirigible[37]

3.2.1 Servo motor SG90

El servo motor SG90 té 3 línies de connexió, dos per l'alimentació del dispositiu i una línia de senyal pel control del seu eix rotatori. La rotació de l'eix del servo motor es controla a partir d'un senyal PWM sobre la seva línia de control. Aquest senyal serà un pin de sortida en l'Arduino MKR WiFi 1010.

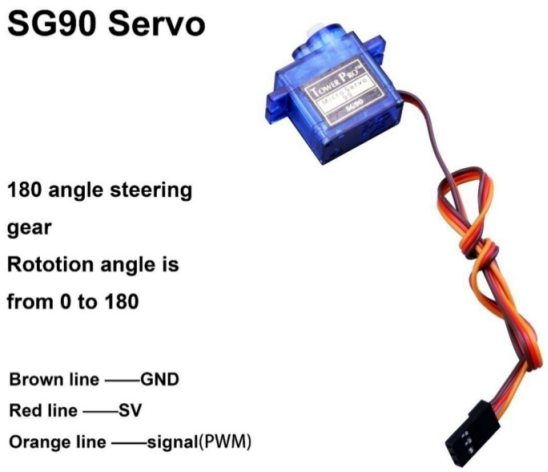

Figura 38: Servo SG90[38]

[37]Font: manual del Robot Car Kit
[38]Font: manual Robot Car Kit

El servo motor SG90 només permet un rang de gir de 180 graus. El control del servo motor es basa en l'enviament d'un senyal PWM des del microcontrolador, de període 20ms i un cicle de treball (nivell en alt lògic) entre 0.5ms i 2.5ms, amb el qual es podrà informar al servo motor quin angle es vol assolir. En la següent figura s'observa l'assignació entre els temps de cicles de treball per cada angle de posicionament.

0.5ms	0 degree
1.0ms	45 degree
1.5ms	90 degree
2.0ms	135 degree
2.5ms	180 degree

Figura 39: Servo Taula d'angles-cicle de treball senyal PWM[39]

Tot i així, el control del servo motor SG90 es farà mitjançat la llibreria **Servo.h** d'Arduino. On principalment, es declara el pin de connexió amb el servo motor, i tot seguit a partir d'una funció es pot assignar quin grau de posicionament es vol assolir.

```
Servo myservo;  // create servo object to control servo
 myservo.attach(3);   // attach servo on pin 3 to servo object
 myservo.write(90); //set servo position according to scaled value
```

Figura 40: Exemple Servo.h: exemple per posicionar l'eix del servo motor en l'angle 90[41]

[39]Font: manual del Robot Car Kit
[41]Font: manual del Robot Car Kit

3.2.2 Ultrasò HC-SR04

L'ultrasò HC-SR04 és un mòdul emprat per la detecció l'objecte a partir d'ultrasons. Aquest sensor té 4 línies de connexió, 2 de les quals són per alimentació, una línia anomenada *trigger* i l'altra anomenada *echo*. Aquestes dues últimes línies s'utilitzen per controlar l'ultrasò.

Main technical parameters (1):
voltage used: DC---5V (2): static
current: less than 2mA (3): level
output: higher than 5V (4): level
output: lower than 0
(5): detection angle: not bigger than 15 degree (6):
detecting distance: 2cm-450cm
(7): high precision: up to 0.2cm
Method of connecting lines: VCC, trig (the end of controlling), echo (the end of receiving), GND

Figura 41: Principals aspectes tècnics HC-SR04[42]

Principalment, el sensor espera un senyal d'activació des de l'Arduino MKR WiFi 1010 sobre el seu pin *trigger*, un cop rep aquest senyal, aquest envia 8 senyals quadrats amb una freqüència de 40 KHz de forma automàtica, i en cas de rebre un senyal de rebot sobre l'objecte, activa un pols sobre el pin *echo*, on la seva durada de cicle de treball en alt lògic, és el temps entre el senyal d'enviament i recepció que capta el sensor d'ultrasò, el qual pot ser detectat per l'Arduino.

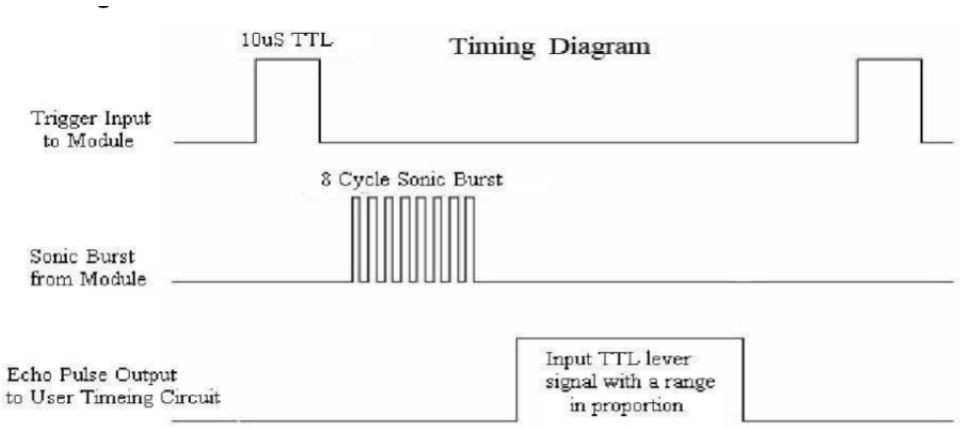

Figura 42: Timing dels senyals trigger i echo[43]

[42]Font: manual del Robot Car Kit
[43]Font: manual del Robot Car Kit

Coneixent, la durada del pols echo en alt lògic i tenint en compte que aquest temps és el que triga el senyal d'ultrasò en ser enviat del sensor a l'objecte i rebotat de l'objecte al sensor, podem aplicar la següent fórmula per conèixer la seva distància al sensor:

$$Distancia = \frac{DuradaPolsEcho \cdot VelocitatSo}{2}$$

Velocitat del sò = 340 metres/segon

Seguint aquest funcionament s'implementarà la llibreria relacionada al sensor d'ultrasò, anomenat **ULTRASOUND**, en concret per captura el temps en alt del pols *echo* es farà amb la funció *pulseIn*[comc] d'Arduino.

3.2.3 Documentació llibreria ULTRASOUND

- **void initUltSound(int Echo, int Trig)**
  - **Descripció:** La funció permet declarar sobre quins pins de l'Arduino MKR WiFi es connectaren els senyals Trigger i Echo del sensor d'ultrasò.
  - **Paràmetres:**

    Echo: El paràmetre és de tipus enter i indica sobre quin pin entrarà el senyal Echo del sensor d'ultrasò.

    Trig: El paràmetre és de tipus enter i indica sobre quin pin sortirà el senyal Trigger al sensor d'ultrasò.

  - **Return:** No retorna cap valor.
- **float distanceCMUltSound(void)**
  - **Descripció:** La funció permet activar el funcionament de l'ultrasò i conèixer a quina distància en centímetres es troba un objecte.
  - **Paràmetres:** No rep cap paràmetre.
  - **Return:** Retorna la distància en centímetres d'un objecte detectat per l'ultrasò.
- **float distanceUltSound(void)**
  - **Descripció:** La funció permet activar el funcionament de l'ultrasò i conèixer a quina distància en metres es troba un objecte.
  - **Paràmetres:** No rep cap paràmetre.
  - **Return:** Retorna la distància en metres d'un objecte detectat per l'ultrasò.

3.3 Mòdul de Sensor de Línia

El mòdul de sensor de línia està compost per tres sensors de llum infraroja, en concret, el model *HCARDU0005*. Principalment, aquest mòdul està situat en la part inferior del Robot Car Kit i està pensat per la detecció i seguiment d'una línia negre sobre una superfície reflectant de llum infraroja, com ara, una superfície de color blanc. La llibreria corresponent a aquest mòdul, anomenada **LINETRACKING**, és la més simple, i consistirà en la lectura dels senyals de sortida (recepció de la llum infraroja) que ofereixen els sensors, que conjuntament amb la llibreria MOTOR es pot implementar un seguidor de línia simple.

Figura 43: Mòdul de Sensor de Línia[44]

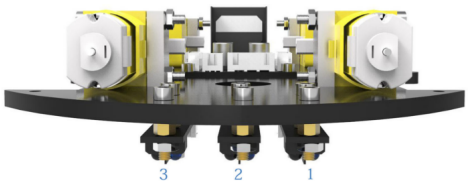

Figura 44: Mòdul de Sensor de Línia,localització en el Robot[45]

[44]Font: manual del Robot Car Kit
[45]Font: manual del Robot Car Kit

En el seguidor de lína, i tal com s'observa en la figura anterior, el sensor marcat amb el número 2 i situat en mig és per la detecció de la línia negre, el sensor macat amb el número 3 és per la detecció d'una zona blanca en l'exterior dret del Robot Car Kit i el sensor marcat amb el número 1 per la zona blanca en l'exterior esquerra. D'aquesta forma un exemple de seguidor de línia simple vindria a ser com:

