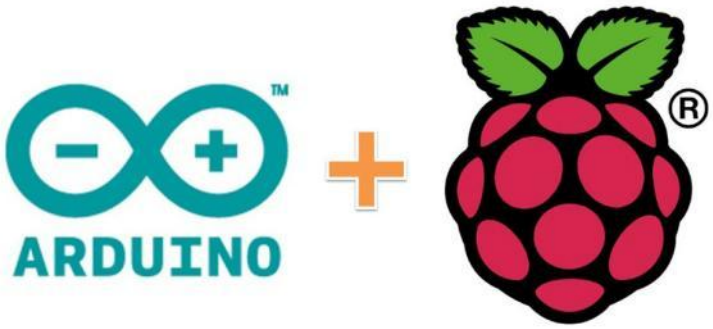
# Treball Fi de Carrera

# Arduino + Raspberry Pi.
## Vehicle RC d'exploració.





| | |
|---|---|
| **ALUMNE:** | **Francisco José Ballesta Torre** |
| **CONSULTOR:** | **Antoni Morell Pérez** |
| **UNIVERSITAT:** | **Universitat Oberta de Catalunya (UOC).** |
| **TITULACIÓ:** | **Enginyeria Tècnica de Telecomunicacions, Telemàtica.** |
| **ÀREA TFC:** | **Integració de xarxes telemàtiques.** |
| **TEMA TFC:** | **Sistemes de comunicació per a RadioControl (RC).** |

# Índex

## 1.- Descripció general

Actualment existeixen nombroses tecnologies que es poden utilitzar pel món del modelisme teledirigit, ja no només pel que fa a les freqüències de treball sinó al vehicle al que se li aplica aquesta tecnologia i al comandament de control que es fa servir.

Des de sempre, la base dels sistemes de modelisme teledirigit i sistemes de radio control en general, s'ha basant en controlar un vehicle (cotxes, avions, helicòpters, vaixells, etc.) mitjançant sistemes de RF (radio freqüència) i un comandament amb exclusiu per aquest ús. Però actualment, existeixen tecnologies com el Bluetooth, WiFi o ZigBee, que, juntament amb plaques de control com Arduino, permeten substituir els comandaments habituals, per altres dispositius més comuns com ara Smartphones, Tablets o qualsevol PC/Portàtil.

Aquest fet, ha originat que cada cop més hi hagi més afició i interès en el món de l'electrònica, la qual es la base fonamental dels sistemes de radio control, i d'aquesta manera, "faciliti" la creació de dispositius teledirigits "fets a mida" i escalables segons les necessitats o voluntats de cadascú.

Per tots aquests motius exposats anteriorment, he optat per fer un projecte en el que s'implantaran aquestes "noves" tecnologies en un sistema radio control. Concretament, el projecte estarà dividit en dues fases, una part teòrica i una altra purament pràctica en la que es fabricarà un dispositiu per radio control totalment "personalitzat".

De forma introductòria, en el projecte es realitzarà un estudi sobre les diferents tecnologies existents en els sistemes radio control més habituals, a fi d'entendre el funcionament de cadascuna, els avantatges i inconvenients, i la utilitat que se'ls hi pot donar en el món actual.

Més endavant, i amb la finalitat d'obtenir els coneixements necessaris per encarar la part pràctica del projecte, s'analitzarà el funcionament i aplicacions dels mòduls electrònics Open Source Arduino i Raspberry Pi. Un cop fet això, s'intentarà dissenyar un vehicle teledirigit, fent servir una placa Arduino per tal de controlar els motors i els diferents sensors que incorporarà el vehicle (Temperatura, humitat, etc.), i a part, també comptarà amb un dispositiu Raspberry Pi, el qual tindrà varies funcions, ja que servirà com a pont de comunicació entre el dispositiu de comandament (PC, smatphone, etc.) i el Arudino pel moviment del vehicle, realitzarà l'emmagatzematge de les dades dels diferents sensors, i a més, incorporarà una petita càmera per tal de poder controlar el vehicle inclús sense tenir visió directe amb ell.

# 2.- Objectius

Aquest projecte, te dos objectius bàsics. El primer, és conèixer i comparar els diferents sistemes de comunicació que s'utilitzen avui en dia en el món del modelisme teledirigit, i el segon, idear i desenvolupar un dispositiu controlat a distancia que utilitzi un sistema de radio control via WiFi.

De forma més detallada, els objectius que s'intentaran assolir en les dues parts principals del projecte, son els següents:

## Part 1: Estudi dels sistemes de comunicació per a radio control existents.

- Explicar de forma senzilla, el funcionament dels bàsic dels diferents sistemes de comunicació RC existents.
- Realitzar una comparativa objectiva dels diferents sistemes de comunicació, segons els avantatges i inconvenients que tenen cadascun.
- Enumerar algunes de les aplicacions actuals dels diferents sistemes de comunicació RC actuals.

## Part 2: Disseny sistema RC d'exploració.

- Analitzar les possibilitats que ens donen els dispositius Arduino i Raspberry Pi i aprendre els conceptes bàsics.
- Fabricar un vehicle de dimensions reduïdes, que analitzi, mitjançant sensors connectats a un Arduino, la temperatura i humitat de l'ambient.
- Interconnectar un Arduino amb una Raspberry Pi, per tal que aquesta última, emmagatzemi les dades dels sensors i permeti accedir-hi via web.
- Controlar el vehicle remotament mitjançant una xarxa WiFi, i poder veure el camí pel que passa el vehicle, mitjançant una càmera connectada a la Raspberry Pi.

## 3.- Planificació (Diagrama de Gantt)

| | Modo de | Nombre de tarea | Duración | Comienzo | Fin | Predecesoras |
|---|---|---|---|---|---|---|
| 1 | | **Projecte TFC - Integració de xarxes telemàtiques** | **93 días** | **mié 17/09/1** | **vie 23/01/15** | |
| 2 | | Elecció projecte TFC | 6 días | mié 17/09/1 | mié 24/09/14 | |
| 3 | | Planificació projecte | 5 días | jue 25/09/14 | mié 01/10/14 | 2 |
| 4 | | Entrega PAC1 | 0 días | mié 01/10/1 | mié 01/10/14 | |
| 5 | | **Execució projecte** | **70 días** | **jue 02/10/14** | **mié 07/01/15** | **3** |
| 6 | | *Estudi sistemas RC habituals* | **7 días** | **jue 02/10/14** | **vie 10/10/14** | |
| 7 | | *Conceptes bàsics freqüències RC* | 4 días | jue 02/10/14 | mar 07/10/14 | |
| 8 | | *Quadre comparatiu freqüències* | 2 días | mié 08/10/1 | jue 09/10/14 | 7 |
| 9 | | *Aplicacions freqüències RC* | 1 día | vie 10/10/14 | vie 10/10/14 | 8 |
| 10 | | *Disseny sistema radio control propi* | **63 días** | **lun 13/10/14** | **mié 07/01/15** | **6** |
| 11 | | *Estructura vehicle* | 2 días | lun 13/10/14 | mar 14/10/14 | |
| 12 | | *Integració Arduino* | **5 días** | **mié 15/10/1** | **mar 21/10/14** | **11** |
| 13 | | Connexionat Motors | 3 días | mié 15/10/1 | vie 17/10/14 | |
| 14 | | Connexionat Sensors | 2 días | lun 20/10/14 | mar 21/10/14 | 13 |
| 15 | | *Integració Raspberry Pi* | 2 días | mié 22/10/1 | jue 23/10/14 | 12 |
| 16 | | *Connexió Arduino <--> Raspberry Pi* | 2 días | vie 24/10/14 | lun 27/10/14 | 15 |
| 17 | | *Sistema de control* | 10 días | mar 28/10/1 | lun 10/11/14 | 16 |
| 18 | | Entrega PAC 2 | 0 días | lun 10/11/14 | lun 10/11/14 | |
| 19 | | *Telemetria* | **27 días** | **mar 11/11/1** | **mié 17/12/14** | **17** |
| 20 | | *Emmagatzematge* | 7 días | mar 11/11/1 | mié 19/11/14 | |
| 21 | | *Visualització "RealTime"* | 20 días | jue 20/11/14 | mié 17/12/14 | 20 |
| 22 | | Entrega PAC 3 | 0 días | mié 17/12/1 | mié 17/12/14 | |
| 23 | | *Streaming càmera On-Board* | 15 días | jue 18/12/14 | mié 07/01/15 | 19 |
| 24 | | Conclusions i valoració final | 4 días | jue 08/01/15 | mar 13/01/15 | 5 |
| 25 | | Entrega Memoria Final | 0 días | mar 13/01/1 | mar 13/01/15 | |
| 26 | | Presentació Final | 3 días | mié 14/01/1 | vie 16/01/15 | 24 |
| 27 | | Entrega Presentació | 0 días | dom 18/01/1 | dom 18/01/15 | |
| 28 | | Inici del Tribunal | 0 días | lun 19/01/15 | lun 19/01/15 | |
| 29 | | Final del Tribunal | 0 días | vie 23/01/15 | vie 23/01/15 | |

Figura 1. Llistat de tasques del projecte.

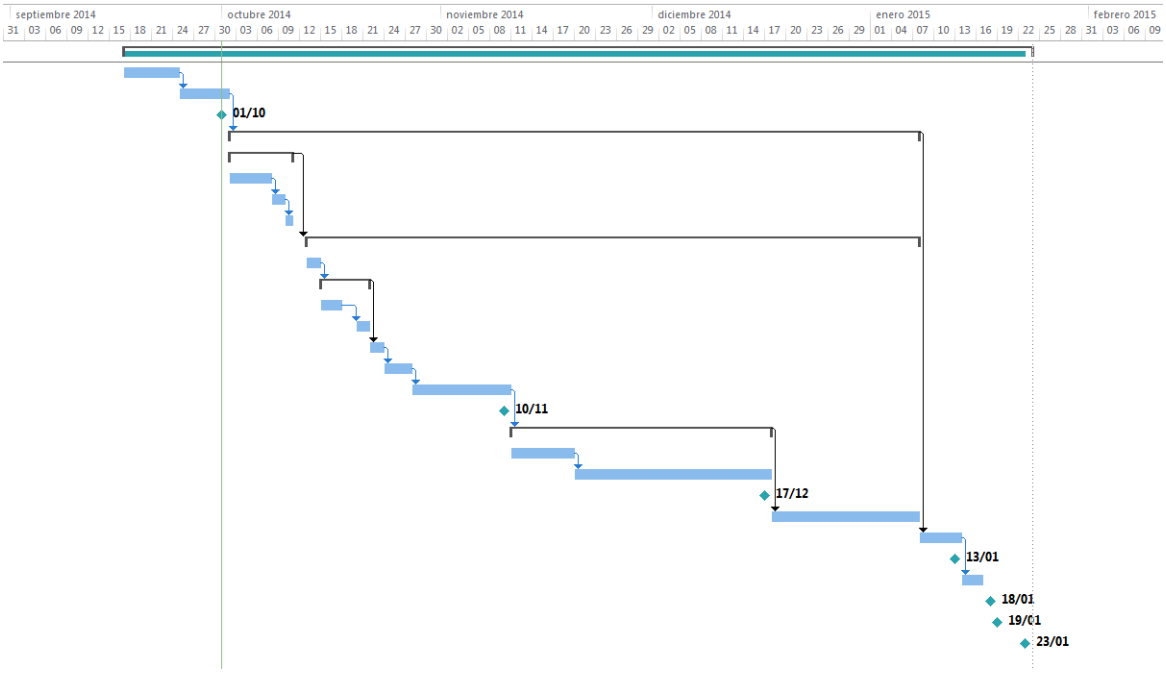



Figura 2. Diagrama de Gantt.

# 4.- Sistemes radio control habituals

## 4.1.- Conceptes bàsics freqüències RC

En el món dels sistemes radio control actuals, alguns d'ells pràcticament en desús, podem trobar que, bàsicament, es fan servir tres tecnologies o tècniques diferents per tal d'enviar les senyals des de les emissores de control fins al receptor ubicat al vehicle, sigui quin sigui el seu medi (aquàtic, terrestre o aeri). Aquests tres sistemes de comunicació son, AM, FM i espectre eixamplat.

### 4.1.1.- AM

El sistema de comunicació AM (Amplitud modulada), es tracta d'un dels mes utilitzats històricament, però no només en els vehicles RC, sinó en molts altres àmbits com la radio.

Es tracta d'una tècnica la qual mante estable la freqüència de treball, modulant l'amplitud de la senyal portadora en funció de la informació que s'està transmitent.

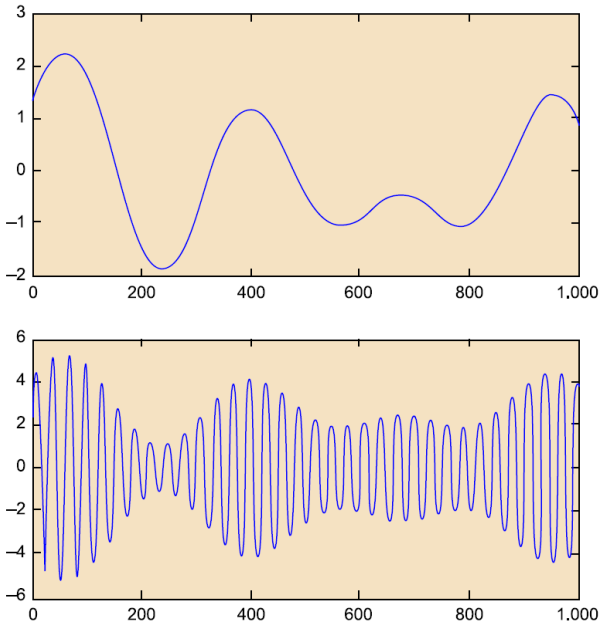



Figura 3. Senyal AM (Font: Apunts de l'assignatura Transmissió Digital - UOC)

#### Avantatges

- La seva desmodulació és relativament senzilla el que simplifica el disseny i cost dels equips de recepció.
- Bona cobertura d'emissió.

#### Inconvenients

- Baixa qualitat de senyal.
- Ample de banda molt limitat.
- La senyal està molt exposada a interferències d'altres aparells o de fenòmens atmosfèrics.

### 4.1.2.- FM

Per la seva banda, els sistemes FM (Freqüència modulada), també han sigut i son actualment, molt populars i utilitzats com en el cas de AM.

En aquest cas, la freqüència de la senyal portadora no es manté constant, sinó que es va variant la freqüència en funció de la informació que s'està enviant.

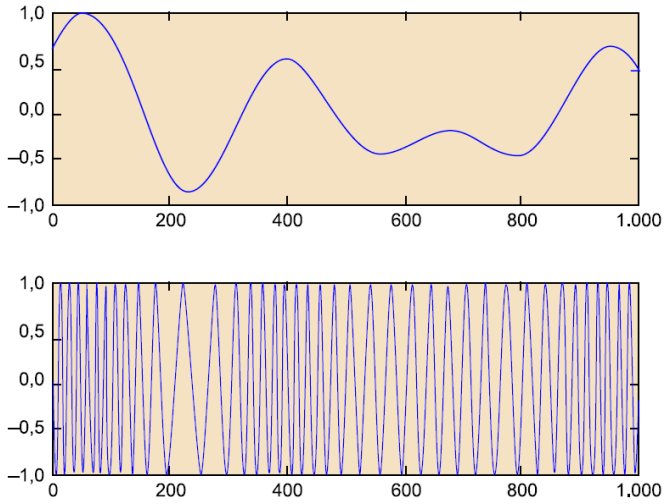



**Figura 4. Senyal FM (Font: Apunts de l'assignatura Transmissió Digital - UOC)**

#### Avantatges

- Mayor ample de banda que la modulació AM.
- Bona qualitat de senyal.
- A la modulació FM no li afecten les interferències que empitjoraven les senyals en AM.

#### Inconvenients

- Cobertura d'emissió més limitada que en AM.
- La senyal viatja en línia recta, per tant, es veu molt perjudicada pels obstacles.

### 4.1.3.- Espectre Eixamplat

Mentre que, tant AM com FM son tècniques de modulacions analògiques, la tècnica d'espectre eixamplat és una forma de modulació digital. Concretament, hi ha dos tècniques diferents en la modulació d'espectre eixamplat, **FHSS** (Frequency Hopping) i **DSSS** (Direct Sequence).

Es tracta del sistema més modern, però que ja està molt estès degut a que es aplicable a una gran quantitat de funcionalitats diferents (WLAN, RC, GPS, Bluetooth, telèfons sense fils, etc.). Fa servir la banda freqüencial dels 2,4 GHz.

#### Avantatges

- Sense interferències. Disposa d'un sistema per treballar en la freqüència que estigui lliure, ja que prèviament al enllaç de la comunicació realitza un escaneig de l'entorn.
- Molt bona qualitat de senyal.
- Gran ample de banda.
- Capacitat per travessar obstacles degut a les altes freqüències que fa servir.
- Baix consum energètic.

#### Inconvenients

- Disseny complex i cost elevat de les emissores i receptors, en comparació amb els de AM i FM.

## 4.2.- Aplicacions de freqüències en vehicles RC

Un cop s'ha vist els conceptes bàsics, i els avantatges i inconvenients, dels tres sistemes de comunicació diferents (AM, FM i espectre eixamplat), es poden extreure una sèrie de conclusions pel que respecta a les utilitzacions de cada sistema segons les seves virtuts.

- **Modelisme aeri:** Ja sigui en avions, helicòpters o "drones", lo més òptim en el cas del aeromodelisme, seria, i des de ja fa bastant de temps és la tendència, utilitzar emissores d'espectre eixamplat. El motiu principal, radica en que es tracta del sistema més fiable pel que fa a les afectacions per interferències, i en aquest àmbit, les interferències poden provocar la pèrdua de control i l'accident posterior del vehicle controlat.

