Universitat de Girona
Escola Politècnica Superior

Treball final de grau

Estudi: Grau en Enginyeria Electrònica Industrial i Automàtica

Títol: Control de maniobres d'un robot mòbil

Document: 1. Memòria

Alumne: Narcís Casellas Arbat

Tutor: Lluís Pacheco
Departament: Enginyeria Elèctrica, Electrònica i Automàtica
Àrea: ESA

Convocatòria (mes/any): gener/2016

ÍNDEX

1. INTRODUCCIÓ

El control de maniobres és molt utilitzat per planificar trajectòries i per fer-ho és fonamental conèixer el model cinemàtic i dinàmic del robot. La trajectòria realitzada depèn clarament de les limitacions físiques del robot.

1.1 Antecedents

Per una banda, els últims avenços tecnològics pel que fa al sector automobilístic estan relacionats amb la conducció autònoma. D'aquesta manera es poden evitar accidents, facilitar la conducció per a persones discapacitades o bé perdre menys temps al volant. Dins d'aquests es podrien destacar la conducció autònoma en embussos desenvolupada per Volvo, el pilotatge automàtic de Nissan i Mercedes i l'aparcament automàtic d'Audi.

Per l'altra banda, l'estri per aconseguir aquests avenços és la robòtica. La qual és una branca de l'enginyeria electrònica industrial que enllaça molts coneixements diversos, no només electrònics. Entre els quals cal destacar la selecció d'actuadors, sistemes de percepció i sensorització, sistemes de control, informàtica i mecànica.

1.2 Objecte

En aquest projecte ens centrarem en aquesta última innovació on el conductor és capaç de baixar de l'automòbil i aparcar-lo a través d'una aplicació de l'Smartphone mitjançant sensors i càmeres de vídeo. Amb aquest nou sistema d'aparcament es pot aconseguir aparcar en places en què ara, per qüestió d'espai, no es podria sortir del cotxe o entrar-hi.

Per simular el control de maniobres en una plaça d'aparcament s'utilitzaran els coneixements en robòtica mòbil amb l'objectiu de manipular un cotxe radio-control a través d'un dispositiu Android. Per poder comunicar el robot mòbil amb el telèfon mòbil s'utilitzarà una placa d'Arduino que es comunicaran a través de Bluetooth. A més es crearà una aplicació personalitzada segons els requeriments del projecte disponible per Smartphones amb sistema operatiu Android.

1.3 Especificacions i abast

Entre les tasques que es realitzaran, primerament, s'haurà de fer un estudi del sistema per tal d'incorporar al cotxe els sensors necessaris per poder controlar els seus moviments. Una vegada el sistema estigui adequat per treballar s'hauran d'adquirir les dades per obtenir els models i poder realitzar una regulació de la posició.

Pel que fa a la part més practica s'haurà de crear el programa de l'Arduino, l'aplicació per a mòbils amb SO Android i realitzar-ne la comunicació.

2. CARACTERÍSTIQUES DEL ROBOT MÒBIL

Els robots mòbils són un dels centres importants en la investigació actual de la robòtica degut en part a la baixada de preus del hardware necessari per a la seva construcció. Es caracteritzen per tenir gran capacitat de desplaçament dotat d'un sistema motriu amb rodes. Són capaços de moure's en el seu entorn i no tenen una base fixa.

En els següents capítols es descriu el funcionament del robot i les modificacions realitzades. És a dir tot allò que fa referència a l'elecció del material utilitzat com ara actuadors, sensors i estructura del robot.

El robot mòbil que s'utilitzarà és un cotxe RC de joguina de quatre rodes, no-holonomic i amb tracció diferencial. S'ha de tenir present que a l'utilitzar una un cotxe de joguina podran sorgir alguns problemes ja que els motors i els engranatges es dissenyen per aconseguir velocitats relativament elevades. Per tant es podran produir problemes de control ja que per realitzar l'aparcament ho ha de fer a una velocitat baixa

2.1 Estructura

Tal com s'ha esmentat anteriorment el projecte parteix d'un cotxe radio-control més concretament l'edició Subaru Impreza WRC de la casa Bycmo.