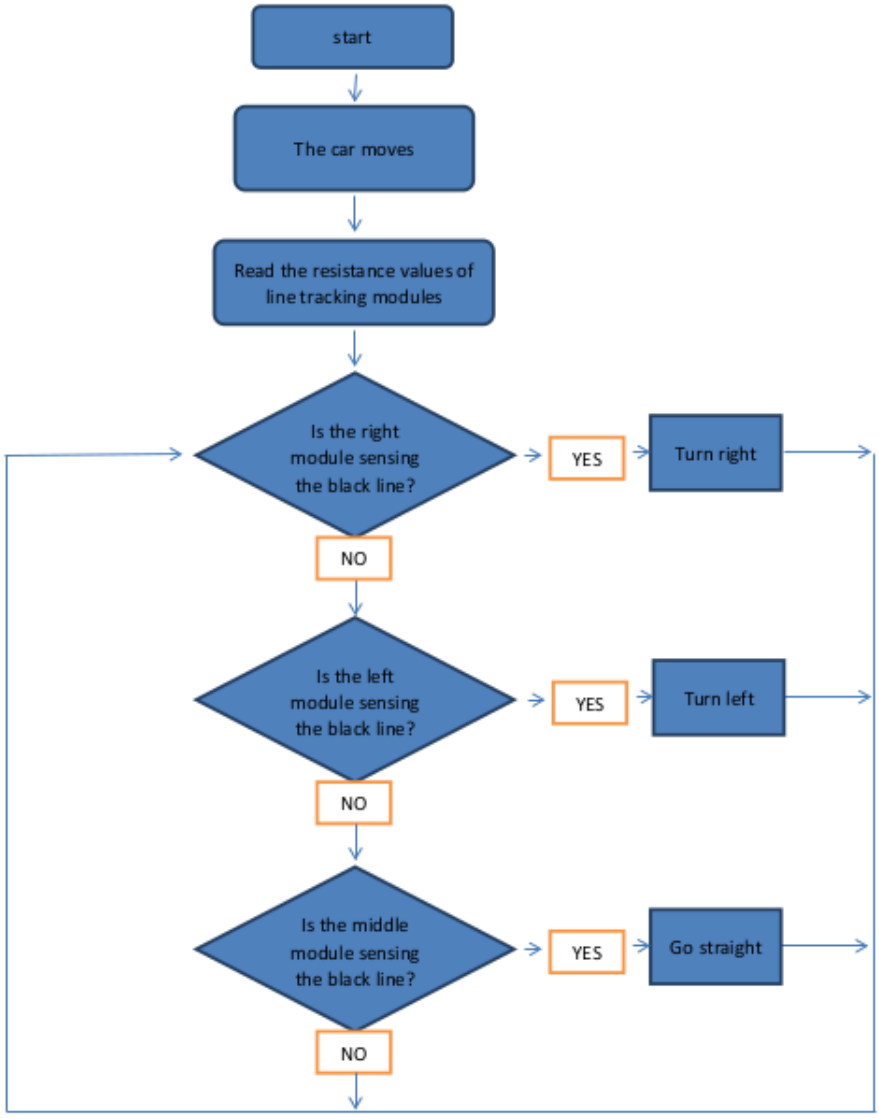

Figura 45: Exemple de l'algorisme d'un seguidor de línia simple[46]

[46]Font: manual del Robot Car Kit

3.3.1 Sensor HCARDU0005

El funcionament del sensor HCARDU0005 consisteix en la transmissió d'una llum infraroja gràcies a un LED de llum infraroja i la seva recepció gràcies a un fotoresistor, on la recepció de la llum transmesa sobre una superfície blanca difereix de forma significativa si fos negre.

El sensor ofereix un potenciòmetre per tal de regular la sensibilitat de la recepció de llum:

Figura 46: Potenciòmetre del sensor HCARDU0005[47]

Respecte als pins de connexió, el sensor té tres pins VCC (V+), GND (G) i Signal (S), els pins VCC i GND són per alimentació a 5 volts i el pin Signal és la sortida del sensor en format TTL, en concret s'obté 3,36 volts (senyal en alt observat en l'oscil·loscopi) quan el sensor detecta una superfície negre (no reflectant) i 0 volts (senyal en baix) quan detecta una superfície blanca (reflectant). Cal esmentar, que la sortida d'aquest sensor pot tindre connexió directa amb l'Arduino MKR WiFi 1010, però, pel seu correcte funcionament cal alimentar al sensor amb 5 volts.

[47]Font: manual del Robot Car Kit

3.3.2 Documentació llibreria LINETRACKING

- **void initLineTracking(int pinSensorLeft, int pinSensorMiddle, int pinSensorRight)**

  - **Descripció:** La funció permet declarar sobre quins pins de l'Arduino MKR WiFi es connectaren els senyals provinents dels 3 sensors de llum infraroja.

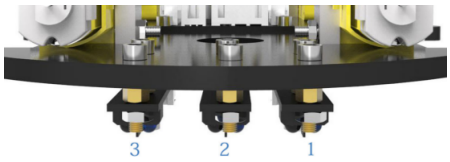

Figura 47: Disposició dels sensors en el Robot: sensor 1 (situat a l'esquerra), sensor 2 (situat en mig), sensor 3 (situat a la dreta)[49]

  - **Paràmetres:**

    pinSensorLeft: El paràmetre és de tipus enter i indica sobre quin pin entrarà el senyal de sortida del sensor de llum infraroja situat a l'esquerra del Robot Car Kit.

    pinSensorMiddle: El paràmetre és de tipus enter i indica sobre quin pin entrarà el senyal de sortida del sensor de llum infraroja situat en mig del Robot Car Kit.

    pinSensorRight: El paràmetre és de tipus enter i indica sobre quin pin entrarà el senyal de sortida del sensor de llum infraroja situat a la dreta del Robot Car Kit.

  - **Return:** No retorna cap valor.

[49]Font: manual del Robot Car Kit

- **bool reflexionLight(char sensor)**
  - **Descripció:** La funció permet conèixer quin dels 3 sensors de llum infraroja declarats en la funció *initLineTracking* detecta una superfície reflectant, es a dir, una superfície blanca.
  - **Paràmetres:** El paràmetre es de tipus char i indica quin sensor es selecciona. Quan val 'L' de left indica el sensor esquerrà, quan val 'M' de middle indica el sensor d'en mig i quan val 'R' de right indica el sensor dretà.
  - **Return:** Retorna el boolea true quan detecta una superfície reflectant (color blanc), i en cas contrari false (superfície negre).

3.4 Connexió Mòduls Robot Car Kit amb Arduino MKR WiFi 1010

Els mòduls del Robot Car Kit han estat desenvolupats per treballar amb l'Arduino Uno, que, a diferència de l'Arduino MKR WiFi 1010, treballa a 5 volts en la comunicació amb els sensors implicats del Robot Car Kit. En canvi, l'Arduino MKR WiFi 1010 treballa a 3.3 volts, la qual cosa no permet la comunicació directa amb els mòduls del Robot Car Kit, ja que, tant els senyals entrants a l'Arduino MKR WiFi 1010 malmetrien el dispositiu o els senyals de sortida de l'Arduino MKR serien incompatibles amb els mòduls del Robot Car Kit.

A causa d'aquest fet, es farà ús del convertidor de nivells BOB 11978 per passar els senyals de 5 volts a 3.3 i el de 3.3 a 5 volts. Alternativament, també es pot implementar un divisor de tensió per passar els senyals de 5 a 3.3 .

Aquest convertidor té dos canals per la conversió, cada canal ofereix dues línies de conversió, una per passar de 5 a 3.3 i l'altre de 3.3 a 5. En concret, els senyals entrants en els pins *TX1* amb 3.3 volts sortiran pels pins *TX0* a 5 volts. Els senyals entrants pels pins *RX1* amb 5 volts sortiran pels pins *RX0* amb 3.3 volts.

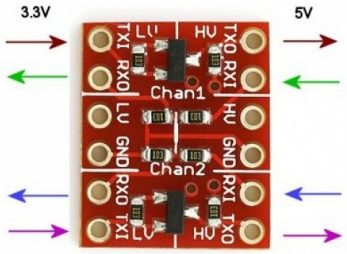

Figura 48: BOB 11978 convertidor lògic de nivell[50]

[50]Font:`http://www.hobbytronics.co.uk/image/cache/data/sparkfun/logic_level_converter_4-500x500.jpg`

4 Aplicacions de mostra

Les aplicacions de mostra es componen de dos sistemes que permeten observar en funcionament les llibreries desenvolupades en aquest projecte. Els dos sistemes tenen l'objectiu de comunicar el Robot Car Kit mitjançat la capa de comunicació oferida per les llibreries TCPSERVER i TCPCLIENT. En les següents seccions es desenvoluparen les aplicacions de mostra.

4.1 Primera aplicació de mostra

La primera aplicació es tracta del desenvolupament d'una aplicació web amb la qual poder controlar diversos programes implementats sobre el Robot Car Kit.

El funcionament de l'aplicació recau sobre quatre components principals:

1. Servidor web.
2. Servidor socket intermedi (*pipe*).
3. Capa de comunicació Robot Car Kit (TCPCLIENT).
4. Programes de mostra Robot Car Kit.

Principalment, el funcionament del sistema es basa en el fet de rebre ordres o comandes des del servidor web, es a dir, informació provinent d'un usuari, i arribar a transmetre'ls al Robot Car Kit. De la mateixa forma, també es busca poder enviar informació des del Robot Car Kit al servidor web, amb el fi d'establir una comunicació bidireccional. Aquesta comunicació bidireccional s'implementa a partir del sevidor socket intermedi i la capa de comunicació del Robot Car Kit en mode client.

Així doncs, la raó per la qual s'utilitza el servidor intermedi per establir la comunicació bidireccional, es a causa del fet que no es pot establir una comunicació bidireccional entre el servidor web i la capa de comunicació del Robot Car Kit en mode client, on l'inici de connexió pot ser produït per qualsevol dels dos extrems. El problema existent és que el servidor web admet com vies de comunicació el mètode per peticions/resposta HTTP i la connexió a un servidor socket en mode client, conseqüentment, es pot establir comunicació des del Robot Car Kit al servidor web, cada cop que el Robot Car Kit realitza una petició HTTP, però per contra, el servidor web no pot establir la comunicació per missatges HTTP, ja que, ha de ser el client qui comenci el cicle de petició/resposta HTTP. Com a resultat d'aquest problema es fa ús d'un servidor socket (TCP/IP) intermedi, que actua com un túnel (*pipe*) entre el servidor web i el Robot Car Kit, i permet l'inici de comunicació per qualsevol dels dos extrems. D'aquesta forma, s'estableix una comunicació bidireccional permanent

entre el servidor socket intermedi (TCP/IP) i la capa de comunicació del Robot Car Kit en mode client.