- **Modelisme terrestre i aquàtic:** Pel que fa a aquestes dues disciplines de modelisme, tenint en compte que la pèrdua momentània del control degut a les interferències no és tant crític com en el cas del aeromodelisme, va fer que fins fa pocs anys enrere, encara s'estiguessin fent servir les emissores AM i FM, però degut a que tots els avantatges que disposen les emissores d'espectre eixamplat i que el seu preu s'ha anat reduint en aquests últims anys, avui en dia ja és habitual que sigui quin sigui el vehicle controlat (aeri, marítim o terrestre), faci servir transmissions a 2,4 GHz.

# 5.- Material utilitzat: Conceptes bàsics

En aquest apartat del projecte, em centraré en dissenyar un vehicle controlat a distancia mitjançant una xarxa WiFi.

Per a aquest projecte, he escollit un vehicle terrestre, encara que moltes de les configuracions i dissenys que realitzaré durant el projecte, serien aplicables a un altre tipus de vehicle (aeri o naval) simplement canviant les ordres del sistema de control de motors.

## 5.1.- Estructura vehicle

### 5.1.1.- Xassís

El primer que s'ha de que fer, és muntar o fabricar l'estructura bàsica del vehicle. Actualment, per Internet, és fàcil trobar kits que ja porten el xassís del vehicle, els motors i les rodes, preparats per muntar. Encara que, hi ha d'altres opcions, com podria ser utilitzar la base d'un cotxe teledirigit de joguina/modelisme, o fabricar un mateix l'estructura amb fusta o metall.

Figura 5. Peces del kit de vehicle RC.

En el meu cas, he fet servir un kit preparat el qual disposa de 4 rodes, amb els seus motors corresponents, una placa controladora de motors L298D i el xassís del vehicle. Aquest xassís, disposa de dos nivells, i això m'ha permès tenir espai suficient per instal·lar totes les peces del vehicle d'exploració. Concretament, la distribució que he fet és la següent:

**Nivell 0 (part inferior):** Interruptor general i motors.
**Nivell 0 (part superior):** Controladora de motors, Arduino i protoboard's de connexió.
**Nivell 1:** Raspberry Pi, càmera, bateria dels motors, bateria Rasberry Pi + Arduino i sensors.

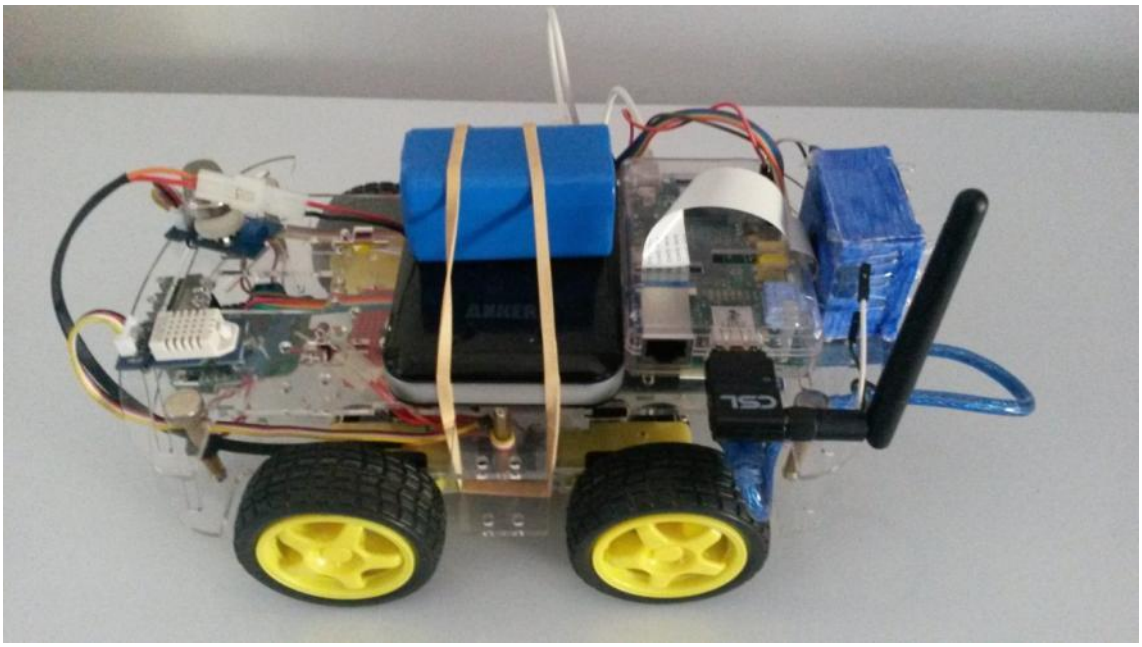


Figura 6. Vehicle RC complet. Vista lateral.

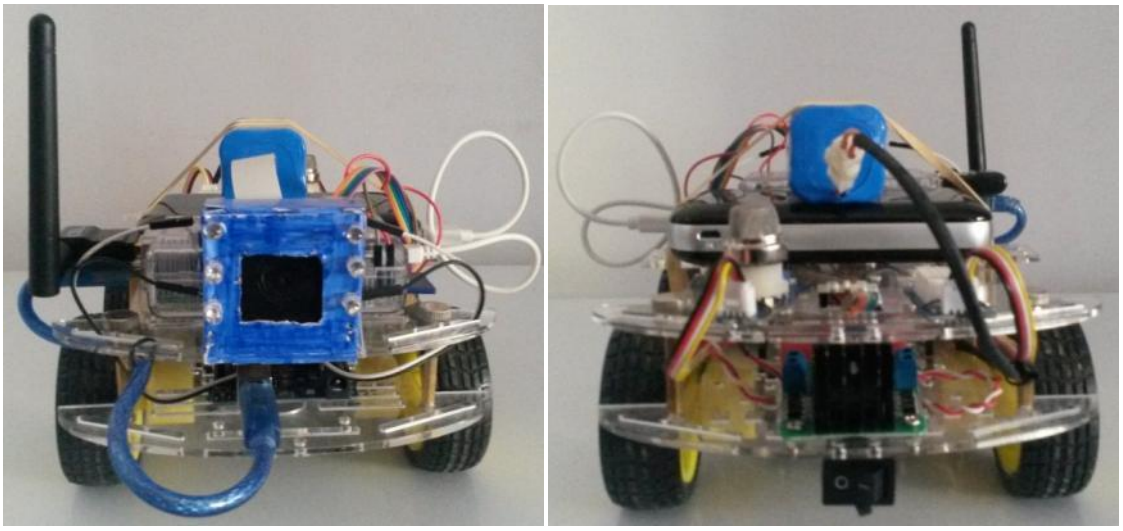


Figura 7. Vehicle RC complet. Vista frontal i posterior.

### 5.1.2.- Controladora Motors

Per tal de poder controlar la velocitat dels diferents motors, i poder invertir el sentit de les rodes per realitzar els girs o anar marxa enrere, serà necessari incorporar una placa de control de motors (driver).

Aquesta controladora, model L298D, serà l'encarregada de gestionar els quatre motors del vehicle en funció de les ordres rebudes des de l'Arduino.

Figura 8. Controladora motors L298D .

### 5.1.3.- Sistema d'alimentació

Segons el fabricant, la controladora de motors L298D te que ser alimentada amb una tensió dintre del rang +5V / +36V aproximadament. Per aquest motiu, he escollit una bateria (reutilitzada d'un robot de neteja domèstic) que genera 16V de corrent continua aproximadament.

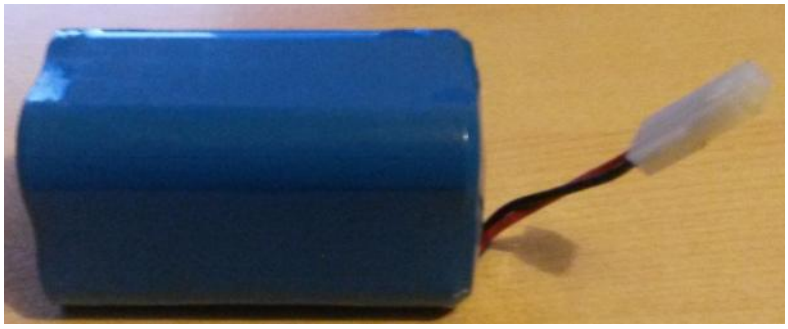


Figura 9. Bateria sistema motor.

Com es pot veure en la figura 30, també s'ha instal·lat un interruptor en la part del darrere del vehicle, per tal de poder tallar la corrent de la bateria fàcilment. Ja que el interruptor S1 de la controladora, no quedava gaire accessible.

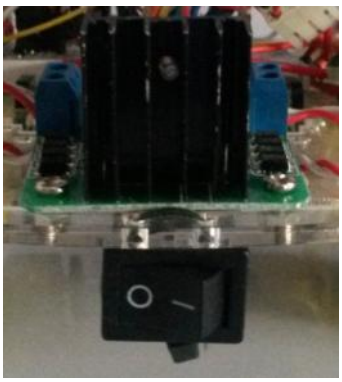


Figura 10. Interruptor sistema motors.

## 5.2.- Arduino

### 5.2.1.- Conceptes bàsics

Arduino es una plataforma de maquinari obert. A grans termes, es tracta d'una placa electrònica amb un microcontrolador, el qual, gestiona una sèrie d'entrades i sortides, ja siguin digitals i/o analògiques, mitjançant un codi que prèviament s'haurà creat fet servir l'entorn de desenvolupament corresponent.

Aquest entorn de desenvolupament és molt senzill, ja que s'ha intentat que gent sense gaires coneixements de programació i informàtics en general, el pogués fer servir. Concretament, aquest entorn de desenvolupament fa servir un llenguatge de programació anomenat Processing/Wiring, el qual està basat en el llenguatge Java.

Existeixen un gran nombre de models de plaques Arduino en el mercat, com son:

- Arduino MEGA 2560
- Arduino UNO
- Arduino Leonardo
- Arduino DUE
- Arduino Micro
- Arduino Mini

I moltes altres més, les quals son ampliacions d'unes altres, afegint-li característiques addicionals com poden ser ports USB, ethernet, WiFi, slots SD, etc.

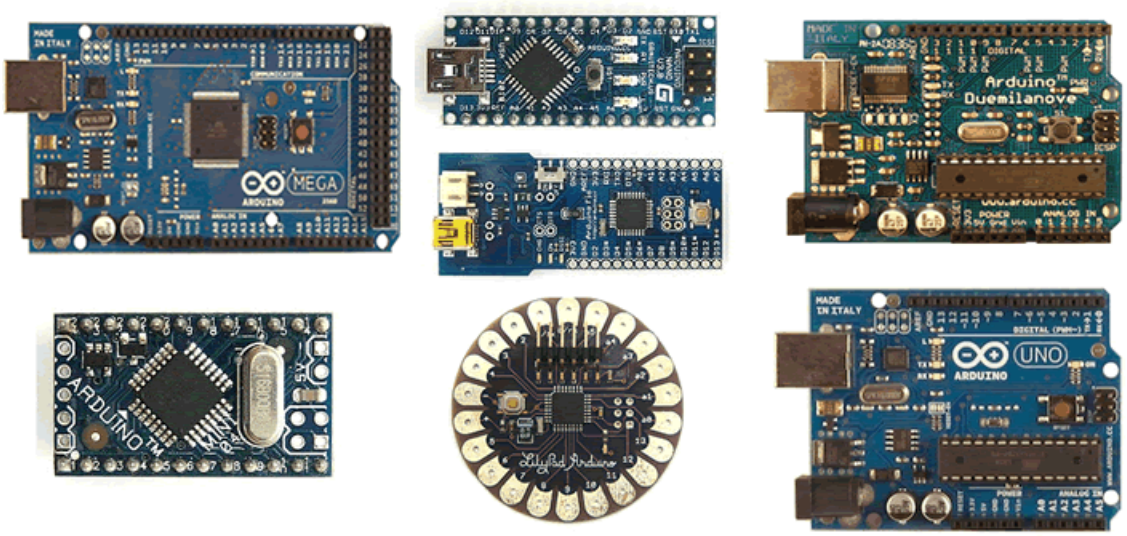



Figura 11. Plaques Arduino.

Tenint en compte tot aquest ventall de plaques, és imprescindible tenir clar quines seran les necessitats del projecte a realitzar abans de decantar-se per una o una altra.

### 5.2.2.- Accessoris Arduino

Hi ha una infinitat d'accessoris disponibles per les plaques Arduino. Aquests accessoris poden ser:

**SHIELD's:** Son unes plaques d'extensions que tenen una finalitat concreta: GSM, GPS, WiFi, ethernet, RFID, XBee, etc.

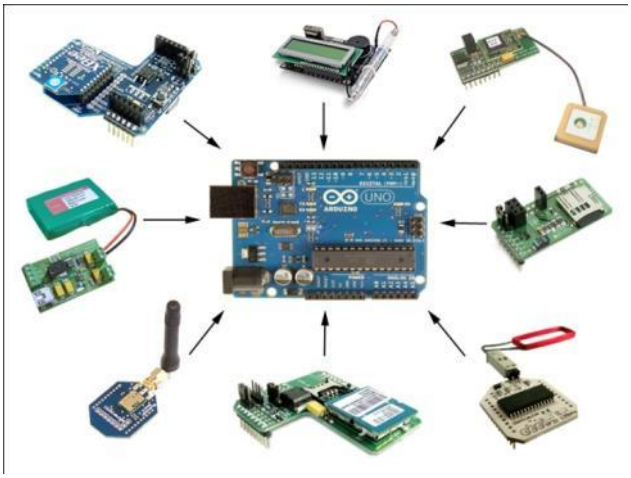


Figura 12. Shield's Arduino.

Figura 13. Mòduls i sensors Arduino.

**Mòduls sensors i control:** Encara que no son dissenyats en exclusiva per Arduino, son accessoris que habitualment es venen en kit amb plaques UNO per exemple. Hi ha de tot tipus: temperatura, humitat, gas, fum, llum, moviment, obstacles, teclats, displays, relés, RFID, i molts més.

**Components electrònics:** Evidentment, per fer funcionar tots aquests mòduls i sensors, és imprescindible disposar de diferents components electrònics per realitzar tota la connectivitat.

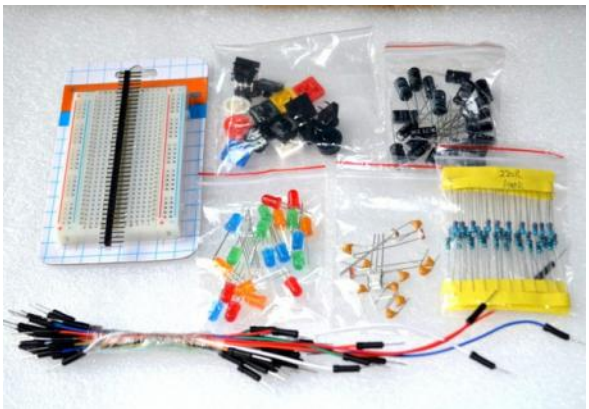


Figura 14. Complements electrónics.

Dit això, com podem observar, tota aquesta varietat d'accessoris, fa que realment es pugui utilitzar aquest tipus de hardware per realitzar projectes totalment diferents uns dels altres i d'aquesta manera arribar a un públic molt divers.

### 5.2.3.- Placa i accessoris utilitzats

A continuació, s'indiquen els materials relacionats amb Arduino, i la pròpia placa Arduino que s'ha escollit, i que seran necessaris per la realització del projecte.

#### 5.2.3.1.- Arduino UNO

Tenint en compte que en aquest projecte no hi haurà una quantitat excessiva de components connectats al Arduino, s'ha optat per fer servir la placa Arduino UNO, la qual disposa de les característiques següents:

| | |
|---|---|
| Microcontrolador: | ATmega328 |
| Voltatge de treball: | 5V DC |
| Voltatge d'entrada (Recomanat): | 7 - 12V DC |
| Voltatge d'entrada (Límits): | 6 - 20V DC |
| Entrades/Sortides Digitals: | 14 |
| Entrades Analògiques: | 6 |
| Corrent DC per cada Pin E/S: | 40 mA |
| Corrent DC del Pin 3.3V: | 50 mA |
| Memòria Flash: | 32 KB |
| SRAM: | 2 KB |
| EEPROM: | 1 KB |
| Velocitat de rellotge: | 16 MHz |


Figura 15. Arduino UNO.

Com es pot observar, disposa de 6 entrades analògiques i 14 entrades/sortides digitals, les quals son suficients per poder connectar els sensors que es faran servir, i la controladora de motors.

#### 5.2.3.2.- Sensor de temperatura i humitat relativa

El vehicle d'exploració, incorporarà un mòdul sensor de temperatura i humitat relativa model DHT22. Es tracta d'un sensor econòmic però amb unes bones prestacions i rangs de treball. Concretament, les característiques d'aquest model de sensor son:

| | |
|---|---|
| Voltatge de treball: | 3 - 5V DC |
| Rang de temperatura: | - 40 / 80ºC |
| Rang d'humitat relativa: | 0 - 100% |

Figura 16. Sensor DHT22.

Figura 17. Sensor DHT11.

Una alternativa una mica més econòmica encara que el sensor DHT22, seria el model DHT11, que proporciona un rang de treball més reduït i una menor precisió, ja que aquest model, només treballa en un rang de temperatura de 0 a 50ºC i en un rang d'humitat del 20 al 80%.

### 5.2.3.3.- Sensor gasos inflamables/combustibles (MQ-2)

Un altre mòdul de sensor que es farà servir, és el anomenat MQ-2. Aquest sensor, detecta l'existència de diferents tipus de gasos inflamables i combustibles en l'ambient, com per exemple: Gas metà, butà, GLP (gas liquat del petroli), i també, fa de detector de fum.

Figura 18. Sensor MQ-2.