Les dimensions del qual són 36 cm de llarg i 18,5 cm d'amplada. El material de la base principal de subjecció és d'alumini de 0,2 mm de gruix.

Les suspensions utilitzades són a través de molles per tal de assegurar en tot moment el contacte de les quatre rodes amb una superfície plana o amb poc relleu.

2.2 Classificació segons els graus de llibertat

Els robots es poden classificar en dos tipus segons la mobilitat, les quals són holonomics i no-holonomics. Un robot és holonomic quan la quantitat de graus de llibertat que es poden controlar és igual a la quantitat de graus de llibertat disponible, en altres paraules són aquells capaços de girar en qualsevol direcció de manera instantània. A diferència d'el cas anterior, en els no-holonomics es poden controlar menys graus de llibertat que els que té

disponibles. Si ho apliquem al cas d'un cotxe significa que no pot girar en qualsevol direcció de manera instantània. Per tant si el cotxe inicialment esta en posició horitzontal i vols que es mogui de manera horitzontal primer s'hauran de fer una sèrie de maniobres. En canvi els holonòmics podrien canviar de direcció de manera pràcticament automàtica. A la seguent figura es pot veure de manera gàfica la principal diferència entre ells.

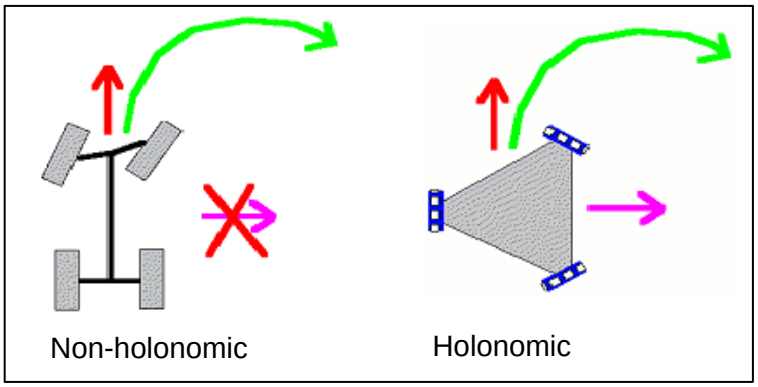


Figura 1. Classificació segons la direcció

En el cas d'aquest projecte el robot utilitzat és no-holonomic per tal de poder fer una simulació més aproximada d'un cotxe real.

2.3 Transmissió

Com el seu nom indica la funció principal és transmetre el moviment de gir provinent del motor als eixos de les rodes i a la vegada es pot generar una reducció o amplificació del moviment. En els robots mòbils generalment es vol reduir la velocitat ja que el motor gira a una velocitat bastant elevada, per tant, l'engranatge que rep el moviment del motor ha de tenir un diàmetre inferior per tant menys nombre de dents que l'engranatge de transmissió o corona.

La característica principal de la transmissió diferencial, que és la utilitzada en aquest projecte, és que pot compensar les diferents velocitats que es produeixen a les rodes en recórrer una corba. La roda que recorre la part exterior ha de recórrer un camí mes llarg que la de la part interior de la corba. Això és molt útil per tal d'evitar que les rodes patinin i aconseguir una millor estabilitat, per tant la velocitat de la roda exterior serà més gran que la interior.

Tant des del punt de vista d'adquisició de dades com en el de programació el disseny diferencial amb dos motors és un dels sistemes de locomoció menys complicats. Tot i que el nostre robot disposa de transmissió diferencial, en aquest cas el sistema motriu està compost per un únic motor.