Respecte a la comunicació entre el servidor web i el servidor socket intermedi es produirà de la següent forma:

1. La comunicació del servidor web al servidor socket es farà amb una connexió en mode client del servidor web al servidor socket intermedi, aquesta comunicació no pot ser permanent i només s'efectuarà cada cop que es produeixi una acció en el servidor web quan es requereixi. En el cas de l'aplicació, aquesta situació es redueix, quan un usuari en el servidor web envia un ordre al Robot Car Kit. La seqüència que segueix l'ordre per arribar al Robot Car Kit és, l'obertura instantània d'una connexió entre el servidor web (en mode client socket) i el servidor socket intermedi, per on el missatge s'enviarà a aquest últim, i un cop arribat fins al servidor intermedi, aquest encaminarà el missatge cap a la capa de comunicació del Robot Car Kit mitjançat la connexió client-servidor permanent existent entre ells.

2. La comunicació del servidor socket intermedi cap al servidor web es farà gràcies a missatges HTTP. En resum, qualsevol informació del Robot Car Kit en sentit al servidor web, passarà en primer lloc per la connexió socket client-servidor existent entre el Robot Car Kit i el servidor intermedi, i en segon lloc, el servidor intermedi encaminarà (enviarà) la informació al servidor web mitjançat una petició HTTP a la seva API web disponible, en concret sobre una petició POST a la direcció *IPServidorWeb:port/api/messageServerPipe*.

En la següent figura s'observa de forma gràfica l'estructura de l'aplicació:

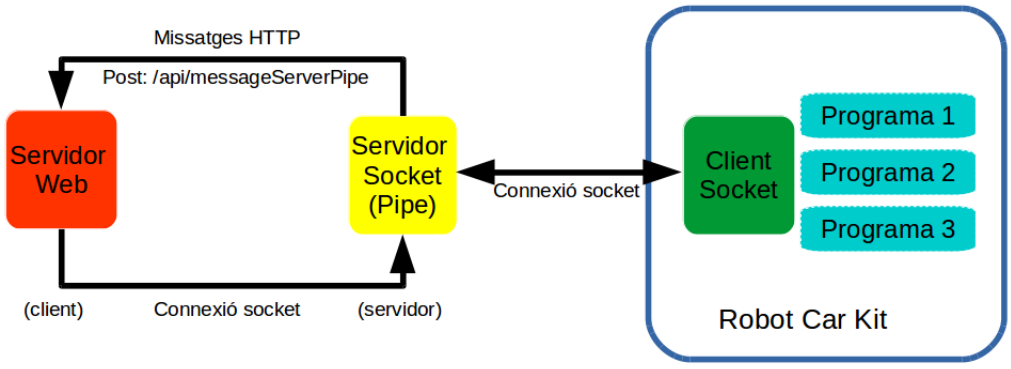

Figura 49: Xarxa de comunicació de l'apliació[51]

[51]Font: imatge pròpia

4.1.1 Servidor Web

El servidor web s'ha implementat amb el framework Laravel. Laravel és un framework de codi obert emprat pel desenvolupament d'aplicacions i serveis web amb PHP, la seva estructura de treball es basa en el sistema *Model-Vista-Controlador (MVC)* utilitzat ampliament en la implementació d'aplicions web, ja que, està orientat a les bones pràctiques en el desenvolupament de codi.

Per una millor comprensió es presenta la definició del sistema *Model-Vista-Controlador (MVC)*:

- Model-vista-controlador (MVC)[comk] és un patró d'arquitectura de software, que separa les dades i la lògica de negoci d'una aplicación de la seva representació i el mòdul encarregat de gestionar els esdeveniments i les comunicacions. Per Això MVC proposa la construcció de tres components diferents que són el model, la vista i el controlador, es a dir, d'una banda defineix components per a la representació de la informació, i d'altra banda per a la interacció del usuario[52].

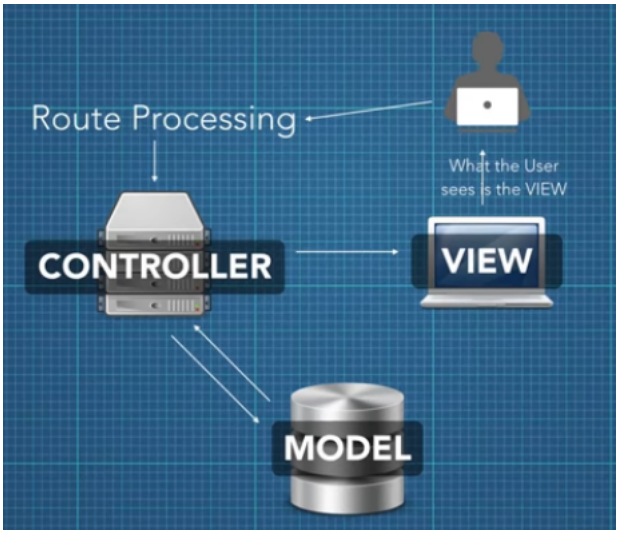

Figura 50: Sistema MVC[53]

[52]Font: defínició extreta de Wikipedia (modelo-vista-controlador)
[53]Font: `https://www.youtube.com/watch?v=1IsL6g2ixak`

Especificar, que sobre aquesta framework es desenvoluparà els pertinents models, vistes i controladors per implementar tant el fronted (representació de la informació) com el backend (lògica de negoci) corresponents als quatre programes disponibles al Robot Car Kit, es a dir, les interfícies d'usuari pertinents i la seva lògica de control.

Pel que fa a la pàgina web, en essència, es compon d'un menú principal on poder seleccionar el programa per executar-se sobre el Robot Car Kit, i les vistes corresponents a cada programa des de les quals poder controlar al Robot.

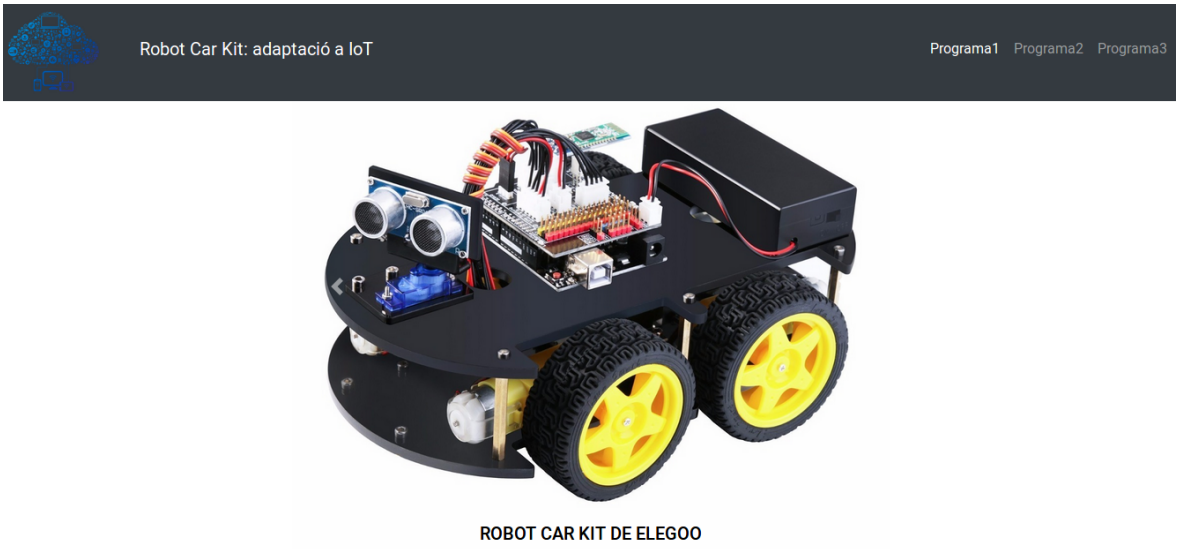

Figura 51: Menú principal de la pàgina web[54]

[54]Font: imatge pròpia

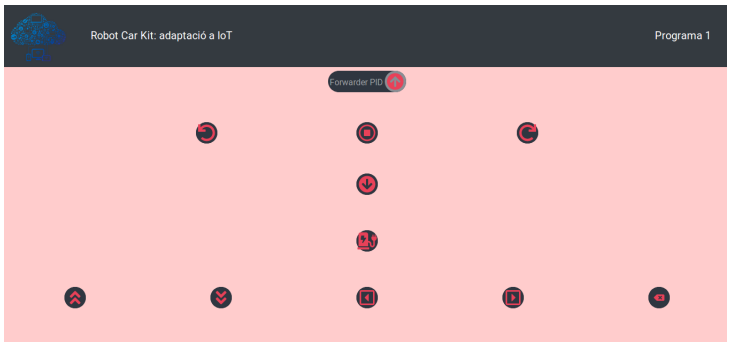

Figura 52: Vista programa 1 de la pàgina web[55]

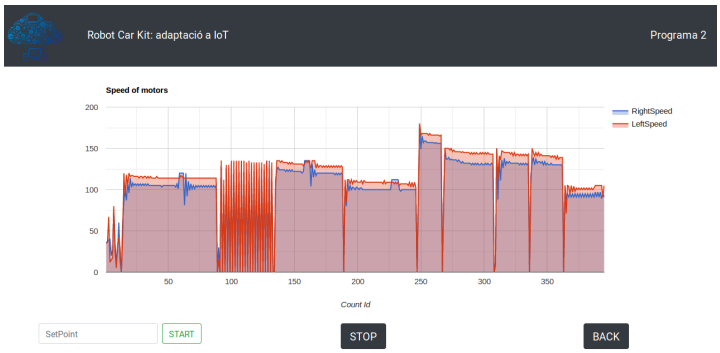

Figura 53: Vista programa 2 de la pàgina web[56]

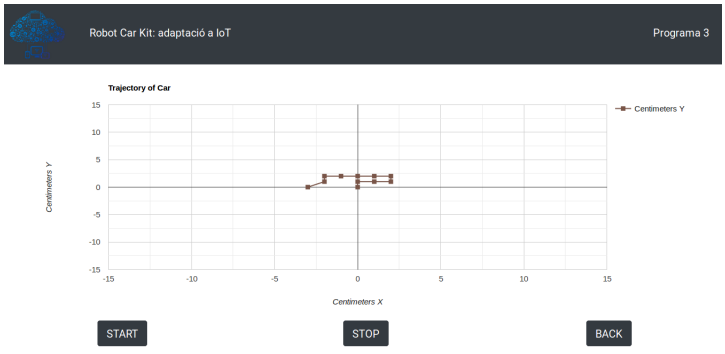

Figura 54: Vista programa 3 de la pàgina web[57]

[55]Font: imatge pròpia
[56]Font: imatge pròpia
[57]Font: imatge pròpia

En concret, pels programes 2 i 3 s'ha implementat una estructura addicional que treballa en conjunt amb el servidor web per l'actualització/captació de dades a temps real sobre els clients connectats al servidor web, ja que, són els dos únics programes en els quals hi ha tràfic de dades provinents del Robot Car Kit en sentit al servidor web, i és necessari conèixer quan hi ha dades noves (recursos nous) disponibles de forma automàtica des dels clients webs, una característica molt habitual en aplicacions IoT.