### 5.2.3.4.- Sistema d'alimentació

Una placa Arduino, pot ser alimentada de dues maneres diferents:

- Mitjançant **el connector "jack"**, al qual podem connectar una bateria/pila o font d'alimentació que estigui dintre dels límits de voltatge indicats anteriorment.

- Mitjançant **el port USB** connectant-lo directament un PC o altre dispositiu el qual li subministrarà 5V.

En aquest vehicle, s'ha optat per connectar la Raspberry Pi al port USB de la placa Arduino i d'aquesta manera, utilitzar aquest USB per l'alimentació i per la comunicació entre els dos dispositius.

## 5.3.- Raspberry Pi

### 5.3.1.- Conceptes bàsics

Com en el cas del Arduino, la Raspberry Pi també forma part del món del hardware lliure.

La Raspberry Pi, és un ordinador de dimensions molt reduïdes, però que te la capacitat de fer moltes de les coses que faries amb un ordinador "estàndard", com per exemple, navegar per Internet, reproduir musica i vídeos, jugar a llocs senzills, i moltes altes coses.

Per això, com tot ordinador, la Raspberry Pi disposa d'una placa base, un processador, memòria RAM, targeta gràfica, targeta de so, interfície de xarxa, emmagatzematge (mitjançant targetes SD) i connexions USB per a perifèrics. Pel que fa al sistema operatiu, bàsicament treballa amb distribucions Linux optimitzades per a oferir un bon rendiment.

El primer model de Raspberry Pi (model A) va sortir a la venda en Febrer de 2012. Posteriorment, es va anar millorant i van treure els models B i B+, tot i que, esta previst que aquest mateix any 2014, aparegui el nou model A+.

Les diferencies entre els models A/A+ i B/B+ son mínimes, i bàsicament es troben en la connectivitat i consum energètic. Les diferencies més importants son les següents:

- **Models A/A+:** No disposen de connexió ethernet, només tenen 1 USB i tenen 256 MB de memòria RAM.

- **Models B/B+:** Tenen un connector RJ45 ethernet, 2 ports USB el model B i 4 el B+, i tenen 512 MB de memòria RAM.

Per una altra banda, també tenim que diferenciar, que entre les versions inicials dels dos models A i B, i les versions més avançades A+ i B+, també hi ha alguna diferencia, evidentment.

En les primeres versions, només hi havia disponibles 8 pins GPIO i el mòdul d'emmagatzematge feia servir una targeta SD, en canvi, les noves versions A+ i B+, disposen de 17 pins GPIO i utilitzen targetes microSD amb sistema push-push, el que assegura el contacte entre la targeta i la placa base.

### 5.3.2.- Accessoris Raspberry Pi

En aquest cas, tenint en compte que estem parlant d'un dispositiu que no deixa de ser un ordinador (miniPC), els accessoris que podem trobar, o millor dit, utilitzar, no es distancien gaire dels que faríem servir en un PC normal i corrent.

**Dispositius USB:** Com ja s'ha comentat, es poden connectar a la Raspberry una gran quantitat de dispositius mitjançant els dos ports USB: Adaptadors WiFi, pendrive, HDD, teclat, mouse, etc. Inclús, es possible connectar un hub USB per tal d'ampliar el número de ports disponibles.

**Targetes de memòria:** Tant si son SD pels models A i B, com microSD pel cas del A+ i B+, les targetes de memòria son un accessori indispensable per tal d'instal·lar el SO de l'equip.

Figura 19. Targeta microSD amb adaptador SD.

**Càmeres:** Actualment trobarem 2 models diferents de càmeres dissenyades per la Raspberry i que es connecten al port CSI de la placa base.

Figura 20. Càmeres Raspberry Pi.

**Displays:** A la placa, trobarem un altre port similar al CSI, anomenat DSI (Display Serial Interface), per tal de poder connectar displays LCD.

**Sensors:** Com en el cas de les plaques Arduino, la Raspberry Pi també te la possibilitat de connectar alguns sensors als seus pins GPIO (Digitals). En el cas de fer servir sensors analògics, es necessari disposar d'un conversos analògic-digital per tal de poder-los fer servir, ja que a diferència de l'Arduino, la Raspberry no disposa de cap pin d'entrada analògic.

### 5.3.3.- Model i accessoris utilitzats

A continuació, hi ha una petita explicació dels motius pels quals s'ha escollit cada accessori o model de Raspberry i les característiques bàsiques d'aquests.

#### 5.3.3.1.- Raspberry Pi Model B

Per a aquest projecte, serà necessari disposar de dos ports USB, per tant, la millor opció és fer servir una Raspberry Pi model B o B+, ja que el model A només te un sol port.

Per una altra banda, com que a part de la pròpia Raspberry Pi, també es farà servir una placa Arduino, no és necessari que la Raspeberry disposi de tants pins GPIO, així que, he decidir fer-ho amb el model B, perquè ja disposava d'un de projectes anteriors.

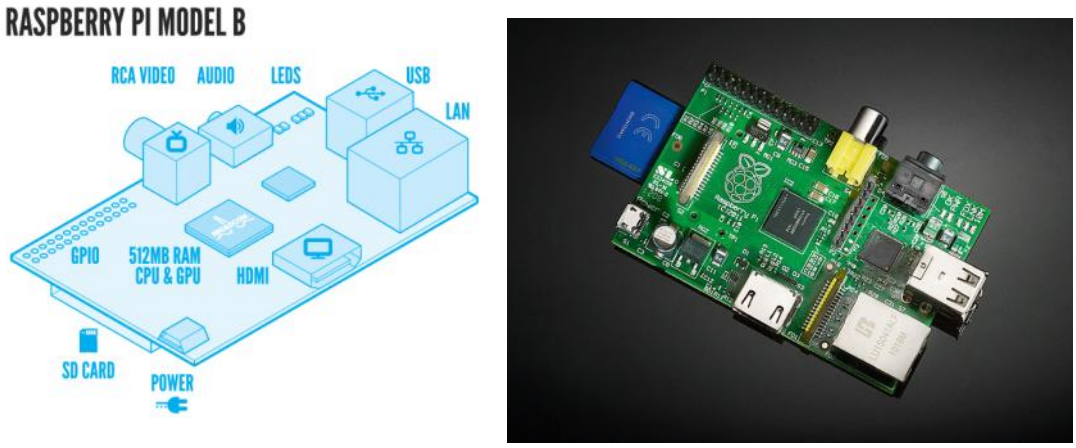



Figura 21. Raspberry Pi model B.

Les característiques més destacables d'aquest model de Raspberry Pi, son les següents:

- Processador: ARM1176JZF-S ( 700 MHz)
- Gràfica: Broadcom VideoCore IV
- Memòria RAM: 512 MB
- Ranura expansió: SD
- Alimentació: 5V/700 mA
- Connexions:
  - 2 ports USB 2.0
  - 1 mini jack 3.5 de sortida d'àudio
  - 1 sortida HDMI
  - 1 sortida RCA
  - 1 port microUSB per alimentació
  - 1 port RJ45 ethernet 10/100

### 5.3.3.1.- Targeta SD i sistema operatiu

Es farà servir una targeta SD de 16 GB per tal d'instal·lar en ella el sistema operatiu de la Raspberry. El sistema operatiu en sí mateix, no requereix una SD de tanta capacitat, però tenint en compte que voldrem emmagatzemar les dades dels sensors, és recomanable que no anem justos d'espai.

Pel que fa al sistema operatiu utilitzat, he escollit el SO Raspbian "Debian Wheezy" ja que és un dels principals SO per Raspberry, un dels més senzills per gent poc experta, i a més, ja ve preparat per programar amb llenguatge Python, i serà necessari més endavant.

### 5.3.3.2.- Adaptador WiFi

Evidentment, encara que la Raspberry Pi ja disposa de interfície ethernet per la connectivitat de xarxa, necessitem afegir-li un dispositiu WiFi per tal de poder-lo connectar a "l'estació base" i enviar i rebre les dades del vehicle.

Per aquest motiu, he escollit un petit dispositiu WiFi (CSL Mini Dongle 802.11 n/b/g) que a part de ser totalment compatible amb el SO Raspbian, disposa d'una petita antena extraïble amb connector RP-SMA.

El fet que m'ha portat a escollir aquest adaptador, a estat precisament el fet que tingues un connector estàndard i que l'antena es pugues canviar. A més, juntament amb aquest dispositiu venia una segona antena omnidireccional de 12 dBi per tal d'ampliar notablement la cobertura de WiFi.

**Figura 22. Adaptador WiFi USB amb antena de 12 dBi.**

### 5.3.3.3.- Càmera NoIR

L'altre accessori que es farà servir juntament amb la Raspberry, és una petita càmera dissenyada especialment per aquest miniPC.

Concretament, la càmera que es muntarà al vehicle és el model Pi NoIR. Aquest model de càmera te una qualitat d'imatge bastant més baixa a la llum del dia que l'altre model existent (RaspiCam), però la diferencia bàsica, resideix en que el model NoIR, amb la ajuda d'uns leds IR (infraroig) ens permet obtenir visió sense cap tipus de llum addicional.


Figura 23. Càmera Raspberry Pi NoIR.

Figura 24. LED's infrarojos (IR).

### 5.3.3.3.- Sistema d'alimentació

Com s'ha vist en les especificacions del model de Raspberry utilitzat, és necessari realitzar l'alimentació amb 5V de tensió i un mínim de 700mA de corrent. El problema, és que com s'ha comentat anteriorment, s'aprofitarà la connexió USB de la Raspberry per alimentar a la seva vegada, la placa Arduino UNO. Si a això li sumem, que la Raspberry també tindrà connectat un dispositiu USB WiFi i una càmera de vídeo, fa necessari subministrar més de 700mA perquè tot funcioni correctament.

Per aquest motiu, s'utilitzarà un acumulador de bateria extern que pot subministrar fins a 2A, a part de tenir una gran autonomia amb una capacitat de 15000mAh.


Figura 25. Bateria externa USB de 15000mAh.

# 6.- Disseny vehicle RC d'exploració

## 6.1.- Sistema de control

El primera pas, serà entendre per a que serveix cada connector de la controladora de motors L298D, i perquè es farà servir cadascuna de les sortides de la placa.

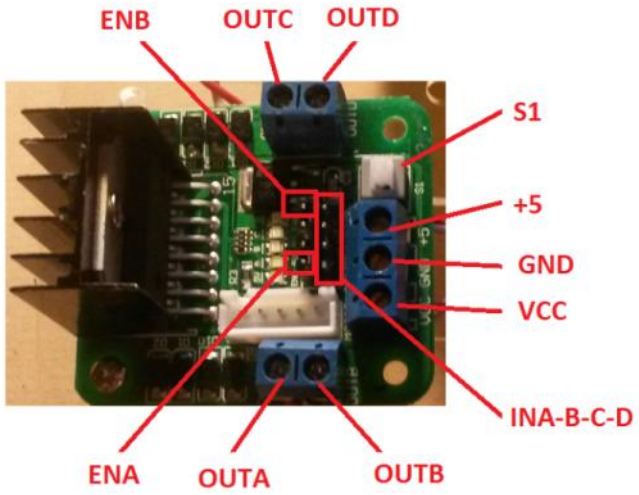



Figura 26. Connexions controladora motors.

**S1:** És un interruptor que ens permetrà tallar o deixar passar la corrent a la placa i als motors.
**VCC:** Entrada de corrent continua per alimentar la placa i motors.
**GND:** Contacte de massa per generar el diferencial de tensió amb VCC i que hi hagi circulació de corrent.
**OUTA:** Pol positiu dels motors de la dreta del vehicle.
**OUTB:** Pol negatiu dels motors de la dreta del vehicle.
**OUTC:** Pol positiu dels motors de l'esquerra del vehicle.
**OUTD:** Pol negatiu dels motors de l'esquerra del vehicle.

> NOTA: Com podem observar, com només tenim 2 sortides de motors (A+B i C+D) cada sortida controlarà 2 rodes i per tant, les dues rodes del mateix costat sempre giraran en el mateix sentit de la marxa.

**INA:** Rep des de l'Arduino, les ordres que te que enviar a la sortida OUTA.
**INB:** Rep des de l'Arduino, les ordres que te que enviar a la sortida OUTB.
**INC:** Rep des de l'Arduino, les ordres que te que enviar a la sortida OUTC.
**IND:** Rep des de l'Arduino, les ordres que te que enviar a la sortida OUTD.
**ENA:** Determina la velocitat a la que giraran els motors connectats en la sortida (OUTA+B)
**ENB:** Determina la velocitat a la que giraran els motors connectats en la sortida (OUTC+D)

### 6.1.1.- Configuració i funcionament

El sistema de control que s'ha ideat per tal de controlar el vehicle, fa servir set tecles, tres per canviar la velocitat dels motors, i quatre per determinar la direcció i moviments del vehicle.

Les tecles en qüestió son les següents:

**W = Davant;** **S = Darrere;** **A = Esquerra;** **D = Dreta;**

**1 = Velocitat lenta;** **2 = Velocitat mitja;** **3 = Velocitat ràpida;**

Per tal de fer arribar aquestes ordres al Arduino que incorpora el vehicle, es pot fer servir tant les pròpies tecles del teclat, com uns botons, anomenats igual que les tecles indicades anteriorment, que incorporarà la interfície d'usuari (explicada en detall en el apartat 6.5.1).

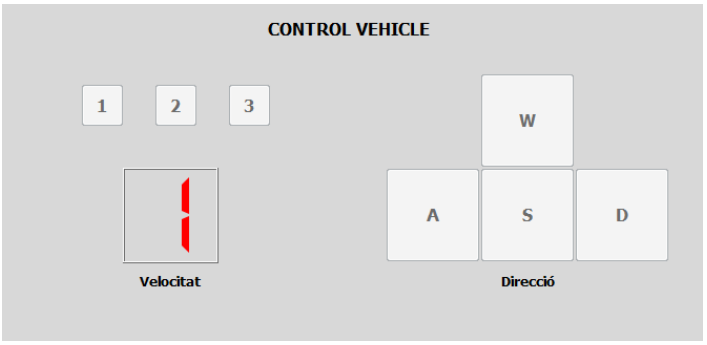



Figura 27. Interfície gràfica d'usuari. Control vehicle.

A més, com que el propi Arduino no té connexió directa amb l'estació base, les ordres de les tecles premudes, s'envien mitjançant un socket TCP creat entre l'estació base i la Raspberry Pi, i és aquesta, que en rebre les ordres les envia mitjançant una connexió sèrie (USB) l'Arduino. L'Arduino, per la seva banda, està programat amb totes les ordres corresponents per tal de modificar la velocitat o fer girar els motors necessaris en un sentit o un altre, segons les ordres rebudes (veure codi Arduino).

### 6.1.2.- Esquema de connexions

Tal i com s'ha comentat en el punt anterior, els motors del mateix costat, estaran controlats per les mateixes sortides de la controladora L298D. Així que, com s'observa en la següent imatge, s'ha realitzat la connexió dels motors de la dreta a les sortides A/B i els de l'esquerra a les sortides C/D.

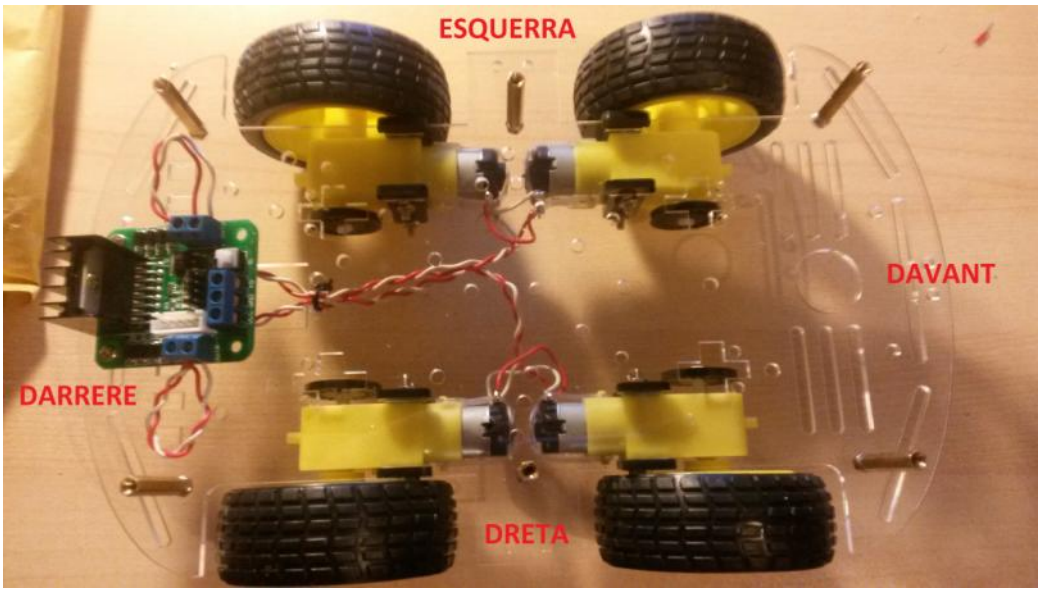



Figura 28. Indicacions de direcció segons el xassís.