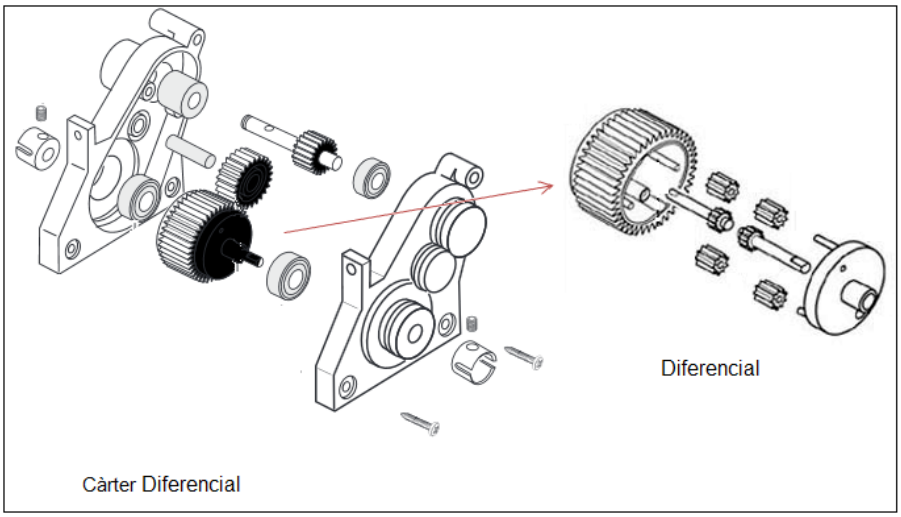


Figura 2. Transmissió

A la figura anterior es pot observar com està format el càrter diferencial i el propi diferencial. Aquest diferencial utilitza un engranatge planetari, és a dir consta de 4 engranatges exteriors que roden sobre el central.

2.4 Actuadors

Els actuadors elèctrics són els dispositius encarregats de transformar l'energia elèctrica per generar moviment. Per optimitzar aquest projecte s'han utilitat tres actuadors, un motor i dos servo-motors.

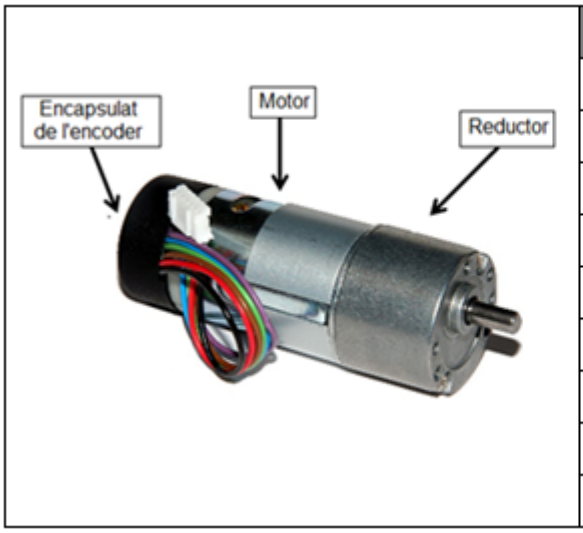
2.4.1 El motor

El robot disposa de només un motor elèctric el qual és el responsable de generar-ne la força motriu. La tracció del cotxe és a les rodes de darrera i és la mateixa força per les dues rodes.

Per aquets projecte s'ha canviat el motor original del cotxe bàsicament per dos motius principals. El primer és que s'ha utilitzat un motor reductor per tal de poder obtenir velocitats més favorables per la seva funció, el segon i més important, és per poder conèixer la posició del cotxe, és a dir s'ha utilitzat un motor amb encoder. El disseny del cotxe tenia la corona més grossa que el pinyó per tal de reduir la velocitat del motor original. Al canviar el motor per un amb reductor incorporat s'ha observat que el moviment era massa lent per això s'han hagut d'intercanviar el pinyó per la corona per obtenir una relació de transmissió correcta.

El diàmetre de l'eix del motor nou és més gran que l'antic, per tant el pinyó té un diàmetre inferior. Per solucionar aquest problema es va haver d'utilitzar un torn per tal de fer un nou forat centrar amb el diàmetre adequat. A més s'hi va fer un forat transversa per poder-lo collar a través d'un espàrrec allen especial.

A la següent figura es pot veure tant l'aspecte físic del motor, on s'hi pot veure clarament el reductor i l'encoder, com la taula amb les especificacions.