Aquest element addicional és un *WebSocket*. L'estructura WebSocket s'ha desenvolupat gràcies a les llibreries *Ratchet* i *ZeroMQ*. En essència, es porta a terme quan diversos clients connectats al mateix temps a un servidor web, requereixen conèixer si algun dels clients ha efectuat una acció que repercuteixi sobre algun recurs compartit amb algun dels clients. El cas de la missatgeria instantània és un bon exemple, si dos clients estan xatejant a temps real mitjançats dos navegadors webs, quan un d'ells envia un missatge a l'altre, el WebSocket és l'element encarregat d'avisar a l'altre client que té un missatge nou o poder d'aquesta forma actualitzar la pàgina web amb la nova informació, d'altre forma, el client que no ha enviat el missatge no podria conèixer que té nous recursos disponibles (missatges pendents) fins que no actualitzi de forma manual la pàgina web.

Tot seguit, es descriu la xarxa de funcionament bàsic d'un WebSocket, on la terminologia *LAMP stack* fa referència a un servidor web:

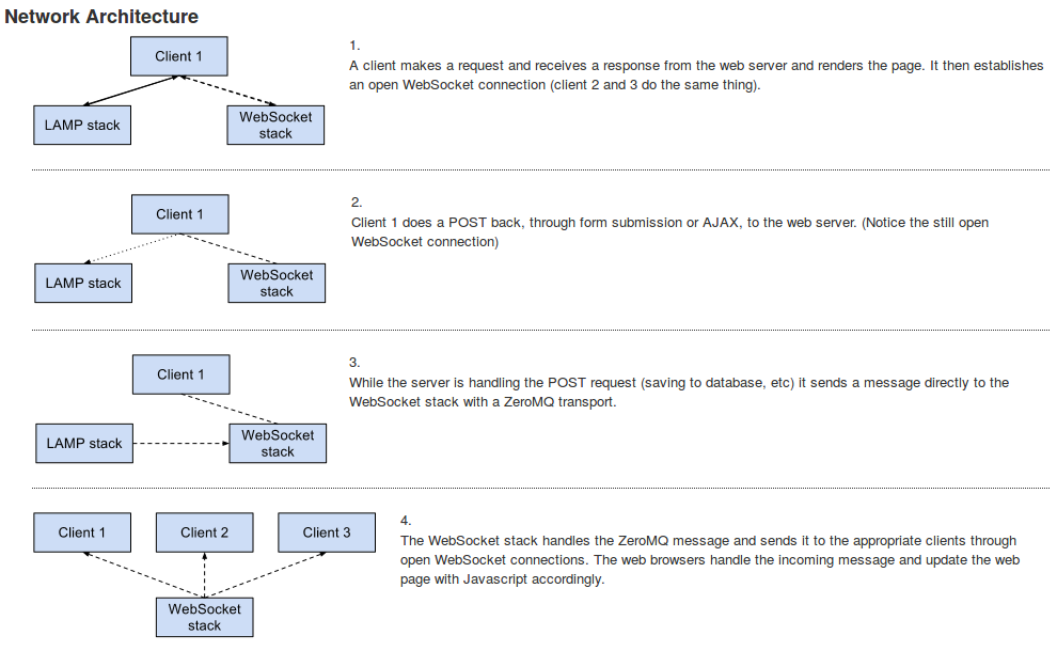

Figura 55: Arquitectura WebSocket[58]

A més, del WebSocket, aquests dos programes necessitaran dues taules simples en una base de dades per emmagatzemar les dades al llarg del temps transcorregut.

Per tant, cal especificar, que fent ús de les llibreries Ratchet i ZeroMQ, es desenvoluparà un servidor WebSocket, similar a un servidor socket (TCP/IP), capacitat per alertar sobre l'existència de nous recursos als clients connectats al servidor web. Així mateix, es programarà l'establiment de connexió en els clients webs (vistes en el navegador) i la lògica necessària per poder actualitzar la pàgina web en els clients cada cop que aquests reben l'alerta de nous recursos per part del servidor WebSocket.

Per més informació sobre el funcionament del framework Laravel i del WebSocket, podeu visitar la documentació oficial de Larevel [Otw] i Ratchet[comh].

4.1.2 Capa de comunicació Robot Car Kit

La implementació de la capa de comunicació del Robot Car Kit per aquesta aplicació s'ha fet amb la llibreria TCPCLIENT i consisteix en una màquina d'estats (**Client Finite State Machine**) que té l'objectiu de proporcionar un client socket en l'Arduino MKR WiFi 1010 capacitat per rebre informació pel protocol TCP/IP. La finalitat d'aquesta màquina d'estat és la creació d'una capa o canal de comunicació entre un servidor socket i la resta de sistemes o programes relacionats amb el Robot Car Kit que es vulguin executar.

Essencialment, la màquina d'estats **Client FSM** consisteix en la connexió cap a un servidor socket per començar a rebre informació per TCP/IP, en concret la informació serà directe per string o cadena de caràcters. El funcionament general de la màquina d'estats és, en primera instància, establir connexió amb el servidor i esperar a rebre la selecció del nom d'un programa específic per començar la seva execució. Tot seguit, tota informació arribant al client serà encaminada o passada al corresponent programa, en concret, la informació vinent serà les comandes o ordres de cada programa. Cal esmentar, que cada programa del Robot Car Kit, serà una màquina d'estat que interactuarà amb la màquina **Client FSM**.

[58]Font: `https://www.youtube.com/watch?v=1IsL6g2ixak`

Tot seguit es presenta la màquina d'estats de tipus Mealy:

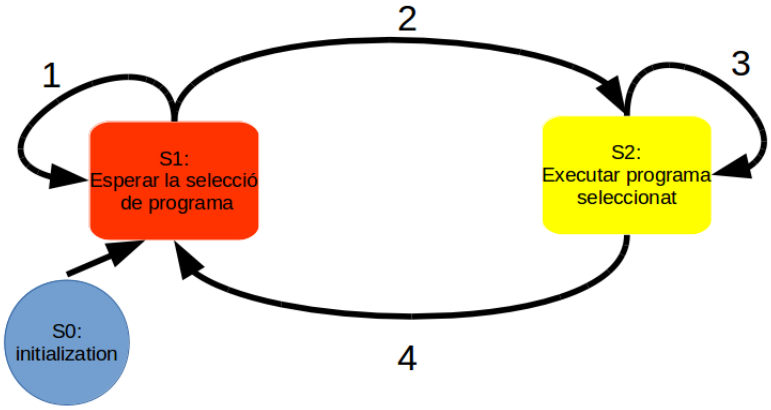

Figura 56: FSM TCPClient[59]

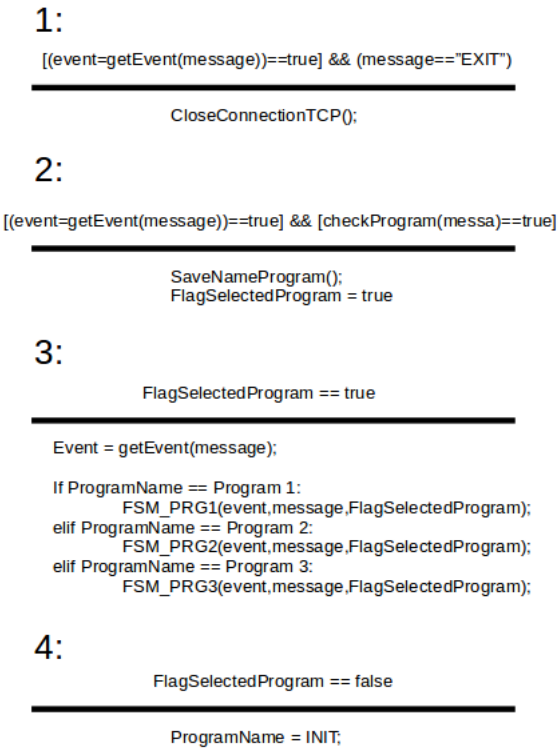

Figura 57: FSM TCPClient condionals/accions[60]

[59]Font: imatge pròpia

Descripcións de funcions i variables de la màquina d'estat:

- **message :**
  Variable on emmagatzemar el missatge provinent del servidor socket.
- **FlagSelectedProgram :**
  La variable indica si hi ha algun programa seleccionat, aquesta variable serà compartida amb les màquines d'estat dels programes 1, 2 i 3. Quan la màquina d'estats del programa seleccionat modifiqui el valor de la variable a false, el programa deixarà d'executar-se i es passarà a l'espera d'una nova selecció de programa.
- **getEvent(message) :**
  La funció retorna true en cas que hi hagin missatges disponibles provinents del servidor, a més, si es dona aquest cas, el missatge es desarà en la variable *message*, en cas contrari, retorna false.
- **checkProgram(message) :**
  La funció retorna true en cas que el missatge coincideixi amb el nom d'un programa, en cas contrari retorna false.
- **FSM_PRGX(event, message, FlagSelectedProgram) :**
  Representa la crida a la màquina d'estats del programa X. Es passen les variables event, message i FlagSelectedProgram, per què la màquina corresponent pugui conèixer quan hi ha un missatge disponible i el seu valor, i a més, poder modificar la variable FlagSelectedProgram quan es decideixi acabar l'execució del programa.
- **closeConnectionTCP() :**
  La funció tancarà la connexió amb el servidor socket.