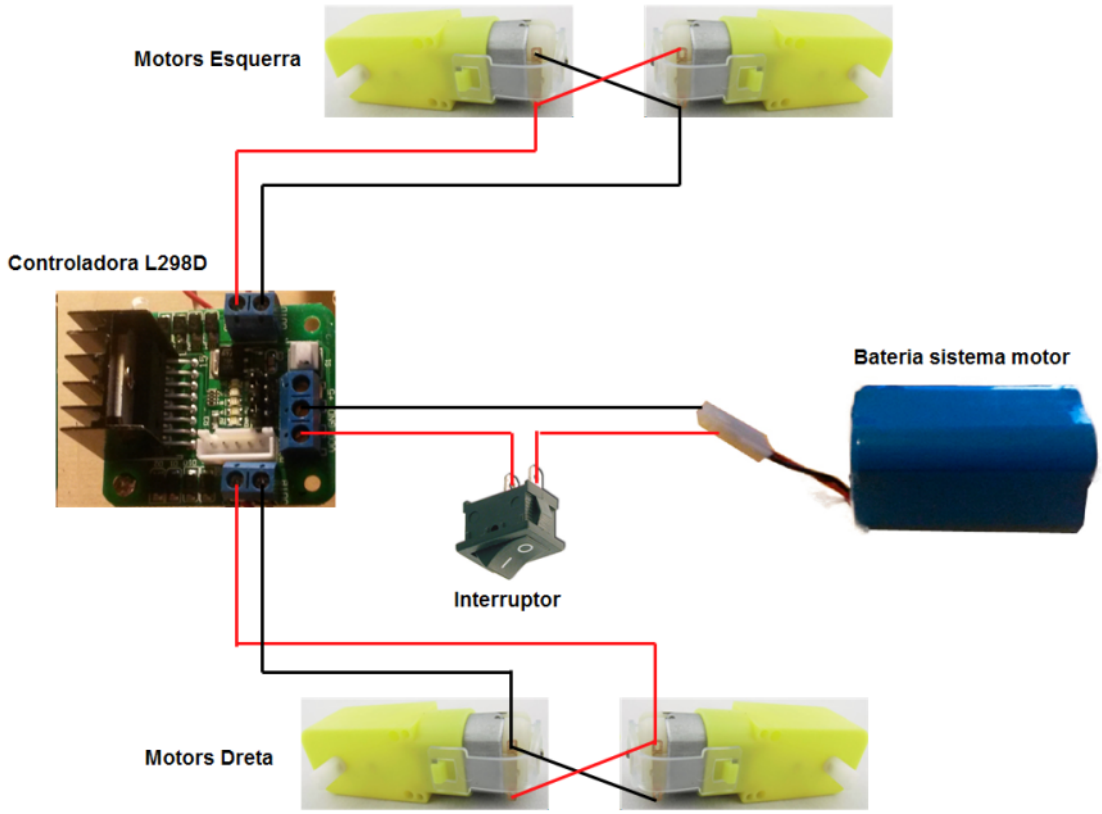



Figura 29. Esquema de connexions del sistema motor.

Per una altra banda, a part de realitzar la connexió dels motors a la controladora L298D, és necessari connectar aquesta al Arduino per tal de rebre les ordres amb les que tindrà que controlar els motors. Per fer això, es realitzaran les connexions que es mostren a la figura 30.

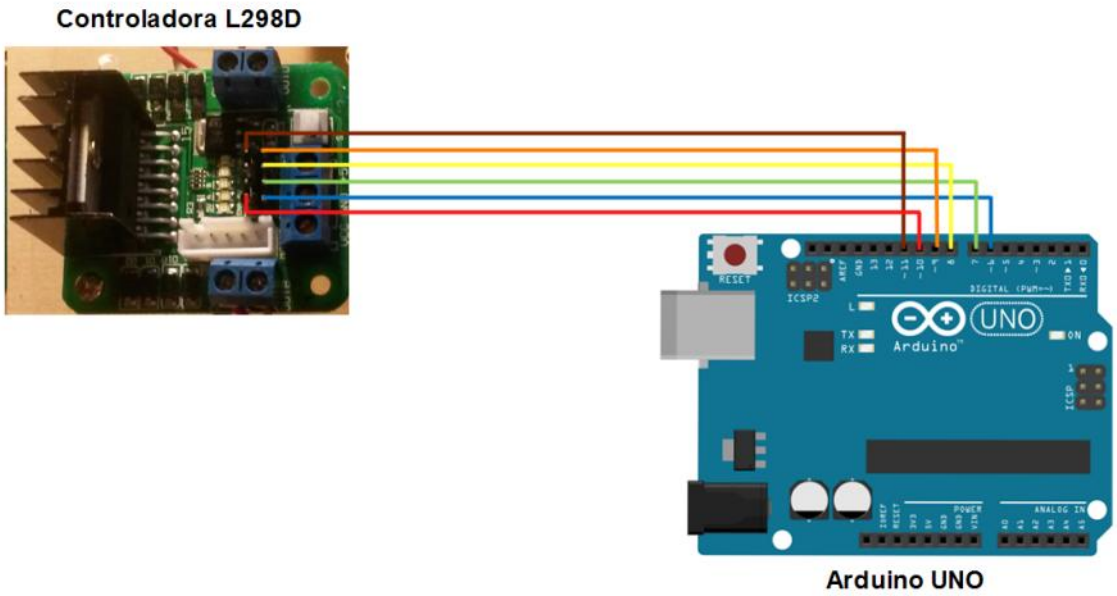



Figura 30. Esquema de connexions de la controladora.

Com podem observar, els pins ENA i ENB de la controladora, estan connectats a dos pins (10 i 11) del tipus PWM (pulse-width modulation), això és així perquè aquests pins s'utilitzaran com a sortida analògica per tal d'enviar el valor de velocitat que tindrà que configurar el driver en els motors.

Per tal de determinar quins motors s'han de posar en marxa, i en quin sentit han de girar (polaritat), es connectaran les entrades del driver, INA-B-C-D, als pins 6, 7, 8 i 9. Aquests pins, no és necessari que siguin PWM, ja que la única ordre és 0 o 1, engegat o aturat (encara que el pin 9 i 6 també son PWM, no és fa servir aquesta funcionalitat).

## 6.2.- Implementació dels sensors

Tal i com s'ha explicat en els apartats anteriors, el vehicle te incorporats dos sensors, un dels quals te doble funció. Es tracta d'un sensor DHT22 que permet determinar el valor de la temperatura ambient i de la humitat relativa, i d'un sensor MQ-2, el qual és capaç de detectar partícules de fum i diferents gasos (metà, propà, butà, etc.).

Inicialment, s'havia plantejat que aquests sensors estiguessin connectats directament a l'Arduino, el qual enviaria per sèrie les dades recollides a la Raspberry Pi, per posteriorment fer-les arribar a l'estació base.

Aquest plantejament inicial, va tenir que ser modificat ja que es va trobar la problemàtica, que al fer servir un tipus de connexió tant limitada com la sèrie per enviar les dades dels sensors entre l'Arduino i la Raspberry Pi, a la mateixa vegada que s'enviaven les ordres per controlar els motors, provocava un "col·lapse" que feia inviable aquesta solució, ja que el moviment del vehicle es veia interromput cada cop que s'enviaven dades dels sensors.

El motiu que inicialment s'hagués pensat en aquesta solució, va ser per la poca complexitat d'implantació, ja que la programació d'aquest sensors en l'entorn Arduino es molt més senzill que en Raspberry Pi, a més, tal i com s'explica en els següents apartats, el sensor MQ-2 és de tipus analògic, el que complica la connexió amb la Raspberry, ja que aquesta disposa d'entrades analògiques.

### 6.1.1.- Configuració i funcionament

La millor solució que s'ha trobat per evitar la problemàtica comentada al punt anterior, és connectar els sensors directament a la Raspberry Pi, evitant així, que interfereixin amb les ordres enviades pel port sèrie entre Raspberry i Arduino.

Per tal de poder tenir els sensors funcionant en la Raspberry, s'ha tingut que tenir en compte certs aspectes per tal de poder fer la implementació.

S'ha fet servir unes llibreries pel sensor DHT22 publicades a Internet per la companyia Adafruit Industries per tal d'agilitzar la implementació d'aquest sensor.

Per una altra banda, un altre problema que s'ha trobat al voler realitzar la connexió dels sensors a la Raspberry Pi, ha sigut que, degut a que el sensor de fum/gas MQ-2 disposa d'una sortida analògica i no digital com en el cas del DHT22, ha fet falta connectar-lo mitjançant un chip convertidor Analògic/Digital model MCP3002, pel qual, també s'ha fet servir una llibreria obtinguda de "botbooks.com".

Un cop solucionats aquests petits inconvenients trobats, s'ha realitzat un petit programa/script en Python a la Raspberry Pi, per tal que realitzi la captura de les dades i el posterior enviament a l'estació base mitjançant un socket TCP independent al de control. Concretament, aquest script (ServidorSensors.py), en rebre una ordre des de l'estació base, se li passa un 1 per indicar que es volen rebre dades dels sensors, es llegeixen les dades dels 2 sensors, es crea un sol string amb totes les dades (temperatura, humitat i nivell de fum) i s'envia cap a la interfície d'usuari de l'estació base.

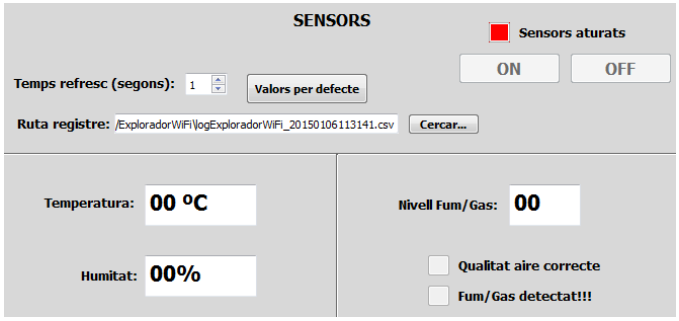



**Figura 31. Interfície gràfica d'usuari. Sensors.**

En rebre les dades el programa de l'estació base, mostra per pantalla els valors actuals dels sensors, i crea un fitxer de registre amb aquesta informació, juntament amb la data i hora actuals per tal de poder treballar amb aquests valors posteriorment (realitzar taules, gràfiques, etc.).

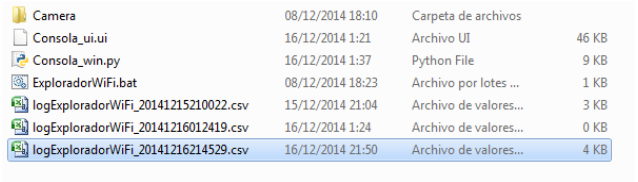



**Figura 32. Exemple de fitxer de registre.**

| | A | B | C | D |
|---|---|---|---|---|
| 1 | 16/12/2014 21:49:14 | 0 | 0 | 0 |
| 2 | 16/12/2014 21:49:15 | 0 | 0 | 0 |
| 3 | 16/12/2014 21:49:16 | 20.0 | 58.7 | 182 |
| 4 | 16/12/2014 21:49:17 | 20.0 | 58.7 | 182 |
| 5 | 16/12/2014 21:49:18 | 20.0 | 58.7 | 182 |
| 6 | 16/12/2014 21:49:19 | 19.8 | 59.1 | 181 |
| 7 | 16/12/2014 21:49:20 | 19.8 | 59.1 | 181 |
| 8 | 16/12/2014 21:49:21 | 19.8 | 59.1 | 181 |
| 9 | 16/12/2014 21:49:22 | 19.8 | 59.1 | 181 |
| 10 | 16/12/2014 21:49:23 | 19.8 | 59.1 | 181 |
| 11 | 16/12/2014 21:49:24 | 19.8 | 59.0 | 181 |
| 12 | 16/12/2014 21:49:25 | 19.8 | 59.0 | 181 |
| 13 | 16/12/2014 21:49:26 | 19.8 | 59.0 | 181 |
| 14 | 16/12/2014 21:49:27 | 19.8 | 59.0 | 181 |
| 15 | 16/12/2014 21:49:28 | 19.8 | 59.0 | 181 |
| 16 | 16/12/2014 21:49:29 | 19.8 | 59.0 | 181 |
| 17 | 16/12/2014 21:49:30 | 19.8 | 59.0 | 174 |
| 18 | 16/12/2014 21:49:31 | 19.8 | 59.0 | 172 |
| 19 | 16/12/2014 21:49:32 | 19.8 | 59.0 | 172 |
| 20 | 16/12/2014 21:49:33 | 19.8 | 59.1 | 171 |
| 21 | 16/12/2014 21:49:34 | 19.8 | 59.1 | 171 |
| 22 | 16/12/2014 21:49:35 | 19.8 | 59.1 | 171 |
| 23 | 16/12/2014 21:49:36 | 19.8 | 59.0 | 171 |
| 24 | 16/12/2014 21:49:37 | 19.8 | 59.1 | 169 |
| 25 | 16/12/2014 21:49:38 | 19.8 | 59.1 | 169 |
| 26 | 16/12/2014 21:49:39 | 19.8 | 59.1 | 169 |
| 27 | 16/12/2014 21:49:40 | 19.8 | 59.1 | 171 |
| 28 | 16/12/2014 21:49:41 | 19.8 | 59.1 | 171 |

**Figura 33. Exemple de dades registrades.**

S'ha de tenir en compte, que el valor que es mostra del sensor MQ-2 de gas i/o fum, no te cap unitat de mesura, degut a que per poder-ho fer del tot correcte, i que retornés el valor de PPM (parts per milió) dels nivells de gas o fum en l'ambient, seria necessari regular-lo amb equips de mesura especialitzats.

Per aquest motiu, per tal de poder indicar d'alguna manera a la GUI que s'ha detectat fum i/o gas, s'ha fet proves exposant directament el sensor a una font de gas (encenedor) i a fum per tal de comprovar fins a quins nivells pujava el valor retornat pel sensor. Amb les dades obtingudes, s'ha estipulat que el llindar per indicar que s'ha detectat fum o gas fos **300**.

NOTA: Si en un futur es volgués comercialitzar aquest vehicle, es tindria que tenir en compte de fer unes llibreries pròpies, tant pel que fa al sensor DHT22, com pel chip mcp3002 ja que es possible que les utilitzades en aquest projecte, no estiguin destinades a fins comercials.

### 6.1.2.- Esquema de connexions

L'esquema de connexions que es va fer servir en primera instancia quan estaven connectats els sensors directament a l'Arduino, era el que es mostra a la figura 34.

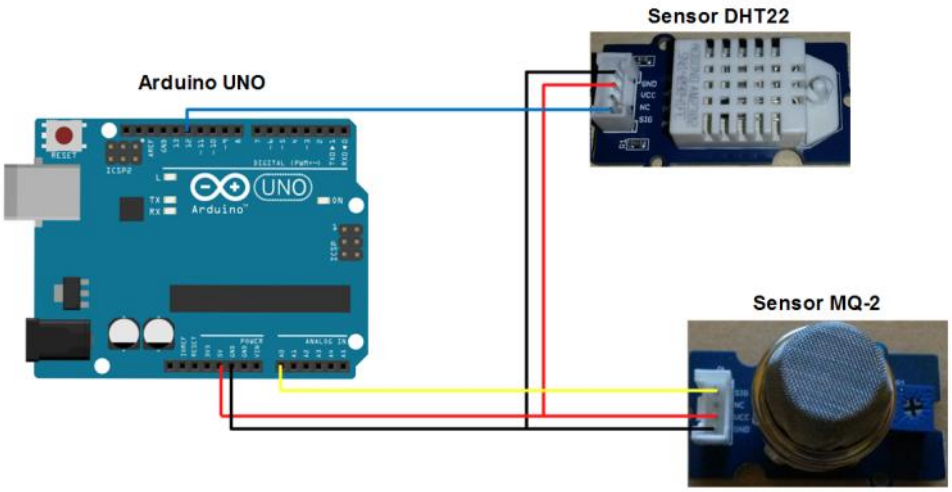



Figura 34. Esquema de connexions dels sensors en Arduino.

Per una altra banda, en la figura 35, es mostra com està realitzada la connexió dels sensors amb el nou plantejament que es va realitzar després dels problemes torbats amb la solució anterior.

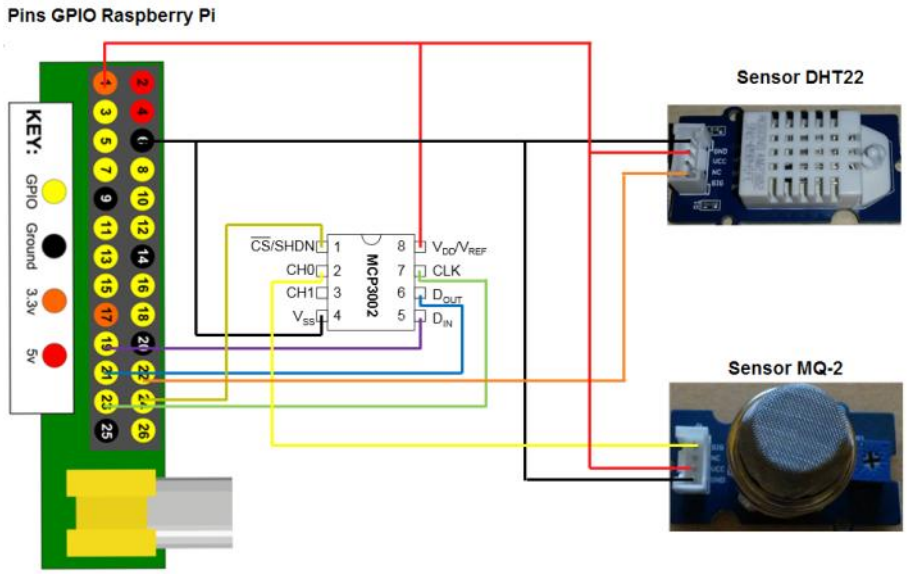



Figura 35. Esquema de connexions dels sensors en la Raspberry Pi.

Com es pot veure a l'esquema, al connectar els sensors a la Raspberry Pi, es tenen que fer servir els pins **GPIO** (General Purpose Input/Output). Per aquest motiu, és necessari tenir clar quina és la funció de cada pin. Bàsicament, com es veu a la imatge dels pins a l'esquema, hi ha pins per a 4 funcions principals: alimentació **3.3V, 5V, terra/GND, i pins d'entrada/sortida (GPIO)**. En la web http://www.raspberrypi.org/documentation/usage/gpio/README.md, es pot trobar tota la informació necessària.