Model EMG30	
Tensió màxima nominal	12 VDC
Càrrega nominal	1.5 Kg/cm
Velocitat nominal	170 rpm
Consum nominal	530 mA
Velocitat sense càrrega	216 rpm
Consum sense càrrega	150 mA
Polsos de l'encoder	90 polsos/volta
Reductor	30:1
Diàmetre de l'eix	5 mm

Figura 3. Motor EMG30

El reductor que ens proporciona el motor redueix la velocitat de l'eix principal 30 vegades obtenint un eix excèntric. Gràcies a això podem aconseguir una velocitat adequada per poder llegir els impulsos amb una resolució de 90 per volta de l'eix de sortida reductora. A més el reductor també ens permet augmentar el parell que ens proporcionaria el motor.

El connector té 6 pins, els dos primers per alimentar el motor a 12VDC, dos més amb el mateix potencial per l'encoder i els dos últims amb els quals en rebem els impulsos. Aquests

dos ens proporciona dos senyals desfasats 90º per poder determinar el sentit de gir del motor. S'ha de tenir en compte que els impulsos que ens dóna la velocitat que obtenim és la de l'eix de sortida del motor un cop ha passat pel reductor per tant no s'ha de dividir per 30.

2.4.2 El servomotor

Els servos són dispositius caracteritzats per poder-los posicionar en qualsevol dels seus angles enviant-li una senyal codificada de PWM. Es disposen de dos servomotors de 3.5 Kg·cm un per poder controlar la direcció i l'altre pel motor.

A la figura següent es pot veure l'aspecte del servo i una petita descripció de les especificacions.

Model Bycmo 2HC	
Voltatge d'alimentació	12 VDC
Rang de control PWM	0-255
Càrrega	3.5Kg·cm

Figura 4. Servomotor Bycmo 2ch

El servo del motor regula el voltatge d'entrada de -7.2 V a 7.2 V per tal de proporcionar al cotxe dues direccions, endavant i endarrere a través del variador de velocitat. El disseny del circuit elèctric del variador permet obtenir tres velocitats amb una transició suau entre elles.

2.5 Sensors

Un transductor és un dispositiu que converteix una senyal d'un tipus d'energia a un altre (mecàniques, òptiques, elèctriques...). En robòtica un sensor utilitza un transductor que capta informació de l'entorn per ser utilitzada en el robot.

Per saber la direcció de les rodes davanteres s'ha utilitzat un potenciòmetre lineal que s'ha acoblat amb una peça de fabricació pròpia a l'eix del servomotor, la qual es pot trobar als plànols. La placa d'Arduino l'alimenta a 12 VDC i envia el valor obtingut del potenciòmetre a una entrada analògica.

També es disposa de l'encoder del motor que ja s'ha comentat a l'apartat anterior que és imprescindible per la lectura de la velocitat i posició. Aquest disposa d'un condensador que actua com a filtre per tal de reduir el soroll. Els encoders son transductors rotatius que transformen un moviment angular en un seguit d'impulsos digitals.

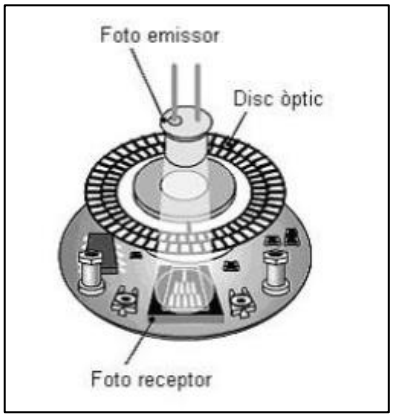


Figura 5. Esquema intern de l'encoder

A la figura anterior podem observar un foto emissor i un de receptor que projecten una llum als dos discs òptics que hi ha. Aquests dos estan desfasats 90º proporcionant-nos dues ones quadrades anomenats canal A i canal B. La lectura de qualsevol dels dos anals ens proporciona la velocitat de gir i si relacionem els dos podem saber el sentit de rotació. La precisió de l'encoder depèn de factors mecànic i elèctrics com ara el fregament dels rodaments o imprecisions del disc òptic.

Per les sortides dels senyals A i B s'ha d'incorporar una resistència Pull-Up per tal d'evitar possibles sorolls elèctrics ja que això ens faria caure el valor de l'impuls en un rang indefinit, el qual no ens permetria saber si està a 0 o a 1.