[60]Font: imatge pròpia

4.1.3 Programes Robot Car Kit

Programa 1 del Robot Car Kit

El programa consisteix en el control remot de les rodes motrius amb diferents funcionalitats. Principalment, es tractarà d'una màquina d'estats responsable de posar en marxa funcionalitats com el control de la velocitat de les rodes amb un PID o poder desplaçar-se una distància concreta en centímetres de forma automàtica.

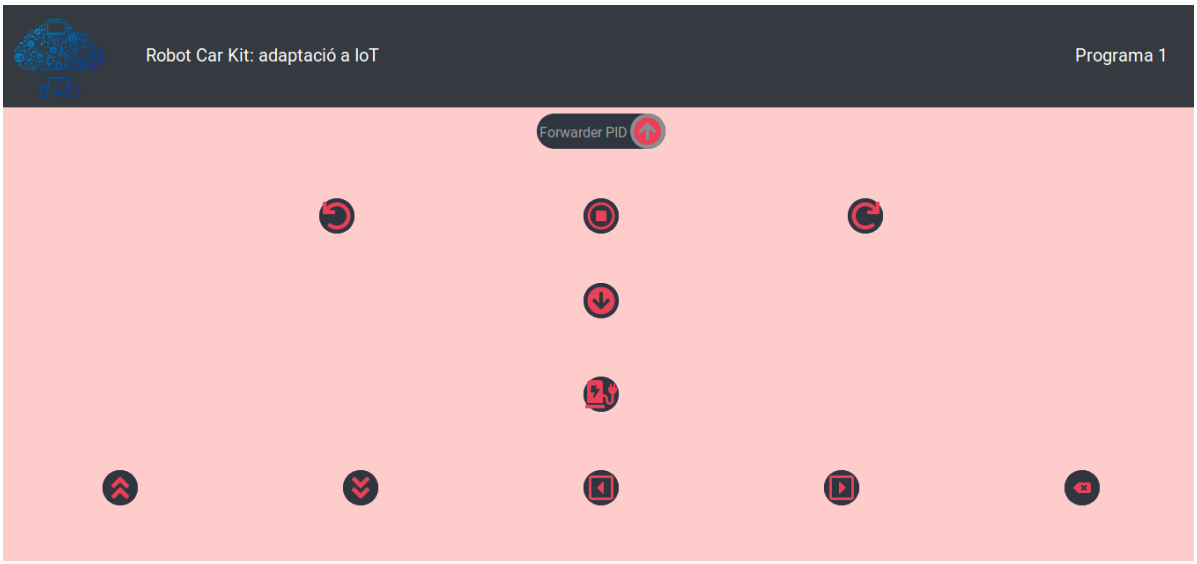

Figura 58: Vista programa 1 de la pàgina web[61]

[61] Font: imatge pròpia

Llistat de funcionalitats/ordres disponibles:

- **Stop:** Parar els motor del Robot Car Kit.
- **Forward PID:** Posa en marxa els motors amb una velocitat RPM de referència i executar un controlador PID sobre aquests. En concret, el sentit de gir de les rodes farà avançar al Robot Car Kit.
- **Backward PID:** Posa en marxa els motors amb una velocitat RPM de referència i executar un controlador PID sobre aquests. En concret, el sentit de gir de les rodes farà retrocedir al Robot Car Kit.
- **Rotate:** Fa rotar el Robot Car Kit a la dreta o l'esquerra durant un centímetre de desplaçament.
- **Rotate Car:** Fa rotar el Robot Car Kit a la dreta o l'esquerra donat un determinat valor de graus sexagesimals. Els valors acceptats són 90, 180, 270 i 360.
- **Forward Centimeters:** Desplaça el Robot Car Kit endavant donat un valor de desplaçament en centímetres. Els valors acceptats van del 1 fins al 255.
- **Backward Centimeters:** Desplaça el Robot Car Kit endarrere donat un valor de desplaçament en centímetres. Els valors acceptats van del 1 fins al 255.
- **Power Rotation:** Fa variar la potència de rotació de les rodes sobre les comandes Rotate, Rotate Car, Forward Centimeters i Backward Centimeters. Els valors acceptats van del 1 fins al 255.
- **Exit:** Provoca el fi d'execució del programa, modificant la variable FlagSelectedProgram a false.

Les funcions emprades en la màquina d'estat corresponen a la llibreria MOTOR.

Tot seguit es presenta la màquina d'estats de tipus Mealy:

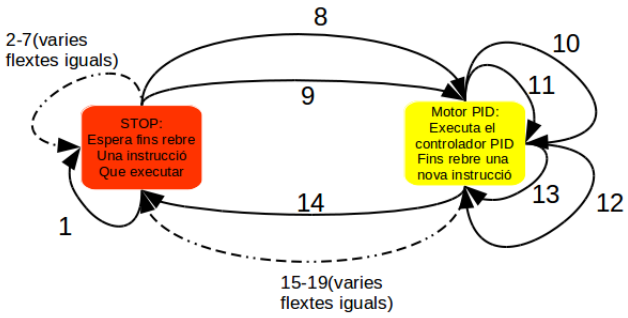

Figura 59: FSM Programa 1 [62]

1 i 14 (mateixa coindició):

(event==true) && (message=="EXIT")

```
setDirection(Both,stop);
FlagSelectedProgram = false;
```

2 i 15 (mateixa coindició):

(event==true) && (message=="ROTATE")

```
If Rotation Right:
        setDirection(Left,Forward);
        moveCentimeterMotor(Left, 1cm);
Elif Rotation Left:
        setDirection(Right,Forward);
        moveCentimeterMotor(Right, 1cm);
```

3 i 16 (mateixa coindició):

(event==true) && (message=="ROTATE CAR")

```
Degrees = getDegrees(message);
If Rotation Right:
        rotateCarMotor(Right, degrees);
Elif Rotation Left:
        rotateCarMotor(Left, degrees);
```

4 i 17 (mateixa coindició):

(event==true) && (message=="FORWARD CENTIMETERS")

```
setDirection(Both,Forward);
Centimeters = getCentimeters();
moveCentimetersMotor(Both,centimeters);
```

5 i 18 (mateixa coindició):

(event==true) && (message=="BACWARD CENTIMETERS")

```
setDirection(Both,Backward);
Centimeters = getCentimeters();
moveCentimetersMotor(Both,centimeters);
```

6 i 10 (mateixa coindició) :

(event==true) && (message=="POWER ROTATION")

```
PowerRotation = getPower();
powerRotationMotor(powerRotation);
```

7 i 19 (mateixa coindició) :

(event==true) && (message=="STOP")

```
setDirectionMotor(Both,Stop);
```

8 i 11 (mateixa coindició):

(event==true) && (message=="FORWARD PID")

```
setDirectionMotor(Both,Forward);
setPoint = getSetPoint();
initPIDMotor(setPoint);
```

9 i 12 (mateixa coindició):

(event==true) && (message=="BACKWARD PID")

```
setDirectionMotor(Both,Backward);
setPoint = getSetPoint();
initPIDMotor(setPoint);
```

13 :

```
Event = = false
```

```
RightRPM , LeftRPM = executePIDMotor();
```

Figura 60: FSM Programa 1 condionals/accions[63]

[62]Font: imatge pròpia
[63]Font: imatge pròpia

Programa 2 del Robot Car Kit

El programa consisteix en la captació de la variació de la velocitat de les rodes motrius quan s'aplica sobre elles un controlador PID. Cal esmentar, que aquesta aplicació captarà a temps real les velocitats actuals de les rodes motrius, cada cop que s'executa el controlador PID s'obtindrà les velocitats en RPM de les rodes de la dreta i esquerra del Robot Car Kit i s'enviaren al servidor socket intermedi. El servidor intermedi enviarà les velocitats al servidor web per la seva representació gràfica.

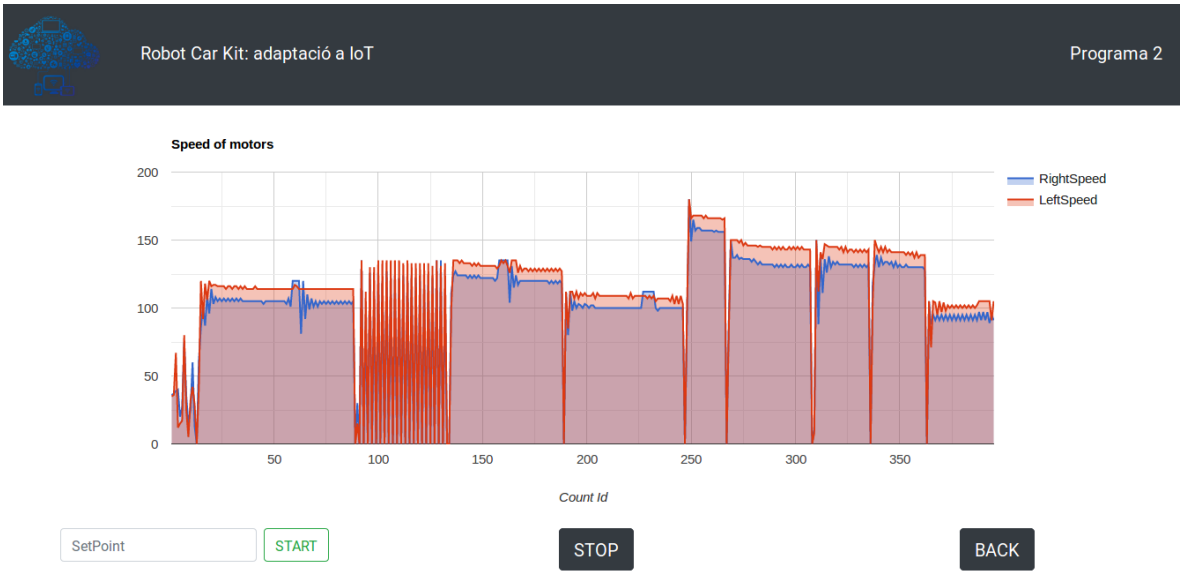

Figura 61: Representació gràfica de les valocitats en la pàgina web[64]

[64]Font: imatge pròpia

Tot seguit es presenta la màquina d'estats de tipus Mealy:

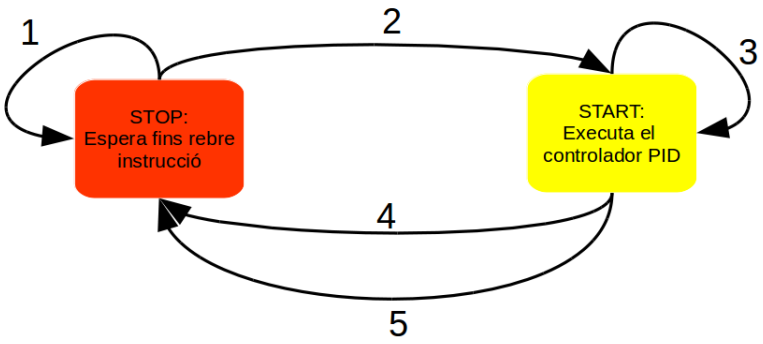

Figura 62: FSM Programa 2[65]

Les funcions emprades en la màquina d'estat corresponen a la llibreria MOTOR.