## 6.3.- Connexió Arduino <--> Raspberry Pi

Com s'ha indicat en punts anteriors, per tal de enviar i rebre dades entre les dues plaques de hardware utilitzades, Arduino i Raspberry Pi, s'ha fet servir el port USB d'aquestes dues per la interconnexió. Al fer-ho d'aquesta manera, obtenim, a part d'una via de comunicació, l'alimentació necessària des de la Raspberry Pi, per poder alimentar la placa Arduino.

Per tal de poder crear aquesta comunicació entre els dos dispositius, la única cosa necessària serà incloure el codi corresponent a la inicialització de la comunicació sèrie, indicant el port i la velocitat del port, tant en la banda de l'Arduino, com en la banda de la Raspberry Pi, en el programa/script principal.

A continuació, es mostren les línies de codi necessàries per realitzar aquesta comunicació:

Codi Arduino per la comunicació sèrie:

```
void Setup(){
        Serial.begin(57600);   #Inicialitza el port sèrie a la velocitat indicada.
        ...
}
...
void loop(){
        while(Serial.available()){        #Determina si la comunicació sèrie està disponible.
                char c = Serial.read();   #Realitza una lectura del port sèrie i l'emmagatzema.
                 switch(c){
                ...
}
```

***Codi Python al script principal de la Raspberry Pi per la comunicació sèrie:***

```
import serial                    #S'importa el mòdul necessari per la comunicació sèrie.
...
arduino = serial.Serial('/dev/ttyACM0', 57600) #Inicialització de la comunicació sèrie.
...
arduino.write(ordre)             #Envia pel port sèrie el contingut de la variable ordre.
...
```

## 6.4.- Streaming càmera On-Board

Amb la intenció de poder controlar el vehicle d'exploració sense necessitat de tenir visió directe amb ell, aquest, disposa d'una videocàmera connectada a la Raspberry Pi.

### 6.4.1.- Configuració i funcionament

Degut a que la càmera utilitzada està especialment dissenyada per la Raspberry Pi, només ha sigut necessari un parell de passos per tal d'habilitar-la en el dispositiu. Per fer-ho, el primer que s'ha de fer, és connectar la càmera al port habilitat per aquesta funció en la Raspberry Pi.

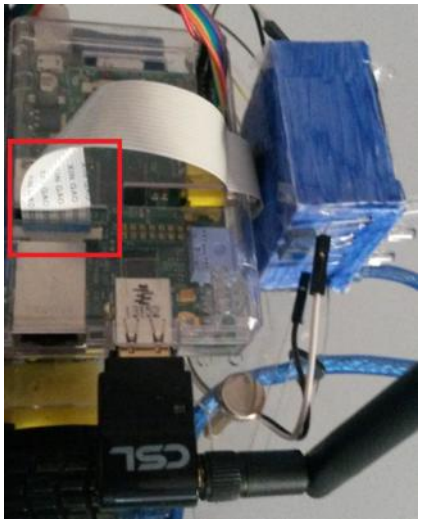


Figura 36. Connexionat càmera en Raspberry Pi.

Un cop connectada, s'ha d'entrar al menú principal de configuració del SO de la Raspberry (**sudo raspi-config)**, i habilitar-la mitjançant l'opció "**camera**".

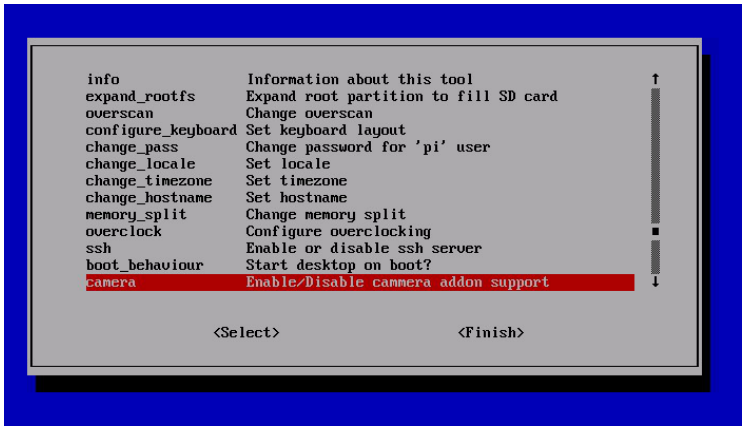



Figura 37. Menú configuració SO Raspbian.

Per tal de realitzar la captura de vídeo i posteriorment generar la difusió d'aquestes imatges i que puguin ser visualitzades des de l'estació base, es fa servir la comanda Raspivid. Però aquesta eina per si sola, només realitza la captura d'imatges, no genera la difusió, per això, inicialment s'havia fet servir juntament amb el programa VLC, per tal que aquest generes la difusió mitjançant un port concret. La comanda que s'havia fet servir era:

**sudo raspivid -t 999999 -w 640 -h 480 -o - | cvlc -vvv stream:///dev/stdin --sout '#rtp{sdp=rtsp://:10002/}' :demux=h264**

Amb això, s'aconseguia crear la difusió de les imatges i que mitjançant un client de vídeo com VLC a l'estació base, es pogués connectar a la URL **rstp://<IP RASPBERRY>:10002** i veure les imatges capturades des del vehicle. El problema va estar, que amb aquesta solució era impossible realitzar una conducció del vehicle de forma optima, ja que les imatges que es veien des de l'estació base, tenien més de 2 segons de retràs.

Per aquest motiu, s'ha optat per una altra solució, que encara que també te els seus inconvenients, té una latència gairebé inapreciable.

La solució utilitzada, no és més que realitzar la captura d'imatges amb la mateixa comanda anterior (via Raspivid), però en comptes de fer servir "cvlc" per crear la difusió, es fa servir el programa NetCat, per tal de crear un socket de comunicació entre l'estació base i el vehicle per tal d'enviar els paquets de dades que contenen les imatges de vídeo.

El major inconvenient trobat amb aquesta solució, radica en el fet que en comptes de que el vehicle estigui en mode servidor (Listen) a l'espera que un client, com l'estació base li demani les imatges, serà l'estació base, qui crearà inicialment el socket en mode escolta, per tal de rebre les imatges enviades des del vehicle.

Comanda executada en l'estació base:

**netcat\nc64.exe -L -p 10002 | mplayer\mplayer.exe -vo direct3d -fps 25 -cache 512 -cache-min 5 -**

Comanda executada en la Raspberry:

**sudo raspivid -t 999999 -w 640 -h 480 -o - | nc <IP estació base> <Port estació base>**

Per automatitzar l'execució d'aquestes comandes, s'ha creat diferents scripts.

En primera instancia, en el moment que l'usuari engega la consola principal del vehicle, ho fa mitjançant un BAT (ExploradorWiFi.bat) el qual, a part d'obrir la consola principal, executa un segon BAT (clientCamera.bat) per tal d'obrir el programa NetCat i crear un socket que es quedarà a l'espera de rebre imatges pel port 10002.

Encara que aquesta part, la creació del socket en l'estació base es realitza només obrir la consola principal, la càmera del vehicle continua aturada per tal de no gastar bateria, recursos de CPU i ample de banda. El que s'ha creat, es un apartat a la consola principal, que permet

engegar i aturar la càmera del vehicle. Concretament, en el moment que l'usuari inicialitza la càmera, el script principal de la Raspberry (ServidorConsola.py) crida al script Camera.sh per tal d'executar la comanda explicada anteriorment a la Raspberry que comenci a capturar les imatges i enviar-les amb NetCat mitjançant el socket creat des de l'estació base.

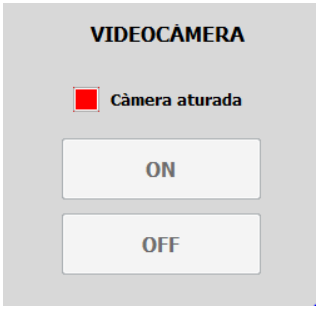



Figura 38. Interfície gràfica d'usuari. Videocàmera.

### 6.4.2.- Visió nocturna (LED's IR)

Com s'ha comentat en l'apartat de material utilitzat, la càmera instal·lada en el vehicle es sensible a la llum infraroja, es va escollir aquesta càmera, amb la intenció d'instal·lar-la juntament amb uns LEDS IR els quals permetin visualitzar les imatges sense disposar de cap altre font de llum natural o artificial.

El primer que s'ha fet, ha sigut realitzar el càlcul per saber quina resistència era necessària instal·lar en el circuit de les dues línies de tres LED's IR que s'ha fet servir. Per poder realitzar aquest càlcul, és necessari disposar prèviament de les característiques tècniques dels LED's utilitzats. En aquests cas, les característiques son les següents:

Tensió del LED: **1,5V** Intensitat de treball del LED: **20mA**

Amb aquestes dues dades, i sabent que l'Arduino subministra una tensió a les línies de LED's de 5V, aplicarem la següent formula:

$$R = \frac{Tensió\ subministrada - (quantitat\ LEDs \cdot Tensió\ al\ LED)}{Intensitat\ de\ treball}$$

$$R = \frac{5V - (3 \cdot 1.5V)}{20mA} = \mathbf{25\Omega}$$

Com podem observar, serà necessari aplicar una resistència de $25\Omega$ en cada línia de LED's. En aquest cas, s'ha fet servir dos resistències variables (1Ω - 10kΩ) regulades a 25Ω amb un multímetre, però és pot fer servir perfectament la resistència de valor més pròxim a 25Ω, com per exemple, una de 24Ω o una de 27Ω.

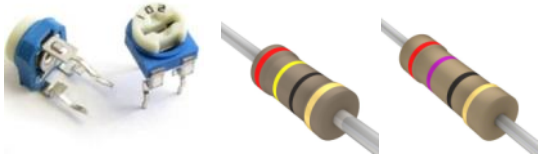


Figura 39. Resistència variable, resistència 24Ω i resistència 27Ω .

Un cop connectades les dues línies de LED's amb les seves resistències corresponents, queda un circuit com el del esquema de la figura 40.

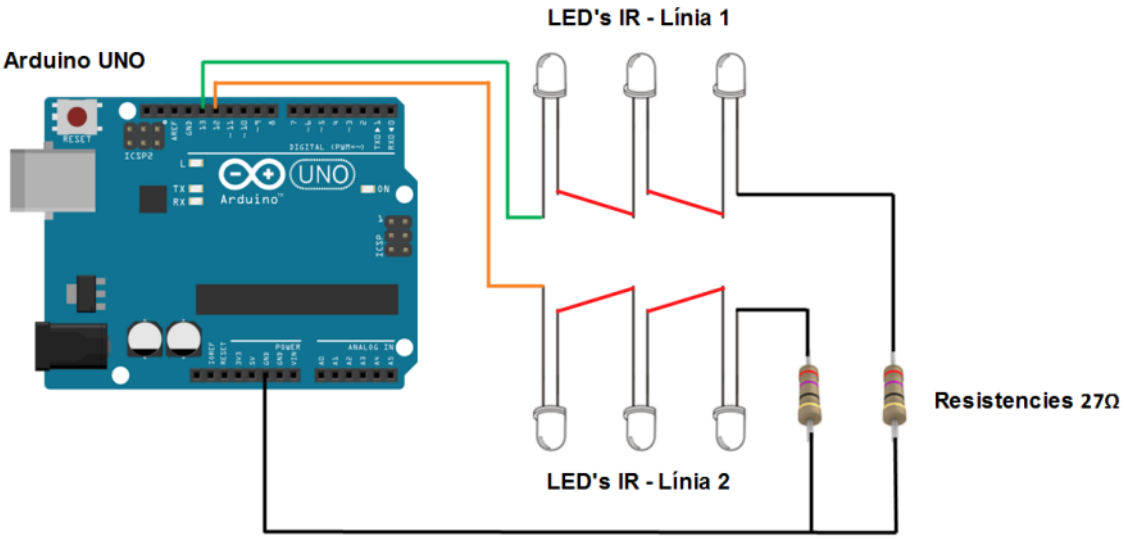



Figura 40. Esquema connexions LED's IR en Arduino.

Com s'observa en aquest esquema, cada línia estarà connectada a un PIN de l'Arduino i a un pin de terra/massa (GND). S'ha fet d'aquesta manera, ja que si agüéssim connectat tots els LED's en sèrie a la mateixa línia, necessitaríem una alimentació de com a mínim 9V, i l'Arduino només ens proporciona 5V.

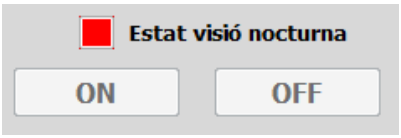



Figura 41. Interfície gràfica d'usuari. LED's IR.

De la mateixa manera que és possible engegar i aturar la càmera del vehicle, també s'ha implementat aquesta possibilitat pels LED's IR, i d'aquesta manera només engegar-los quan realment sigui necessari, estalviant d'aquesta manera, bateria i recursos.

## 6.5.- Estació base / Equip de control

### 6.5.1.- Disseny de la interfície gràfica d'usuari (GUI)

A l'hora de realitzar la interfície gràfica d'usuari (GUI) per tal de controlar tots els aspectes i funcionalitats del vehicle, hi havien moltes alternatives, ja fos pel llenguatge a utilitzar, com per la ubicació d'aquesta GUI (en local en l'equip de control, o en web en la Raspberry Pi).

Finalment, s'ha optat per realitzar la GUI en llenguatge Python degut a la seva facilitat de programació i de que es tracta d'un llenguatge molt estès, sobretot, en la comunitat de Linux, i això fa que hi hagi moltes aplicacions i llibreries desenvolupades en aquest llenguatge.

Per una altra banda, s'ha optat inicialment per realitzar una GUI que s'executi directament en l'equip de control, aquest fet, perjudica la flexibilitat pel que fa al equip de control, ja que serà necessari tenir instal·lat el intèrpret de Python en els equips que es vulguin fer servir per a aquest propòsit. Però en canvi, ens aporta el avantatge de alliberar el consum de recursos de la Raspberry Pi al no tenir constantment un servei web executant-se.

El primer pas per crear la GUI, ha estat elaborar tota la part gràfica (botons, camps de text, etiquetes de text, etc.), fent servir un programa destinat a aquesta tasca.

Concretament, s'ha fet servir el programa Qt Designer (www.qt.io), el qual disposa de llicencies tan GPL com llicencies per desenvolupaments comercials.

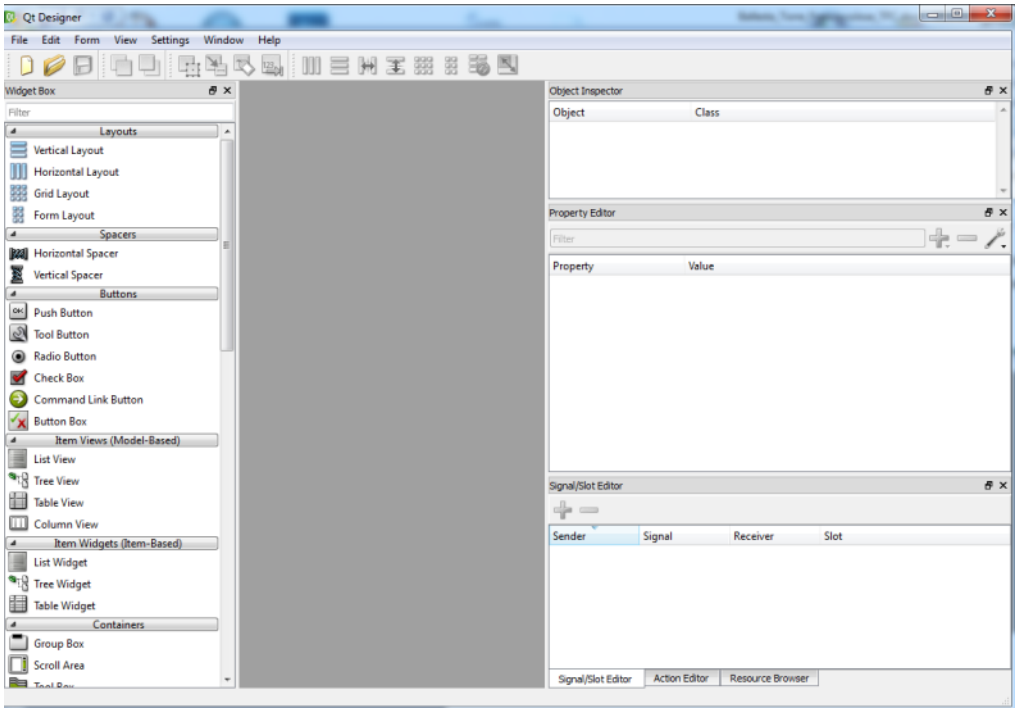



Figura 42. Interfície principal del programa Qt Designer.

Al generar tota l'estrutcura gràfica de la GUI amb aquesta aplicació, et genera un fitxer amb extensió **.ui,** el qual, mitjançant una serie de llibreries, es pot exportar en el programa principal creat en Python, on es realitzaran totes les funcions i tasques que tenen que fer els botons, camps de text, etc., creats amb el Qt Dessigner.

A continuació, en la figura 43, es pot veure, com s'importen aquestes llibreries PyQt4, per tal de poder treballar amb els elements creats anteriorment.