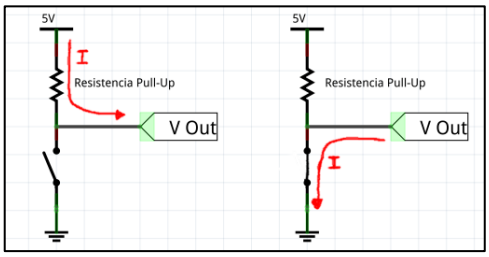


Figura 6. Esquema Pull-Up

D'aquesta manera quan l'interruptor està obert la corrent elèctrica va des de la font d'alimentació a la sortida, donant un 1. En canvi quan l'interruptor està tancat la corrent es mou cap a terra deixant un 0 a la sortida.

Per poder percebre la proximitat d'obstacles seria de gran utilitat incorporar sensors de proximitat però no s'han implementat al projecte per tal de centrar-lo en el control del moviment. Per tant la simulació de l'aparcament es farà amb unes condicions inicials de posició preestablertes i conegudes.

2.6 Hardware

En la següent figura podem observar com intervenen els principals dispositius que s'utilitzen en el projecte. El centre de control és l'Arduino UNO, el qual és l'enllaç entre el robot i l'usuari que el controla.

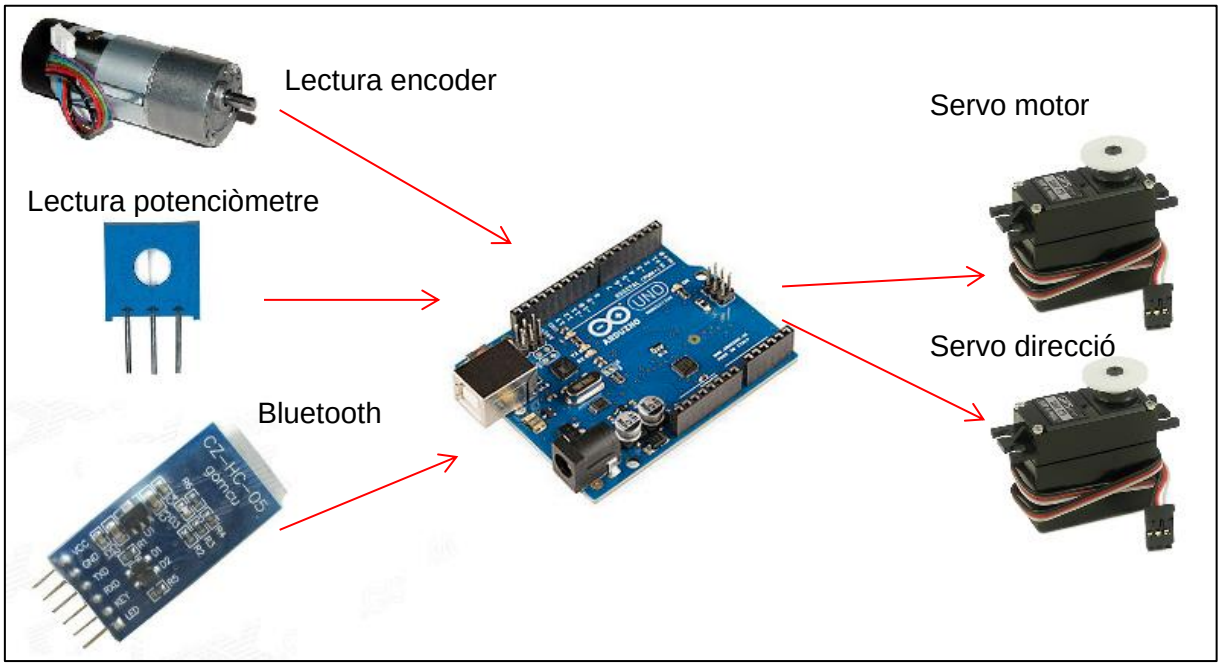


Figura 7. Interacció del hardware

2.6.1 Placa Arduino UNO

Actualment hi ha molts microcontroladors diferents útils per a l'aplicació que es necessita fer, però s'ha elegit l'Arduino UNO per la seva senzillesa.