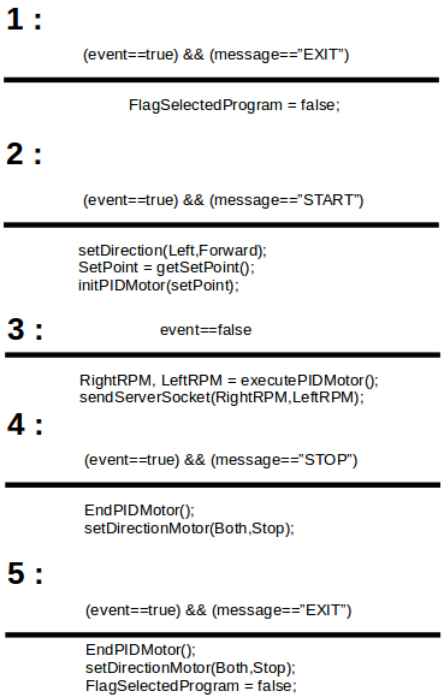

Figura 63: FSM Programa 2 condionals/accions[66]

[65]Font: imatge pròpia
[66]Font: imatge pròpia

Programa 3 del Robot Car Kit

El programa consisteix en la implementació d'un seguidor d'obstacle amb captació de la trajectòria realitzada pel Robot Car Kit. El programa s'implementarà amb el sensor d'ultrasò dirigible i el control de les rodes motrius.

Fonamentalment, s'implementarà una màquina d'estats, el qual anirà mostrejant quina distància s'obté entre el Robot Car Kit i l'objecte més pròxim. La detecció de la distància es farà amb el mòdul del sensor d'ultrasò dirigible en les posicions corresponents als graus sexagesimals 0, 45, 90, 135 i 180, si la distància és major a 5 centímetres i inferior a 25, els graus sexagesimals i la distància detectada s'enviaran al servidor soket intermedi el qual ho enviarà al servidor web. El fet d'establir els llindars de distància de l'objecte entre 5 i 25, és a causa del fet d'evitar la detecció de qualsevol objecte no proper al Robot Car Kit, com ara, les parets. Un cop, les dades de distància i graus captats s'envien al servidor web, dintre d'aquest i gràcies a les dades s'aplica un algorisme simple per tal de conèixer en quina direcció, sentit i distància s'ha desplaçat el Robot Car Kit, i poder d'aquesta forma representar la seva trajectòria sobre un pla cartesià, on el punt de partida serà X=0 i Y=0.

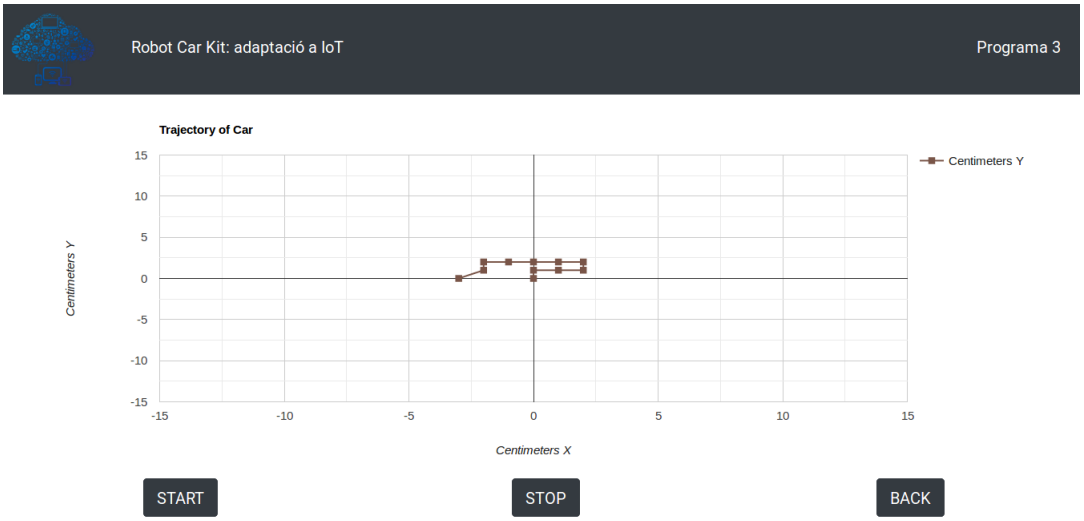

Figura 64: Vista programa 3: trajectoria robot[67]

[67]Font: imatge pròpia

L'algorisme consisteix en tenir un punt de referència amb el qual marcar la direcció i sentit dels pròxims punts de desplaçament, aquest punt de referència serà la posició inicial de partida del Robot Car Kit. A més, a partir dels graus detectats determinarà el pròxim punt de la trajectòria en el pla cartesià de la següent forma:

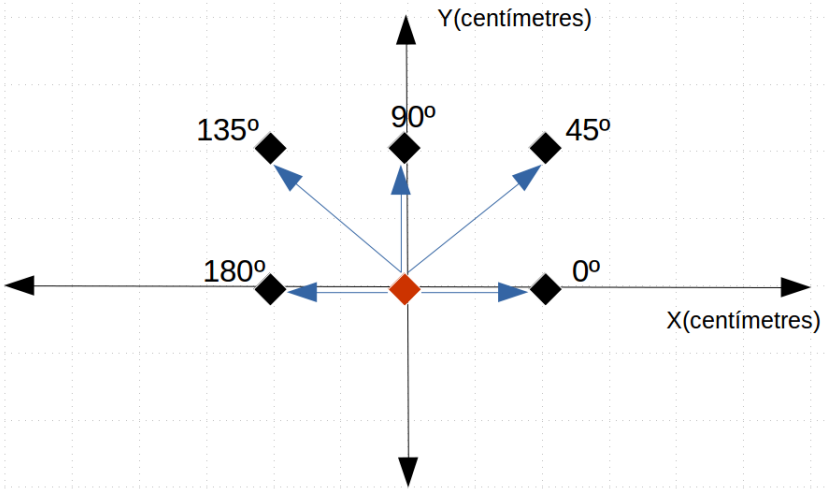

Figura 65: Pla cartesià trajectoria Robot Car Kit[68]

Respecte a la distància desplaçada, aquesta dona la informació de quantes unitats cal desplaçar-se sobre el pla cartesià en referència al punt previ. En poques paraules, s'acaba construint un vector, on la distància és el seu mòdul i els graus la seva direcció i sentit.

[68]Font: imatge pròpia

Tot seguit es presenta la màquina d'estats de tipus Mealy:

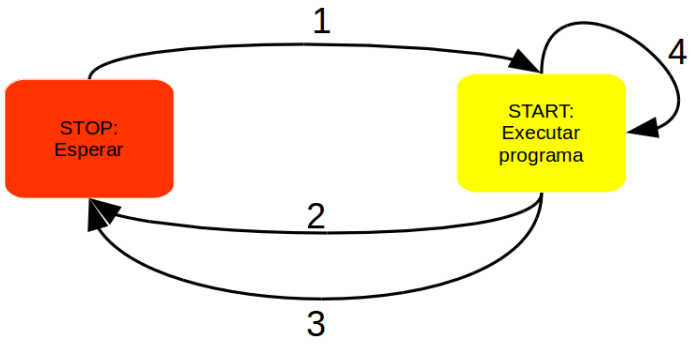

Figura 66: FSM Programa 3[69]

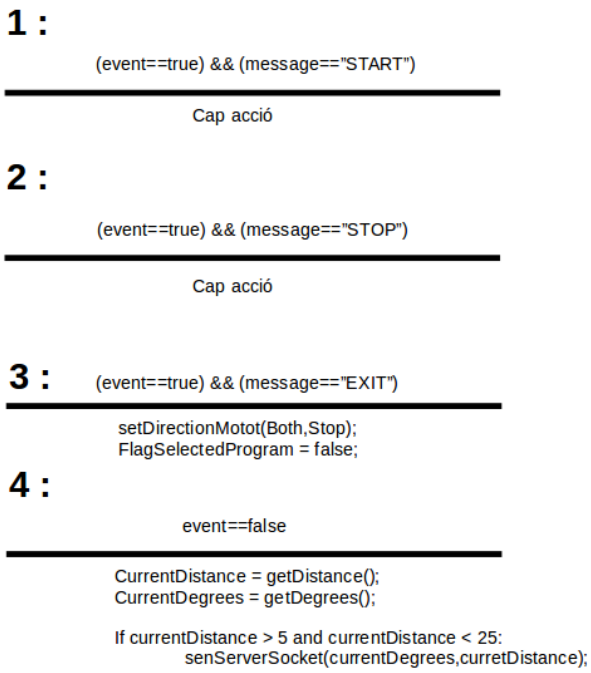

Figura 67: FSM Programa 3 condionals/accions[70]

[69]Font: imatge pròpia
[70]Font: imatge pròpia

4.2 Segona aplicació de mostra

La segona aplicació, es tracta del desenvolupament d'un servidor web en l'Arduino MKR WiFi 1010 amb la llibreria TCPSERVER, a partir del qual es controlarà un programa sobre les rodes motrius del Robot Car Kit. De forma anàloga, que en l'aplicació 1, el funcionament del sistema es basa en el fet de rebre ordres o comandes d'un client mitjançades peticions HTTP i arribar a transmetre'ls al Robot Car Kit.