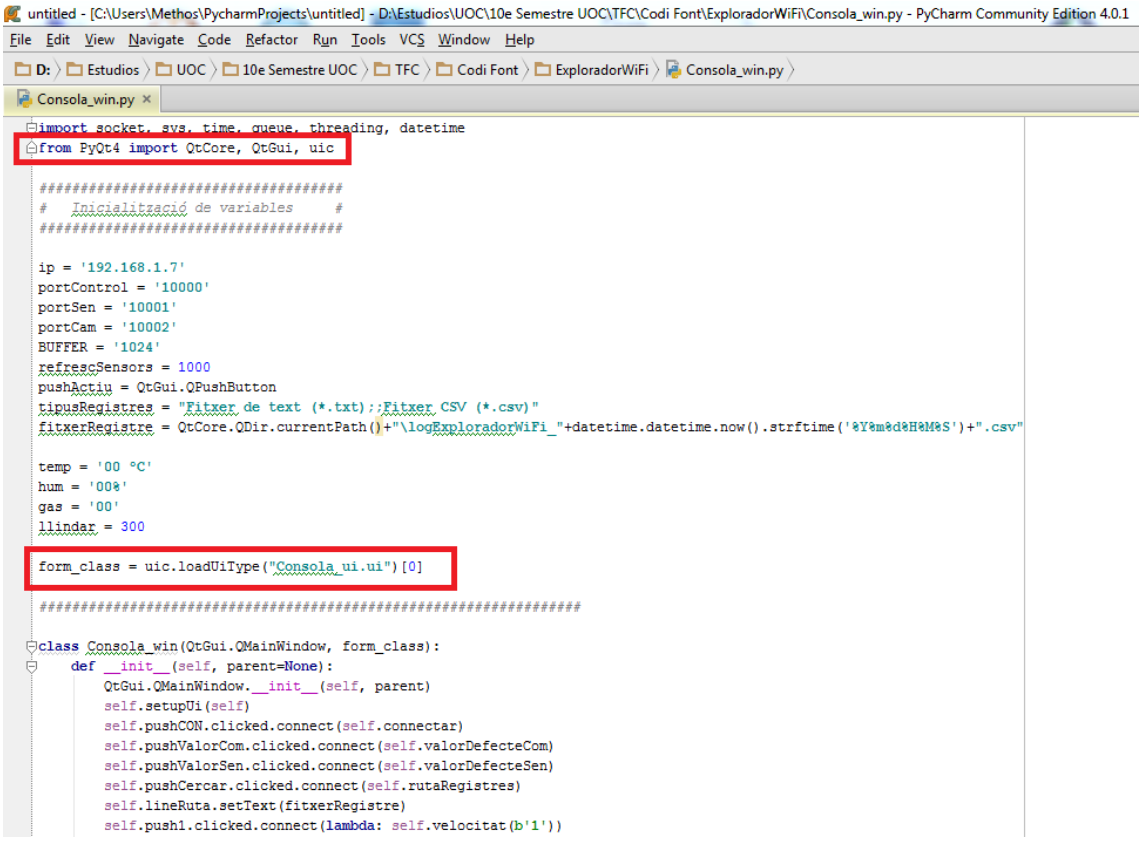



**Figura 43. Interfície principal programa PyCharm on es veu la importació del fitxer UI.**

Pel que fa a la programació de totes les funcions de la consola principal (Consola_win.py), desenvolupada en Python, s'ha utilitzat l'entorn de desenvolupament **PyCharm**, el qual, de la mateixa manera que Qt Designer, disposa tant de llicencia per a fins comercials, com llicencia per a projectes "open source" o d'àmbit acadèmic.

Un cop ja està creada l'estructura gràfica, i s'han programat totes les funcions del programa, el codi del fitxer principal "Consola_win.py" quedarà tal i com es mostra en l'Annex 4.

Visualment, la GUI creada quedarà com es mostra a la figura 44.

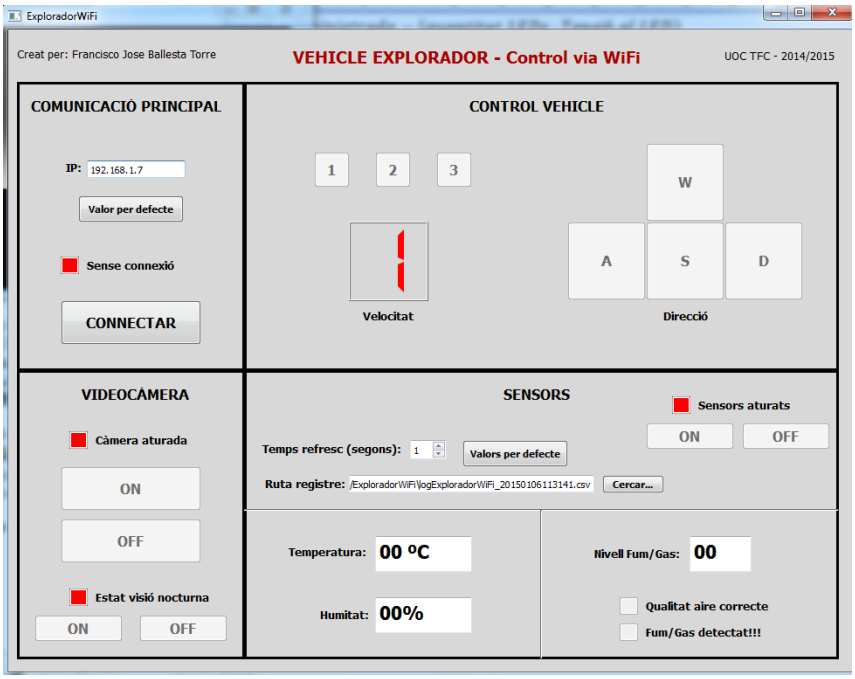
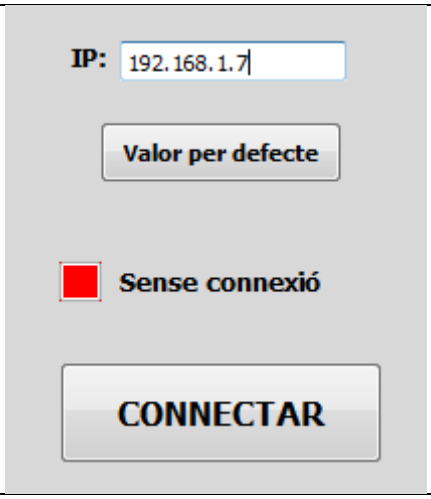


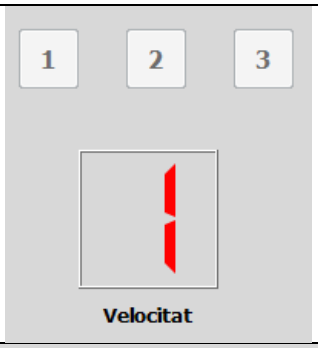

**Figura 44. Interfície gràfica d'usuari. Visió general.**

Aquesta GUI, es pot subdividir en quatre apartats generals, els quals es detallen a continuació.

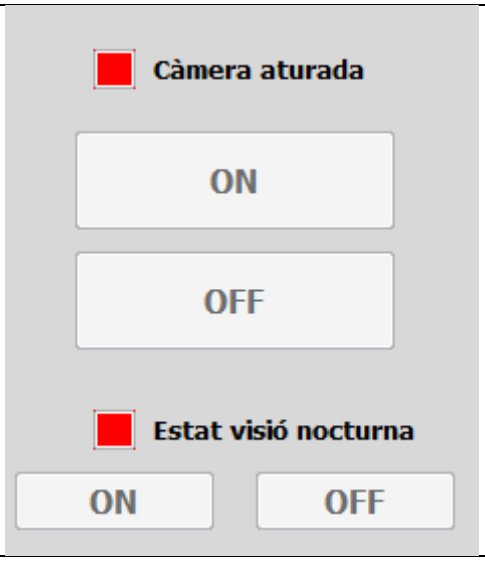
El primer que trobem, és la part de "**Comunicació Principal**", trobem els elements següents:

| | |
|---|---|
|  | Camp per indicar l'adreça IP de connexió del vehicle d'exploració.<br>Botó per tornar a posar al camp anterior, la IP per defecte (192.168.1.7).<br>Indicador d'estat de la connexió. Al connectar amb el vehicle, el quadrat es torna verd i les lletres indiquen "Connexió establerta".<br>Amb el botó "Connectar", s'establirà la connexió des de l'equip de control fins el socket que hi ha en mode "Listen" al port 10000 de la Raspberry Pi. |

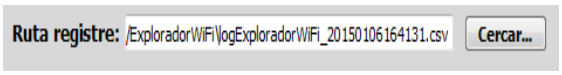
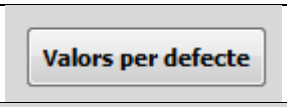
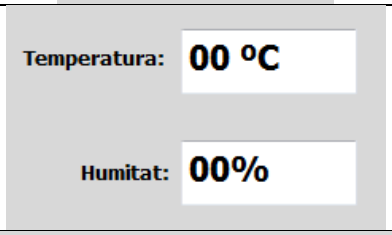
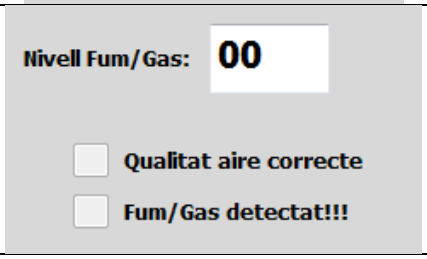
El següent apartat, és el que correspon al **control del vehicle**. Els seus elements son:

| | |
|---|---|
|  | Amb aquests tres botons, es selecciona la velocitat a la que anirà el vehicle. De 1 a 3, sent 1 la més lenta i 3 la més ràpida. Per defecte, sempre que s'inicia el vehicle, comença amb velocitat 1.<br><br>Just a sota dels botons, es troba un display on es mostra la velocitat que està activa en aquest moment. |
|  | En aquest apartat, es controlen els moviments del vehicle. Aquestes quatre lletres, tenen les funcions següents:<br><br>• W: Davant<br>• S: Darrere<br>• A: Esquerra<br>• D: Dreta |

Després, hi ha l'apartat que controla el sistema de visionat de la **videocàmera**, amb els següents elements:

| | |
|---|---|
|  | Indicador amb l'estat en que es troba la càmera. En engegar-la, el quadre es posa verd i les lletres canvien a "Càmera connectada".<br><br>Amb aquests dos botons (ON, OFF), s'engega i atura la càmera.<br><br>Indicador amb l'estat de la visió nocturna (LED's IR). En engegar-los, el quadre es posa verd.<br><br>Botons per engegar (ON) i aturar (OFF) la visió nocturna del vehicle (LED's IR). |

Per últim, tenim l'apartat del sistema de **sensors** del vehicle d'exploració, el qual incorpora els elements següents:

| | |
|---|---|
|  | Indicador amb l'estat dels sensors. En connectar-los, el quadre es posa verd i les text canvia a "Sensors connectats". |
| | Botons per engegar (ON) i aturar (OFF) els sensors del vehicle. |
|  | Camp per seleccionar els segons de refresc de les dades dels sensors, en un rang de 1 a 60 segons. |
|  | Camp per indicar la ubicació i nom del fitxer on es guardaran les dades obtingudes dels sensors. |
|  | Botó per tornar a posar la ruta del registre i el temps de refresc per defecte (1 segon). |
|  | Display on es mostra el valor actual que està detectant el sensor de temperatura en graus Celsius. |
| | Display on es mostra el valor actual que està detectant el sensor d'humitat en tant per cent. |
|  | Display on es mostra el valor actual que està detectant el sensor de fum/gas. |
| | Aquests elements, indiquen si s'ha detectat o no fum i/o gas en l'ambient. |

En el moment que tenim tot connectat i funcionant, podem tenir una imatge semblant a la que es mostra a continuació.

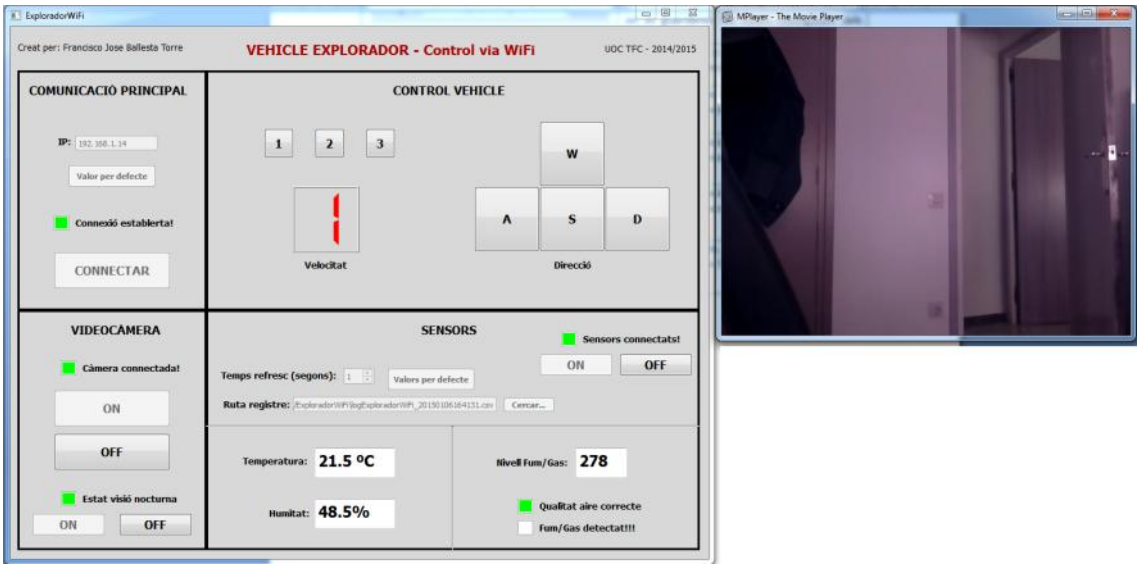



**Figura 45. Interfície gràfica d'usuari en funcionament, amb visió de càmera.**

Com és evident, per tal que aquesta GUI es pugui fer servir, la part del vehicle te que estar preparada per rebre una connexió del equip de control al fer clic en el botó "CONNECTAR". Per aquest motiu, en el moment que s'engega el vehicle i es posa en funcionament la Raspberry Pi, s'executa automàticament al inici el script principal "ServidorConsola.py", el qual crearà el socket d'escolta al port 10000 com s'ha comentat anteriorment.

Per poder fer que aquest script s'executi al inici de forma automàtica, s'ha tingut que modificar el fitxer de configuració "crontab" del programa **Cron** el qual va comprovant de forma periòdica el fitxer crontab, per saber quines comandes té que executar en funció del moment que sigui. Concretament, la línia que s'ha tingut que afegir al fitxer crontab, ha estat la següent:

**@reboot python /home/pi/ExploradorWiFi/ServidorConsola.py &**

Aquesta línia, significa que, cada cop que s'iniciï el sistema (**reboot**), ja sigui per engegada o reinici, s'executarà el script de python **ServidorConsola.py**, i ho farà en segon pla (**&**).

### 6.5.2.- Comunicació vehicle <--> Equip de control

Pel que fa a la comunicació entre l'estació base i el vehicle, s'ha volgut fer servir un sistema que fos **estàndard**, per tal que els diferents dispositius i plataformes puguin comunicar-se sense problemes, **flexible**, que existeixi al mercat un gran nombre d'equips i dispositius que permetin realitzar aquesta comunicació, i sobretot, **amb bona cobertura**, que permeti controlar el vehicle i rebre totes les dades a distancies relativament grans (en funció del material utilitzat).

Per aquests motius, s'ha escollit la comunicació WiFi, ja que es tracta d'un sistema àmpliament estès, amb un ventall de dispositius molt gran per crear les comunicacions (punts d'accés, receptors, antenes, etc.), i amb cobertures que poden arribar a quilometres, segons l'equip d'emissió i recepció utilitzat (antenes i punt d'accés) i segons l'entorn (obstacles, interferències, etc.).

Concretament, l'esquema de xarxa WiFi que es fa servir en aquest projecte, és molt senzill i econòmic, per aquest motiu, el rang de cobertura també serà bastant limitat (similar a la cobertura WiFi domèstica).

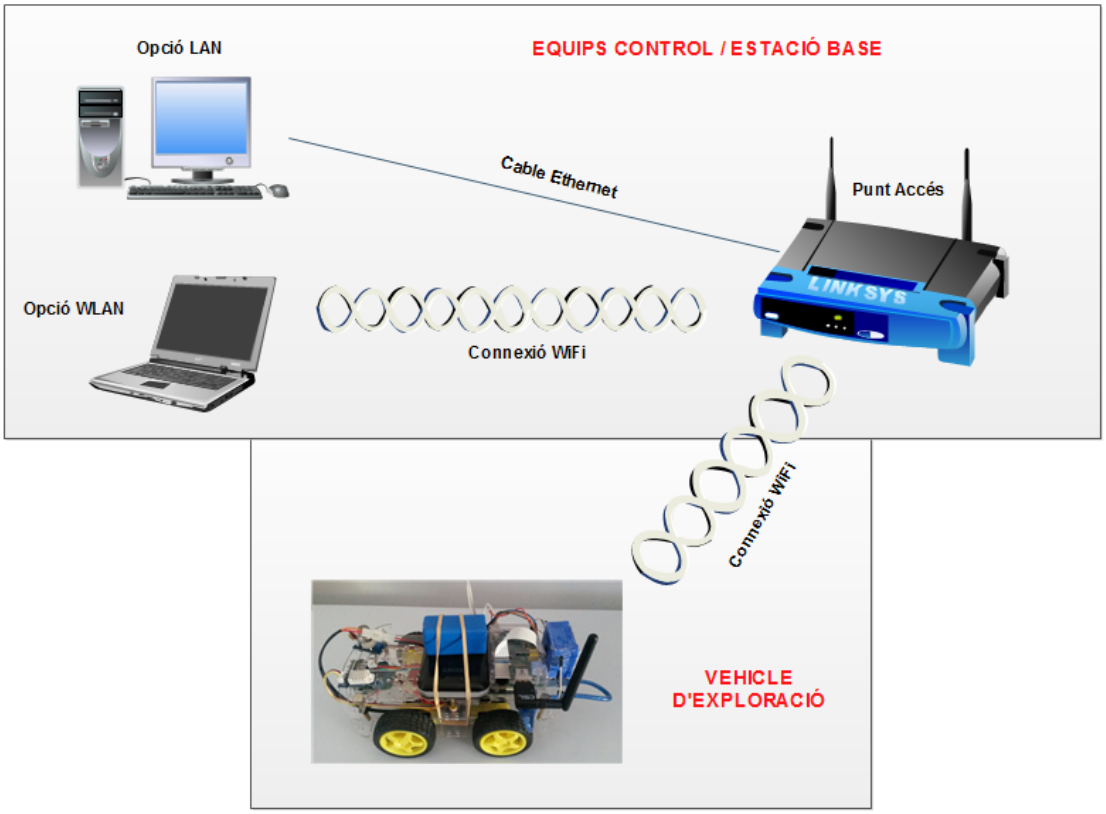



Figura 46. Esquema de la xarxa WiFi utilitzada.