La seqüència de treball de l'aplicació és la següent:

1. Un client HTTP, enviarà les comandes pel camp de *query string* d'un URL i pel mètode GET de la següent forma:
   http://host:port/api/sendMessage?message=COMMAND

   En aquest cas, s'utilitzarà el programa POSTMAN[comg] com client HTTP i poder realitzar les peticions.

2. En l'Arduino MKR s'executarà una màquina d'estats corresponent al servidor web. Aquesta màquina extreurà la informació del paràmetre *message* quan la petició sigui atesa.

   Pel que fa a la màquina d'estats, aquesta espera rebre el missatge *INIT* per inicialitzar el programa associat a les rodes motrius i tot seguit executar-lo.

3. Un cop es començar a executar el programa de les rodes motrius, tot missatge provinent per la petició HTTP es passarà directament en aquest programa, fins a rebre l'obre de *exit*, el qual tancarà el programa. Aquest programa és el mateix que el *programa 1* de l'aplicació 1.

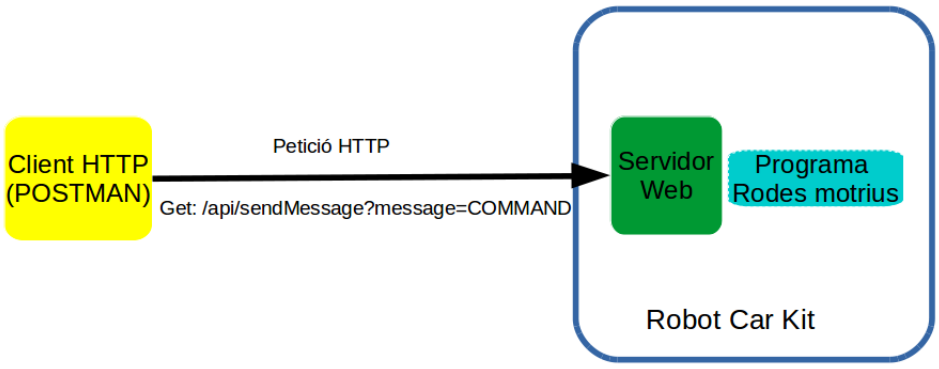

Figura 68: Xarxa de comunicació de l'aplicació 2[71]

[71]Font: imatge pròpia

4.2.1 Màquina d'estats servidor web

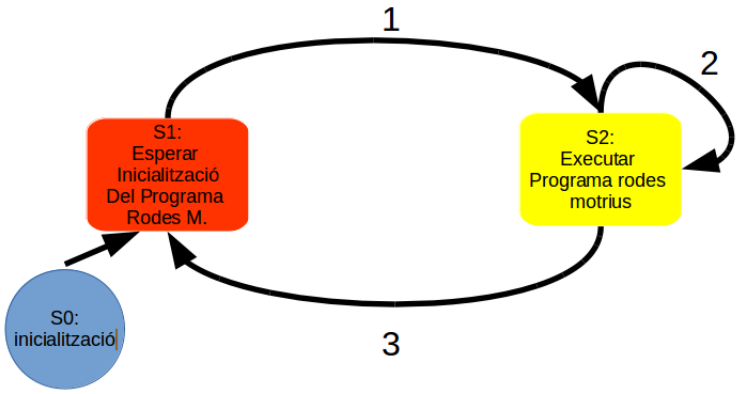

Figura 69: Màquina d'estats Servidor Web[72]

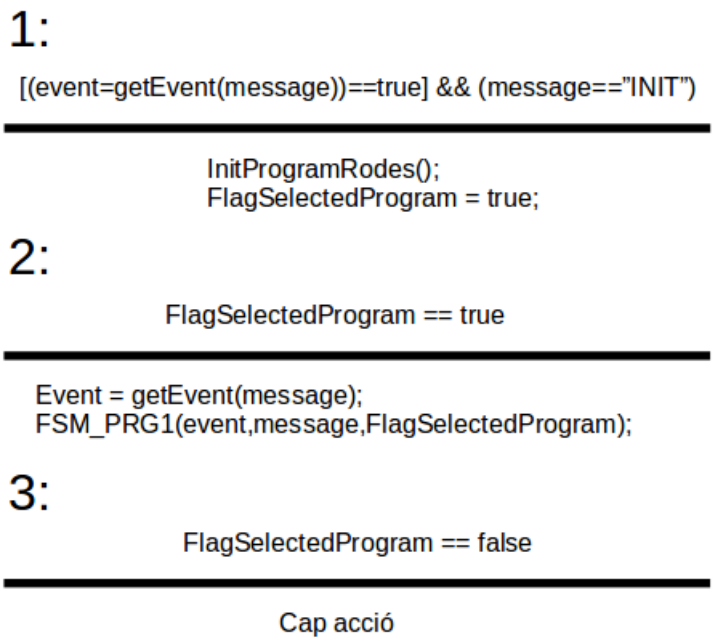

Figura 70: Màquina d'estats Servidor Web- condicionals/acció[73]

[72]Font: imatge pròpia
[73]Font: imatge pròpia

Descripcións de funcions i variables de la màquina d'estat:

- **message :**
  Variable on emmagatzemar el missatge provinent del client HTTP en el paràmetre *message*.

- **FlagSelectedProgram :**
  La variable indica si hi ha algun programa seleccionat, aquesta variable serà compartida amb la màquina d'estat del programa de les rodes motrius. Quan la màquina d'estats del programa seleccionat modifiqui el valor de la variable a false, el programa deixarà d'executar-se.

- **getEvent(message) :**
  La funció retorna true en cas que hi hagi missatges disponibles provinents del client, a més, si es dóna aquest cas, el missatge es desarà en la variable *message*, en cas contrari, retorna false.

- **FSM_PRG1(event, message, FlagSelectedProgram) :**
  Representa la crida a la màquina d'estats del programa de les rodes motrius. Es passen les variables event, message i FlagSelectedProgram, perquè la màquina corresponent pugui coneixer quan hi ha un missatge disponible i el seu valor, i a més, poder modificar la variable FlagSelectedProgram quan es decideixi acabar l'execució del programa.

4.2.2 Màquina d'estats programa rodes motrius

El programa consisteix en el control remot de les rodes motrius amb diferents funcionalitats. Principalment, es tractarà d'una màquina d'estats responsable de posar en marxa funcionalitats com el control de la velocitat de les rodes amb un PID o poder desplaçar-se una distància concreta en centímetres de forma automàtica.

Llistat de funcionalitats/ordres disponibles:

- **Stop:** Parar els motor del Robot Car Kit.
- **Forward PID:** Posa en marxa els motors amb una velocitat RPM de referència i executar un controlador PID sobre aquests. En concret, el sentit de gir de les rodes farà avançar al Robot Car Kit.
- **Backward PID:** Posa en marxa els motors amb una velocitat RPM de referència i executar un controlador PID sobre aquests. En concret, el sentit de gir de les rodes farà retrocedir al Robot Car Kit.
- **Rotate:** Fa rotar el Robot Car Kit a la dreta o l'esquerra durant un centímetre de desplaçament.
- **Rotate Car:** Fa rotar el Robot Car Kit a la dreta o l'esquerra donat un determinat valor de graus sexagesimals. Els valors acceptats són 90, 180, 270 i 360.
- **Forward Centimeters:** Desplaça el Robot Car Kit endavant donat un valor de desplaçament en centímetres. Els valors acceptats van del 1 fins al 255.
- **Backward Centimeters:** Desplaça el Robot Car Kit endarrere donat un valor de desplaçament en centímetres. Els valors acceptats van del 1 fins al 255.
- **Power Rotation:** Fa variar la potència de rotació de les rodes sobre les comandes Rotate, Rotate Car, Forward Centimeters i Backward Centimeters. Els valors acceptats van del 1 fins al 255.
- **Exit:** Provoca el fi d'execució del programa, modificant la variable FlagSelectedProgram a false.

Les funcions emprades en la màquina d'estat corresponen a la llibreria MOTOR.