Com es pot veure en l'esquema de la figura 46, per una banda disposem de l'equip de control del vehicle, que tant pot ser un equip connectat al punt d'accés via cable (ethernet), o via WiFi.

Per una altra banda, el vehicle d'exploració, es connectarà al punt d'accés mitjançant un dispositiu USB WiFi connectat a la Raspberry Pi. Com es veu a la imatge del vehicle, aquest dispositiu disposa d'una petita antena connectada al adaptador USB. En cas de voler millorar la cobertura a la part receptora (vehicle) es podria substituir l'antena que incorpora per una de més guany, com per exemple, la de la imatge següent, la quan proporciona 12 dBi de guany.

**Figura 47. Adaptador WiFi USB amb antena de 12 dBi.**

Pel que fa al punt d'accés utilitzat, es tracta d'un punt accés de la marca Linksys, model WRT54GL al qual se li ha aplicat el firmware lliure DD-WRT (www.dd-wrt.com).


**Figura 48. Punt d'accés Linksys WRT54GL.**

En l'apartat de configuració, cal destacar que el que s'ha fet, ha sigut habilitar una xarxa WiFi anomenada "**RC_TEST**" en el punt d'accés, amb xifrat WPA2-PSK i com que la IP del vehicle i del equip de control, s'han configurat de forma manual per evitar que canviïn, s'ha deshabilitat el servei de DHCP del punt d'accés.

En el cas de la configuració de la Raspberry Pi del vehicle, per tal que mantingués la configuració després de cada aturada, s'ha modificat el fitxer "**/etc/network/interfaces**" tal i com es mostra a l'Annex 6, en el que es pot veure, que la part de la configuració de la xarxa WiFi a la que es connecta, està definida en el fitxer de configuració del programa Wpa Supplicant, ubicat a la ruta "**/etc/wpa_supplicant/wpa_supplicant.conf**"

# 7.- Possibles millores/ampliacions

Després d'haver investigat sobre totes les possibilitats i utilitats que se li poden donar a aquest tipus de maquinari lliure, i després de veure la quantitat d'accessoris i dispositius que es poden fer servir en els projectes d'Arduino i/o Raspberry Pi, queda clar que les possibilitats son infinites.

En concret, per a aquest projecte hi ha varies coses que podrien millorar significativament alguna de les parts del sistema, ja sigui la part de software (GUI, scripts, etc.) o la part de hardware (vehicle, sensors, accessoris, etc.). A continuació hi ha un llistat amb les més significatives:

## Software

- Creació de gràfiques en temps real de les dades de sensors.
- Integració de la GUI en web per eliminar la necessitat d'utilitzar un equip concret amb el software instal·lat.
- Programar una interfície de control per dispositius mòbils.
- Integrar la pantalla de visualització de la càmera en la mateixa GUI de control, no en una finestra separada.

## Hardware

- Utilització de més tipus de sensors (pressió atmosfèrica, CO, CO2, micròfon, etc.).
- Utilització d'un sensor d'intensitat lumínica per realitzar l'encesa i aturada dels LED's IR, mantenint però, la possibilitat de fer-ho manualment.
- Incorporar un mòdul GPS ja sigui en Arduino o Raspberry Pi, per tenir el posicionament geogràfic.
- Incorporar un mòdul GPRS per tal de poder realitzar el control via Internet amb xarxes 3G o 4G.
- Utilització d'un punt d'accés amb antenes de llarg abast, per ampliar la distancia d'utilització.
- Incorporar una petita placa solar per carregar les bateries.

# 8.- Pressupost i viabilitat

## 8.1.- Pressupost vehicle exploració

Per tal de determinar la viabilitat o no d'aquest projecte en el món real, primer, és imprescindible analitzar els costos que té aquest projecte, tant pel que fa a la part material, com a la part de coneixement/mà d'obra necessària per a portar-ho a terme.

| Unitat | Concepte | Preu Unitari | Preu Total |
|---|---|---|---|
| 1 | Kit xassis + motors + rodes | 27,95 € | 27,95 € |
| 1 | Controladora motors | 5,50 € | 5,50 € |
| 1 | Bateria motors Li-Po 4S 2200mAh* | 15,00 € | 15,00 € |
| 1 | Interruptor motors | 1,30 € | 1,30 € |
| 1 | Bateria externa USB 15000mAh | 39,90 € | 39,90 € |
| 1 | Arduino UNO | 19,00 € | 19,00 € |
| 1 | Raspberry Pi Model B | 31,50 € | 31,50 € |
| 1 | Carcasa Raspberry Pi | 4,00 € | 4,00 € |
| 1 | Camera NoIR | 26,80 € | 26,80 € |
| 1 | Carcasa Camera NoIR | 3,00 € | 3,00 € |
| 6 | LED IR 150mW | 0,25 € | 1,50 € |
| 1 | Targeta SD 16GB | 11,95 € | 11,95 € |
| 1 | Dispositiu USB WiFi + Antena 12 dBi | 21,85 € | 21,85 € |
| 1 | Sensor DHT22 | 13,45 € | 13,45 € |
| 1 | Sensor MQ-2 | 9,75 € | 9,75 € |
| 1 | Chip convertidor A/D MCP3002 | 1,50 € | 1,50 € |
| 2 | Resistencia 27 ohms | 0,05 € | 0,10 € |
| 2 | Protoboard 400 contactes | 2,00 € | 4,00 € |
| 25 | Cables varis | 0,05 € | 1,25 € |
| **SUBTOTAL** | | | **239,30 €** |
| 1 | AP Linksys WRT54GL (Opcional) | 49,00 € | 49,00 € |
| **TOTAL AMB OPCIONALS** | | | **288,30 €** |

Figura 49. Pressupost material utilitzat.

*S'ha pressupostat una bateria del tipus Li-Po, les més freqüents en radiocontrol, de 14,8V i de 2200mAh, ja que la utilitzada en el projecte era reaprofitada.

En cas de voler reduir costos, es podria canviar la bateria externa USB per una d'autonomia inferior (segons les necessitats), i aprofitar algun punt d'accés existent o un més econòmic per fer la comunicació WiFi. Tenint en compte que és molt habitual disposar d'algun router amb WiFi o punt d'accés, aquest concepte s'ha marcat com a opcional al pressupost.

Un altre punt a tenir en compte, és que, en aquest pressupost, només s'ha comptabilitzat els materials que han sigut necessaris per fabricar el vehicle, però no altres materials com podrien ser les llicencies del software de desenvolupament, les hores dedicades a la investigació i desenvolupament del projecte inicial, etc. Els motius d'això, s'expliquen en el punt següent.

## 8.2.- Viabilitat i comercialització

Un cop calculat el cost total dels components utilitzats en el vehicle d'exploració, es pot començar a plantejar les possibilitats comercials que se li poden treure.

Bàsicament, aquest tipus de producte, estaria destinat a empreses/persones que tinguessin la necessitat d'explorar determinades zones, com a pas previ de l'accés de persones. Com per exemple, en situacions similars a les següents:

- Determinar l'existència de gasos perillosos en sistemes clavegueram, fabriques, o qualsevol altre lloc.
- Trobar fuites de gas en edificis o canalitzacions al aire lliure, amb risc d'explosió.
- Exploració visual d'un edifici amb danys estructurals i possibilitat d'ensorrament.
- Exploració de coves, mines, o altres indrets de difícil accés.
- I moltes altres.

En aquest cas, es podrien plantejar dos escenaris diferents de comercialització, en funció de com i perquè s'hagués desenvolupat aquest projecte. Es a dir, es podria donar el cas, que aquest projecte s'hagués desenvolupat per "voluntat pròpia" amb la intenció de treure el producte al mercat, o per una altra banda, que s'hagués realitzat com una petició expressa d'un client concret.

### Producte comercial (voluntat pròpia)

Si es tractés del primer cas, a part del preu del material calculat anteriorment, es tindria que ficar un marge de benefici per tal de rentabilitzar les hores dedicades al estudi inicial, les hores necessàries pel muntatge de cada vehicle que s'arribés a vendre, i els costos de mantenir les eines, software i altres costos varis de l'empresa per tal de poder tenir una activitat mercantil.

Dit això, un exemple de preu de mercat per aquest producte podria ser el següent:

- **Preu de materials = 288,30€**
  - *Si s'arribés a tenir varies comandes al mateix temps, els preus dels materials podria reduir-se en comprar més volum.*
- **Cost de preparació = 8hores x 15€/hora = 120€**
  - *S'ha tingut en compte 8h de muntatge del equip, ja que els procediments de muntatge i programes ja estarien estandarditzats per agilitzar la producció. A més, s'ha tingut en compte un possible preu/hora de tècnic informàtic/electrònic.*
- **Preu de venda al públic = 625€**
  - *El preu de venda s'ha marcat per intentar obtenir per cada unitat venuda un percentatge de beneficis del 35% aproximadament.*
- **Benefici per cada venda = 625 - 288,30 - 120 = 216,70€**

### Producte a mida d'un client concret

Com s'ha comentat, l'altre possibilitat, es que aquest projecte hagués sigut demanat expressament per un client concret, llavors, estaríem parlant d'un producte fet a mida que no es comercialitzarà. Per tant, en aquest cas, no s'avaluarà el preu del producte segons el benefici desitjat per unitat venuda, sinó es comptabilitzarà el cost de les hores dedicades al estudi i realització del projecte com a tal.

En aquest cas, el pressupost final del projecte seria com el següent:

- **Preu de materials = 288,30€**
    - *En aquest cas no es contempla la possibilitat d'estalviar costos, ja que, inicialment, només es realitzarà un sol vehicle.*
- **Cost d'estudi i preparació = 279 hores x 5€/hora = 1395€**
    - *S'ha tingut en compte una mitja de 3 hores diàries pels 93 dies que s'havia planificat inicialment en el projecte. A més, s'ha tingut en compte un preu/hora corresponent a un becari/estudiant de Telecomunicacions o Informàtica.*
- **Preu total del projecte = 1395 + 288,30 = 1683,3€**
    - *Aquest seria el cost total de cara al client, on s'ha evitat incrementar el preu del material, ja que el benefici del projecte resideix en la mà d'obra i no en el marge que es pugés treure del material.*
- **Benefici total = 1683,3 - 288,3 = 1395€**
    - Evidentment, com no hi ha marge de benefici en el material, els guanys d'aquest projecte correspondrien al cost del estudi. S'ha de tenir en compte però, que en cas d'haver tingut que adquirir algun tipus de llicencia o software utilitzat, els beneficis no serien del 100%, encara que en el cas del software es pot tractar com una inversió de futur.

## 9.- Conclusions

Com s'ha pogut veure durant tot el desenvolupament del projecte, el maquinari i software obert que es pot trobar avui en dia en el mercat, proporciona a l'usuari, ja sigui més o menys expert en l'àmbit de la programació, o tingui més o menys coneixements d'electrònica, un món de possibilitats gairebé il·limitades per poder crear qualsevol tipus de dispositiu.

Per una banda, s'ha vist com tots aquests equips (Arduino, Raspberry Pi, i altres dispositius similars) es poden relacionar entre ells de forma senzilla i efectiva mitjançant xarxes sense fils o amb connexió directa mitjançant ports USB que actuen com una connexió "serial", el que permet ampliar molt més les funcionalitats dels aparells creats.

Per una altra banda, pel que fa al tipus de connexió utilitzada en el projecte per tal de realitzar el control remot, s'ha pogut observar com les xarxes d'espectre eixamplat, ja sigui a 2,4 GHz o a 5 GHz (en aquest cas s'ha fet servir una xarxa a 2,4 GHz), permet interconnectar entre sí, diferents equips, i si es té en compte que cada cop més es tendeix a que els diferents aparells i màquines, ja siguin domèstics o d'ús professional, tinguin aquest tipus de connectivitat, ens obre un món en que tot el que ens envolta pugui ser controlat a distancia mitjançant un sol dispositiu.

## 10.- Bibliografia

**Informació general Arduino i Raspberry Pi**
http://arduino.cc/
http://www.raspberrypi.org/

**Freqüències habituals en RC**
http://es.wikipedia.org/wiki/Bandas_de_frecuencia
http://es.wikipedia.org/wiki/Emisora_radiocontrol
http://www.hooked-on-rc-airplanes.com/spread-spectrum.html

**Controladora motors**
http://blog.whatgeek.com.pt/arduino/l298-dual-h-bridge-motor-driver/

**Sensors DHT22 i MQ-2**
https://learn.adafruit.com/dht/overview
https://learn.adafruit.com/downloads/pdf/dht-humidity-sensing-on-raspberry-pi-with-gdocs-logging.pdf
http://playground.arduino.cc/Main/MQGasSensors
http://www.seeedstudio.com/wiki/Grove_-_Gas_Sensor%28MQ2%29

**Convertidor A/D MCP3002**
http://botbooks.com

**Programació en Python**
https://docs.python.org/3.4/#
http://learnpythonthehardway.org/book/index.html
https://wiki.python.org/moin/FrontPage

**Creació GUI Python**
http://doc.qt.io/qt-5/qtdesigner-manual.html
https://blog.safaribooksonline.com/2014/01/22/create-basic-gui-using-pyqt/

**Connexió i configuració càmera Raspberry Pi**
http://www.raspberrypi.org/help/camera-module-setup/
http://wiki.oz9aec.net/index.php/Raspberry_Pi_Camera

**Codificació resistències elèctriques.**
http://es.wikipedia.org/wiki/Codificaci%C3%B3n_de_colores

## ANNEX 1.- Codi pel control de motors i LED's IR (Arduino)

**Arduino: Fitxer ControlMotors.ino**

---

```
/*************************************************************/
/*  PROJECTE: TFC - Arduino+RPI: Vehicle RC d'exploració     */
/*  CODI: Control de motors i LED's IR                       */
/*  NOM:  Francisco Jose Ballesta Torre                      */
/*************************************************************/

/*************************************************************/
/*  Definició de les variables utilitzades pel driver de motors */
/*************************************************************/

#define LENT 80
#define NORMAL 100
#define RAPID 180

#define PIN_INA 6
#define PIN_INB 7
#define PIN_INC 8
#define PIN_IND 9
#define PIN_ENA 10
#define PIN_ENB 11

/*********************************************************/
/*  Definició de les variables utilitzades pels leds IR  */
/*********************************************************/

#define IR1 12
#define IR2 13

/*****************************************/
/*  Inicialització i declaració de variables  */
/*****************************************/

int MD1 = PIN_INA; // Control entrada/sortida A driver.
int MD2 = PIN_INB; // Control entrada/sortida B driver.
int ME1 = PIN_INC; // Control entrada/sortida C driver.
int ME2 = PIN_IND; // Control entrada/sortida D driver.
int VELD = PIN_ENA; // Control velocitat motors dreta.
int VELE = PIN_ENB; // Control velocitat motors esquerra.

int vel = LENT; //La velocitat del vehicle al iniciar-se, serà la definida a la variable "LENT".
```

```
/*****************************************/
/*  Configuració dels paràmetres inicials  */
/*****************************************/

void setup(){

  Serial.begin(57600);
  pinMode(MD1, OUTPUT);
  pinMode(MD2, OUTPUT);
  pinMode(ME1, OUTPUT);
  pinMode(ME2, OUTPUT);
  pinMode(VELD, OUTPUT);
  pinMode(VELE, OUTPUT);
  pinMode(IR1, OUTPUT);
  pinMode(IR2, OUTPUT);
}

/*************************************/
/*  Funcions de selecció de direcció  */
/*************************************/

void darrera(){
  marxa(vel);
  digitalWrite(MD1, LOW);
  digitalWrite(MD2, HIGH);
  digitalWrite(ME1, LOW);
  digitalWrite(ME2, HIGH);
}

void davant(){
  marxa(vel);
  digitalWrite(MD1, HIGH);
  digitalWrite(MD2, LOW);
  digitalWrite(ME1, HIGH);
  digitalWrite(ME2, LOW);
}

void esquerra(){
  marxa(vel);
  digitalWrite(MD1, HIGH);
  digitalWrite(MD2, LOW);
  digitalWrite(ME1, LOW);
  digitalWrite(ME2, HIGH);
}

void dreta(){
  marxa(vel);
  digitalWrite(MD1, LOW);
  digitalWrite(MD2, HIGH);
  digitalWrite(ME1, HIGH);
  digitalWrite(ME2, LOW);
}
```

```
/************************************************/
/*  Funcions per regular la velocitat dels motors  */
/************************************************/

void marxa(int v){
  analogWrite(VELD, v);
  analogWrite(VELE, v);
}

void aturar(){
  analogWrite(VELD, 0);
  analogWrite(VELE, 0);
  digitalWrite(MD1, LOW);
  digitalWrite(MD2, LOW);
  digitalWrite(ME1, LOW);
  digitalWrite(ME2, LOW);
}

/************************************/
/*  Funcions per regular els leds IR  */
/************************************/

void irON(){
  digitalWrite(IR1, HIGH);
  digitalWrite(IR2, HIGH);
}

void irOFF(){
  digitalWrite(IR1, LOW);
  digitalWrite(IR2, LOW);
}
```

```
/***********************************/
/*  Bucle principal d'instruccions  */
/***********************************/

void loop(){

  while(Serial.available()){
    char c = Serial.read();
    switch(c){
      case '1':
        vel = LENT;
        break;
      case '2':
        vel = NORMAL;
        break;
      case '3':
        vel = RAPID;
        break;
      case 'w':
        davant();
        break;
      case 's':
        darrera();
        break;
      case 'a':
        esquerra();
        break;
      case 'd':
        dreta();
        break;
      case 'p':
        aturar();
        break;
      case 'e':
        irON();
        break;
      case 'q':
        irOFF();
        break;
    }
  }
}
```