Tot seguit es presenta la màquina d'estats de tipus Mealy:

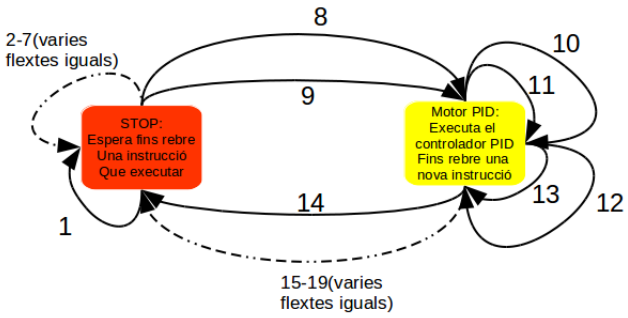

Figura 71: FSM Programa 1 [74]

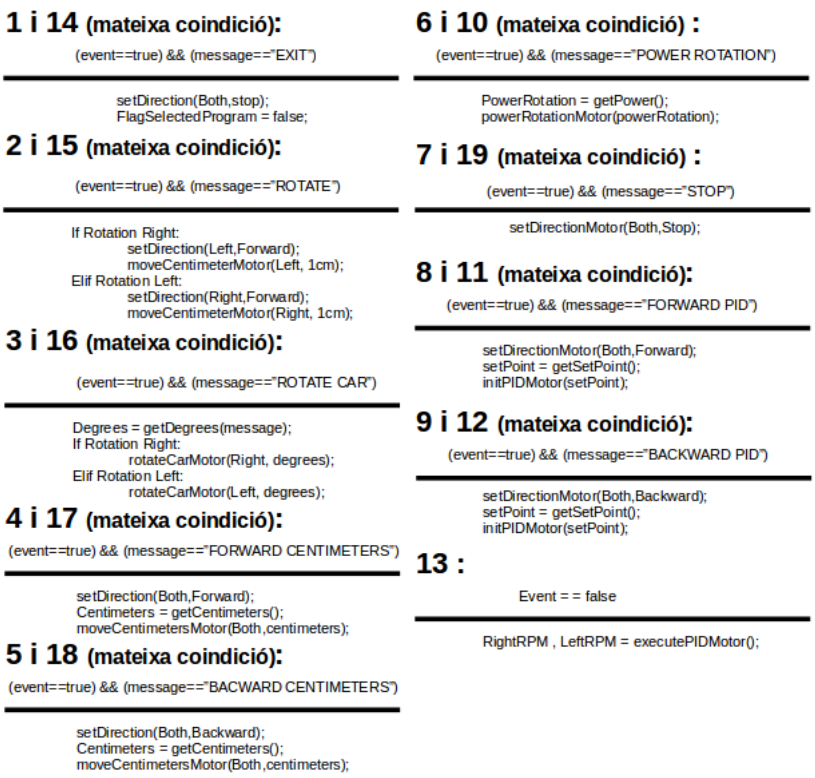

Figura 72: FSM Programa 1 condionals/accions[75]

[74]Font: imatge pròpia
[75]Font: imatge pròpia

5 Conclusions

Per concloure, aquest projecte m'ha aportat personalment poder conèixer i aprofundir sobre el funcionament i la programació de diversos components electrònics, també, m'hi ha permès ampliar els meus coneixements sobre les funcionalitats que ofereix un mòdul WiFi i poder treballar amb el microcontrolador Arduino MKR WiFi 1010 m'ha permès ampliar tant els meus coneixements com en experiència sobre aquests.

En concret, ha estat d'utilitat poder conèixer més en profunditat i experimentar un mètode d'implementació de llaç de control sobre la velocitat i posicionament en motors de corrent contínua amb el driver L298N i els encoder de quadratura SJ01. Així mateix, he pogut observa un cas pràctic d'ús dels senyals PWM, el qual és una funcionalitat comuna en la majoria de microcontroladors. També, he pogut aprofundir en el funcionament dels encoders de quadratura i les seves característiques que ens ofereix, a fi de poder conèixer les velocitats assolides pels motors o conèixer el sentit de gir a partir del codi Gray que generen els senyals provinents d'aquest.

Per finalitzar, i respecte al Robot Car Kit i la seva adaptació a la Internet de les Coses (IoT),m'hi ha suposat interessant i profitós el repte de cerca els mètodes més adequats per dur a terme aquesta tasca. En concret, per una banda, he observat la utilitat i versatilitat que ofereix tenir una via o capa de comunicació entre un sistema electrònic i una xarxa WiFi de cara a poder gestionar el seu control o l'adquisició de dades de forma simple, i per altra banda, m'ha permès apendre en detall les comunicacions per missatges HTTP.

6 Referències

•

[coma] Arduino's company. *AnalogWrite()*. URL: https://www.arduino.cc/reference/en/language/functions/analog-io/analogwrite/. (accessed: 28.01.2019).

[comc] Arduino's company. *pulseIn()*. URL: https://www.arduino.cc/reference/en/language/functions/advanced-io/pulsein/. (accessed: 28.01.2019).

[comd] Arduino's company. *WiFiNINA library*. URL: https://www.arduino.cc/en/Reference/WiFiNINA. (accessed: 28.01.2019).

[come] DFRobot's company. $Micro_DC_Motor_with_Encoder - SJ01_SKU_FIT0450$. URL: https://wiki.dfrobot.com/Micro_DC_Motor_with_Encoder-SJ01_SKU__FIT0450. (accessed: 28.01.2019).

[comi] Wikipedia's company. *Cliente-servidor*. URL: https://es.wikipedia.org/wiki/Cliente-servidor. (accessed: 28.01.2019).

[comj] Wikipedia's company. *Internet protocol suite*. URL: https://en.wikipedia.org/wiki/Internet_protocol_suite. (accessed: 28.01.2019).

[comk] Wikipedia's company. *Modelo–vista–controlador*. URL: https://es.wikipedia.org/wiki/Modelo%E2%80%93vista%E2%80%93controlador. (accessed: 28.01.2019).

[coml] Wikipedia's company. *Socket de Internet*. URL: https://es.wikipedia.org/wiki/Socket_de_Internet. (accessed: 28.01.2019).

[Hir] Timothy Hirzel. *PWM*. URL: https://www.arduino.cc/en/Tutorial/PWM. (accessed: 28.01.2019).

[mdna] mdnwebdocs-bot. *Access-Control-Allow-Origin*. URL: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Access-Control-Allow-Origin. (accessed: 28.01.2019).

[mdnb] mdnwebdocs-bot. *Mensajes HTTP*. URL: https://developer.mozilla.org/es/docs/Web/HTTP/Messages. (accessed: 28.01.2019).

[rob] Andromina robot. *encoder-de-cuadratura-y-arduino*. URL: http://androminarobot.blogspot.com/2016/08/encoder-de-cuadratura-y-arduino.html. (accessed: 28.01.2019).

7 Bibliografia

•

[comb] Arduino's company. *ARDUINO MKR WIFI 1010*. URL: `https://store.arduino.cc/mkr-wifi-1010`. (accessed: 28.01.2019).

[comf] Elegoo's company. *Elegoo UNO Project Smart Robot Car Kit V 3.0 with UNO R3*. URL: `https://www.elegoo.com/product/arduinocarv3-0/`. (accessed: 28.01.2019).

[comg] Postman's company. *Postman*. URL: `https://www.getpostman.com/`. (accessed: 28.01.2019).

[comh] Ratchet's company. *Ratchet Documentation*. URL: `http://socketo.me/`. (accessed: 28.01.2019).

[Otw] Taylor Otwell. *Laravel Documentation*. URL: `https://laravel.com/docs/5.8`. (accessed: 28.01.2019).

8 Llista d'acrònims

- AI: Artificial Inteligence.
- API: Application Programming Interface.
- ASCII: American Standard Code for Information Interchange.
- DC: Direct Current.
- FSM: Finite State Machine.
- HTTP: Hypertext Transfer Protocol.
- IoT: Internet of Things.
- IP: Internet Protocol.
- MVC: Model–View–Controller.
- PC: Personal Computer.
- PID: Proportional–Integral–Derivative (controller).
- PWM: Pulse-Width Modulation.
- RPM: Revolutions Per Minute.
- SSID: Service Set Identifier.
- TCP: Transmission Control Protocol.
- TTL: Transistor–Transistor Logic.
- UDP: User Datagram Protocol.
- URL: Uniform Resource Locator.
- WAP: Wireless Application Protocol.
- WEP: Wired Equivalent Privacy.
- WWW: World Wide Web.

9 Annexos

9.1 Esquema circuital final del Robot Car Kit i l'Arduino MKR per les aplicacions de mostra

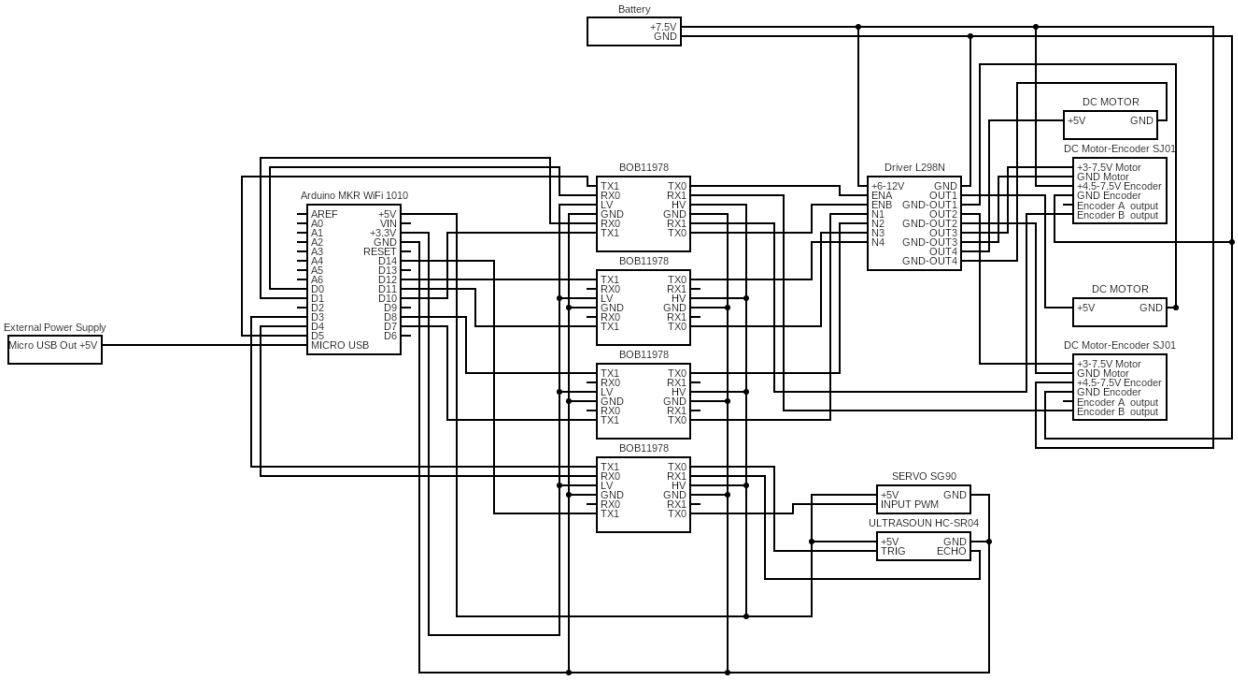

Figura 73: Esquema circuital de connexions[76]

[76]Font: imatge pròpia