## ANNEX 2.- Codi del servidor per la consola/GUI (Raspberry Pi - Python)

**Raspberry Pi: Fitxer "ExploradorWiFi/ServidorConsola.py"**

---

```
#!/usr/bin/python

######################################################################
#       PROYECTE TFC - Arduino+RPI: Vehicle RC d'exploracio          #
#       CODI: Servidor per la consola de control (Python).           #
#       NOM: Francisco Jose Ballesta Torre                           #
######################################################################

import serial, socket, os, subprocess     #Importacio de llibreries necessaries.

arduino = serial.Serial('/dev/ttyACM0', 57600)  #Establiment de la connexio amb el Arduino.


########################################
#       Inicialitzacio de variables    #
########################################

ip = ''
port = 10000
camPort = '10002'
buffer = 20


############################################################
#       Creacio i configuracio del socket principal        #
############################################################

connexio = socket.socket(socket.AF_INET, socket.SOCK_STREAM)        #Creacio del socket.
connexio.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)  #Configuracio del
socket per poder-ho reutilitzar.
connexio.bind((ip, port))                                       #COnfiguracio de la IP i port del
socket.
connexio.listen(1)                                              #Inicialitzacio del socket en
mode Listen.
```

```
##################################
#         Bucle principal        #
##################################

while True:
  con_client, ip_client = connexio.accept()        #Acceptacio de la connexio des de l'equip de control.
  print 'Connexio des de la IP:', ip_client

  while True:
        ordre = con_client.recv(buffer)            #Recepcio de les ordres dades des de l'equip de control.
        if not ordre: break
        if ordre == '8':   #En rebre l'ordre '8', s'inicia un subproces amb el script del servidor dels sensors.
                senProc = subprocess.Popen(['/home/pi/ExploradorWiFi/ServidorSensors.py'])
        if ordre == '9':   #En rebre l'ordre '9', es finalitza el subproces iniciat dels sensors.
                senProc.kill()
        if ordre == '6':   #En rebre l'ordre '6', s'inicia un subproces amb el script de la camera.
                camProc = subprocess.Popen(['/home/pi/ExploradorWiFi/Camera.sh',ip_client[0],camPort])
        if ordre == '7':   #En rebre l'rdre '7', es finalitza el subproces iniciat de la camera i els programes relacionats.
                camProc.kill()
                subprocess.Popen(['sudo','pkill','raspivid'])
                subprocess.Popen(['sudo','pkill','nc'])
        else:                      #En rebre qualsevol ordre que no siguin les anteriors les envia al Arduino per tractar-les.
                arduino.write(ordre)

con_client.close()        #Es finalitza la connexio TCP amb l'estacio de control.
arduino.close() #Es finalitza la connexio serie amb Arduino.
```

## ANNEX 3.- Codi del servidor dels sensors (Raspberry Pi - Python)

**Raspberry Pi: Fitxer "ExploradorWiFi/ServidorSensors.py"**

---

```
#!/usr/bin/python

#########################################################################
#      PROYECTE TFC - Arduino+RPI: Vehicle RC d'exploracio     #
#      CODI: Servidor pels sensors del vehicle (Python).   #
#      NOM: Francisco Jose Ballesta Torre                  #
#########################################################################

########################################
#          Importacio de llibreries #
########################################

import sys, socket, os
import botbook_mcp3002 as mcp
import Adafruit_DHT

########################################
#          Inicialitzacio de variables          #
########################################

nivellGas = 0
DHT = 22
GPIO = 25
ip = ''
port = 10001
buffer = 20

#############################################################
#      Creacio i configuracio del socket principal     #
#############################################################

connexio = socket.socket(socket.AF_INET, socket.SOCK_STREAM)    #Creacio del socket.
connexio.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)  #Configuracio del socket
per poder-ho reutilitzar.
connexio.bind((ip, port))                       #COnfiguracio de la IP i port del socket.
connexio.listen(1)                              #Inicialitzacio del socket en mode Listen.
```

```
###################################
#      Bucle principal        #
###################################

while True:
  con_client, ip_client = connexio.accept()        #Acceptacio de connexio des de la consola de
control.
  print 'Connexio des de la IP:', ip_client

  while True:
        ordre = con_client.recv(buffer)
        if not ordre: break
        nivellGas = mcp.readAnalog()
        #Lectura del valor del sensor MQ-2.
        humitat, temp =Adafruit_DHT.read_retry(DHT, GPIO)
        #Lectura del valor del sensor DHT22.
        dades = str(round(temp,2)) + ";" + str(round(humitat,2)) + ";" + str(nivellGas)
        #Creacio d'un string unic amb els valors.
        con_client.send(dades)
        #Enviament del string a la consola de control.

con_client.close()        #Es finalitza la connexio TCP amb l'estacio de control.
```

## ANNEX 4.- Codi de la interfície gràfica d'usuari - GUI (Equip de control - Python)

**Equip de control: Fitxer "ExploradorWiFi/Consola_win.py"**

```
--------------------------------------------------------------------------
#########################################################################
#       PROYECTE TFC - Arduino+RPI: Vehicle RC d'exploracio     #
#       CODI: Consola de control principal (Python).          #
#       NOM: Francisco Jose Ballesta Torre                  #
#########################################################################

########################################
#       Importacio de llibreries       #
########################################
import socket, sys, time, queue, threading, datetime
from PyQt4 import QtCore, QtGui, uic

#####################################
#   Inicialització de variables     #
#####################################
ip = '192.168.1.7'
portControl = '10000'
portSen = '10001'
portCam = '10002'
BUFFER = '1024'
refrescSensors = 1000
pushActiu = QtGui.QPushButton
tipusRegistres = "Fitxer de text (*.txt);;Fitxer CSV (*.csv)"
fitxerRegistre                                                                  =
QtCore.QDir.currentPath()+"\logExploradorWiFi_"+datetime.datetime.now().strftime('%Y%m%
d%H%M%S')+".csv"
temp = '00 ºC'
hum = '00%'
gas = '00'
llindar = 300

form_class = uic.loadUiType("Consola_ui.ui")[0]     #Importacio de l'estructura visual del fitxer
Consola_ui.ui, creat amb Qt Designer.

class Consola_win(QtGui.QMainWindow, form_class):

   #Funcio principal del programa, la qual cridara les funcions corresponents segons l'accio que
es faci a la consola.
   def __init__(self, parent=None):
      QtGui.QMainWindow.__init__(self, parent)
      self.setupUi(self)
      self.pushCON.clicked.connect(self.connectar)
      self.pushValorCom.clicked.connect(self.valorDefecteCom)
      self.pushValorSen.clicked.connect(self.valorDefecteSen)
      self.pushCercar.clicked.connect(self.rutaRegistres)
      self.lineRuta.setText(fitxerRegistre)
      self.push1.clicked.connect(lambda: self.velocitat(b'1'))
      self.push2.clicked.connect(lambda: self.velocitat(b'2'))
```

```
        self.push3.clicked.connect(lambda: self.velocitat(b'3'))
        self.pushW.pressed.connect(lambda: self.direccio(b'w'))
        self.pushS.pressed.connect(lambda: self.direccio(b's'))
        self.pushA.pressed.connect(lambda: self.direccio(b'a'))
        self.pushD.pressed.connect(lambda: self.direccio(b'd'))
        self.pushW.released.connect(lambda: self.direccio(b'p'))
        self.pushS.released.connect(lambda: self.direccio(b'p'))
        self.pushA.released.connect(lambda: self.direccio(b'p'))
        self.pushD.released.connect(lambda: self.direccio(b'p'))
        self.pushONcam.clicked.connect(lambda: self.camera(b'6'))
        self.pushOFFcam.clicked.connect(lambda: self.camera(b'7'))
        self.pushONir.clicked.connect(lambda: self.ledsIr(b'e'))
        self.pushOFFir.clicked.connect(lambda: self.ledsIr(b'q'))
        self.pushONsen.clicked.connect(self.SenOn)
        self.pushOFFsen.clicked.connect(self.SenOff)

    #Funcio per detectar la utilitzacio de les tecles w,s,a,d,1,2 i/o 3 al teclat, per fer la mateixa
funcio que els botons de la consola.
    def keyPressEvent(self, event):
        global pushActiu
        if event.text() == 'w':
            pushActiu = self.pushW
            accio = "dir"
        elif event.text()== 's':
            pushActiu = self.pushS
            accio = "dir"
        elif event.text()== 'a':
            pushActiu = self.pushA
            accio = "dir"
        elif event.text()== 'd':
            pushActiu = self.pushD
            accio = "dir"
        elif event.text()== '1':
            pushActiu = self.push1
            accio = "vel"
        elif event.text()== '2':
            pushActiu = self.push2
            accio = "vel"
        elif event.text()== '3':
            pushActiu = self.push3
            accio = "vel"

        if accio == "dir":
            self.direccio(event.text().encode('UTF-8'))
        elif accio == "vel":
            self.velocitat(event.text().encode('UTF-8'))
        pushActiu.setDefault(True)

    #Funcio per detectar quan es deixa de premer una tecla del teclat i actuar en consequencia.
    def keyReleaseEvent(self, event):
        global pushActiu
        pushActiu.setDefault(False)
```

```
        self.direccio(b'p')

    #Funcio per tornar a establir la IP per defecte.
    def valorDefecteCom(self):
        self.lineIP.setText(ip)

    #Funcio per tornar a establir el path del fitxer de registre i el temps de refresc per defecte.
    def valorDefecteSen(self):
        self.lineRuta.setText(fitxerRegistre)
        self.spinTemps.setValue(refrescSensors/1000)

    #Funcio per realitzar la connexio principal amb el vehicle, i habilitar la resta de funcionalitats
de la consola.
    def connectar(self):
        global sControl
        sControl = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        ip = self.lineIP.text()
        sControl.connect((ip, int(portControl)))
        self.estatCom.setStyleSheet("background-color: lime")
        self.labelEstatCom.setText("Connexió establerta!")
        self.lineIP.setDisabled(True)
        self.pushValorCom.setDisabled(True)
        self.pushCON.setDisabled(True)
        self.velocitat(b'1')
        self.direccio(b'p')
        self.pushValorSen.setDisabled(False)
        self.push1.setDisabled(False)
        self.push2.setDisabled(False)
        self.push3.setDisabled(False)
        self.pushW.setDisabled(False)
        self.pushS.setDisabled(False)
        self.pushA.setDisabled(False)
        self.pushD.setDisabled(False)
        self.pushONcam.setDisabled(False)
        self.pushONsen.setDisabled(False)

    #Funcio per obrir una finestra d'explorador per determinar la ubicacio i nom del fitxer de
registre.
    def rutaRegistres(self):
        self.lineRuta.setText(QtGui.QFileDialog.getSaveFileName(self, "", None, tipusRegistres))

    #Funcio per evniar les ordres al vehicle per escollir velocitat i indicar-la al display.
    def velocitat(self, vel):
        self.lcdMarxa.display(int(vel))
        sControl.send(vel)

    #Funcio per enviar las ordres al vehicle per escollir direccio.
    def direccio(self, dir):
        sControl.send(dir)

    #Funcio per engegar o aturar la camera.
    def camera(self, cam):
```

```
        sControl.send(cam)
        if cam == b'6':
            self.estatCam.setStyleSheet("background-color: lime")
            self.labelEstatCam.setText("Càmera connectada!")
            self.pushONcam.setDisabled(True)
            self.pushONir.setDisabled(False)
            self.pushOFFcam.setDisabled(False)
        elif cam == b'7':
            self.pushONcam.setDisabled(False)
            self.pushOFFcam.setDisabled(True)
            self.pushONir.setDisabled(True)
            self.pushOFFir.setDisabled(True)
            self.ledsIr(b'q')
            self.estatCam.setStyleSheet("background-color: red")
            self.labelEstatCam.setText("Càmera aturada")

    #Funcio per engegar o aturar els LED's IR.
    def ledsIr(self, leds):
        sControl.send(leds)
        if leds == b'e':
            self.estatIr.setStyleSheet("background-color: lime")
            self.pushONir.setDisabled(True)
            self.pushOFFir.setDisabled(False)
        elif leds == b'q':
            self.estatIr.setStyleSheet("background-color: red")
            self.pushONir.setDisabled(False)
            self.pushOFFir.setDisabled(True)

    #Funcio per iniciar els sensors del vehicle.
    def SenOn(self):
        global sSensors
        global registre
        registre = open(self.lineRuta.text(),'a')
        sSensors = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        self.cronoSensors = QtCore.QTimer()
        self.cronoSensors.timeout.connect(self.dadesSensors)
        self.cronoSensors.start(self.spinTemps.value()*1000)
        self.spinTemps.setDisabled(True)
        self.pushValorSen.setDisabled(True)
        self.lineRuta.setDisabled(True)
        self.pushCercar.setDisabled(True)
        ip = self.lineIP.text()
        sControl.send(b'8')
        sSensors.connect((ip, int(portSen)))
        self.estatSen.setStyleSheet("background-color: lime")
        self.labelEstatSen.setText("Sensors connectats!")
        self.pushONsen.setDisabled(True)
        self.pushOFFsen.setDisabled(False)
        tSensors = threading.Thread(target=self.getDades)
        tSensors.start()

    #Funcio per aturar els sensors del vehicle.
```

```
    def SenOff(self):
        sControl.send(b'9')
        sSensors.close()
        registre.close()
        self.cronoSensors.stop()
        self.spinTemps.setDisabled(False)
        self.pushValorSen.setDisabled(False)
        self.lineRuta.setDisabled(False)
        self.pushCercar.setDisabled(False)
        self.pushONsen.setDisabled(False)
        self.pushOFFsen.setDisabled(True)
        self.estatSen.setStyleSheet("background-color: red")
        self.labelEstatSen.setText("Sensors aturats")
        self.textTemp.setText("00 ºC")
        self.textHum.setText("00%")
        self.textGas.setText("00")

    #Funcio per mostrar les dades dels sensors i desar-les al registre.
    def dadesSensors(self):
        self.textTemp.clear()
        self.textHum.clear()
        self.textGas.clear()
        self.textTemp.setText(temp)
        self.textHum.setText(hum)
        self.textGas.setText(gas)
        data = datetime.datetime.now().strftime('%Y-%m-%d %H:%M:%S')
        registre.write(data+';'+temp.strip(" ºC")+';'+hum.strip("%")+';'+gas+"\n")
        if int(gas) <= llindar and int(gas) != 00:
            self.estatGasNo.setStyleSheet("background-color: lime")
            self.estatGasSi.setStyleSheet("background-color: white")
        elif int(gas) > llindar:
            self.estatGasNo.setStyleSheet("background-color: white")
            self.estatGasSi.setStyleSheet("background-color: red")

    #Funcio per rebre les dades dels sensors i emmagatzemar-les en variables.
    def getDades(self):
        global temp
        global hum
        global gas
        while True:
            sSensors.send(b'1')
            dades = sSensors.recv(50)
            strDades = str(dades)
            temp = strDades.split(';')[0].strip("b', '') + ' ºC'
            hum = strDades.split(';')[1].strip("b', '') + '%'
            gas = strDades.split(';')[2].strip("b', '')

app = QtGui.QApplication(sys.argv)
win = Consola_win(None)
win.show()
app.exec_()
```

## ANNEX 5.- Comandes videocàmera (Raspberry Pi i Equip de control)

**Equip de control: Fitxer "clientCamera.bat" ubicat a la carpeta ExploradorWiFi\Camera**

---

.\Camera\netcat\nc64.exe -L -p 10002 | .\Camera\mplayer\mplayer.exe -vo direct3d -fps 31 -cache 512 -cache-min 5 -

**Raspberry Pi: Contingut script "Camera.sh" ubicat a la carpeta /home/pi/ExploradorWiFi**

---

```
#!/bin/sh

sudo raspivid -t 999999 -w 640 -h 480 -o - | nc $1 $2
```

## ANNEX 6.- Configuració WiFi al vehicle (Raspberry Pi)

**Raspberry Pi: Fitxer "/etc/network/interfaces"**

---

```
auto lo

iface lo inet loopback

iface eth0 inet static
        address 192.168.1.14
        netmask 255.255.255.0
        gateway 192.168.1.1

allow-hotplug wlan0
iface wlan0 inet manual
wpa-roam /etc/wpa_supplicant/wpa_supplicant.conf

iface RC_TEST inet static
        address 192.168.1.7
        netmask 255.255.255.0
        gateway 192.168.1.1
```

**Raspberry Pi: Fitxer "/etc/wpa_supplicant/wpa_wupplicant.conf"**

---

```
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
network={
        ssid="RC_TEST"
        proto=RSN
        key_mgmt=WPA-PSK
        pairwise=CCMP
        group=CCMP
        auth_alg=OPEN
        psk="#UOCrcTFC#"
        id_str="RC_TEST"
}
```

## ANNEX 7.- Configuració inici automàtic (Raspberry Pi)

**Raspberry Pi: Contingut Crontab --> "sudo crontab -e"**

---

```
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command
```

**@reboot python /home/pi/ExploradorWiFi/ServidorConsola.py &**

## ANNEX 8.- Configuració inici programa principal (Equip control)

**Equip de control: Fitxer "ExploradorWiFi.bat" ubicat a la carpeta ExploradorWiFi**

---

```
START /b .\Camera\clientCamera.bat
START /b .\Consola_win.py
```