L'Arduino és una plataforma de hardware lliure basada en un microcontrolador i en un entorn de desenvolupament dissenyat especialment per facilitar la seva utilització en projectes electrònics. Hi ha diversos models, entre els quals s'hi troba l'Arduino UNO. El seu hardware consisteix en un microcontrolador Atmega 328 i ports d'entrada i sortida. Pel que fa al

software que utilitza, és un programa lliure basat en el llenguatge de programació Processing el qual disposa d'un emulador de càrrega executable.

Les especificacions que el carcteritzen són les següents:

Arduino UNO	
Microcontrolador	Atmega 328
Voltatge d'entrada	7-12V
Voltatge de sortida	5V
Freqüència del rellotge	16MHz
E/S digitals	14
Entrades analògiques	6
PWM	6
Memòria Flash	32Kb

Figura 8. Arduino UNO

Com es pot observar a la figura 7 es pot alimentar a un voltatge d'entrada de 7 a 12V via USB o amb un connector jack. Mentre es realitza la programació i les emulacions és més pràctic utilitzar el port USB però quan es necessita autonomia es pot fer a través d'una bateria o piles gràcies al port jack.

La placa disposa de 14 pins digitals que es poden configurar com a entrades o sortides, 6 dels quals són configurables com a sortides de PWM. La freqüència del rellotge és de 16MHz, tot i que es pot escalar si es programa el microcontrolador Atmega. La intensitat dels pins d'entrades i sortides és de 20mA i la de la sortida de 3.3V és de 50mA

L'Arduino és l'encarregat d'interpretar i executar les accions realitzades per l'usuari a partir de l'aplicació del dispositiu Android. Per fer-ho es comunicarà a través d'un mòdul de Bluetooth i interactuarà amb el robot a través dels sensor i accionadors mostrats a la figura 6.

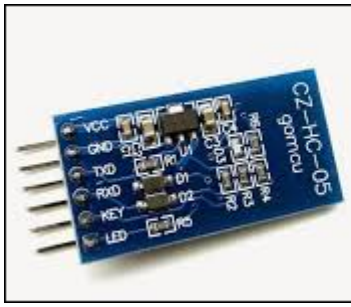
Per enllaçar els diferents dispositius s'ha utilitzat una Perfboard. És una placa de circuit perforada, els forats dels quals estan envoltats per un material conductor, normalment coure, però que no estan connectats entre ells. Aquests tipus de plaques requereixen que cada component estigui soldat entre ells i a més les interconnexions entre ells es realitzen a

través de cables o camins de soldadura. Per connectar-los s'han utilitzat tires de pins mascles o femelles depenent de cada connector tal i com es pot observar al plànol del document 2 corresponent.

2.6.2 Mòdul Bluetooth

Per controlar els moviments del robot mòbil es farà la programació mitjançant el mòdul Bluetooth que rebrà les accions provinents de l'aplicació de smartphone i les transferirà a la placa Arduino UNO.

El dispositiu utilitzat és el mòdul HC-05 del qual podem veure les característiques a continuació:



Bluetooth HC-05	
Voltatge d'alimentació	5V
Versió de Bluetooth	2.0
Mode de configuració	Comandes AT (port sèrie)
Temperatura de treball	-20ºC a +75ºC
Intensitat màxima	30mA

Figura 9. Bluetooth HC-05

Com es reflecteix a la figura anterior, s'alimenta a 5V de corrent continu i amb una intensitat màxima de 30 mA. El dispositiu es pot configurar com a mestre o esclau depenent de les seves necessitats, en aquest cas s'ha programat a través de les comandes AT com a esclau ja que només s'utilitza per rebre ordres i transferir-les a l'Arduino.

S'ha escollit aquest mòdul per diverses raons, entre les quals destaca el seu econòmic preu i compleix a la perfecció la utilitat que li volem donar.

Disposa de 6 pins, dos dels quals són per l'alimentació, dos més per enviar i rebre les dades amb la placa de control, el pin KEY que serveix per bloquejar la seva configuració i un últim pin per encendre un led de comunicació.

La connexió a realitzar entre el mòdul i la placa de control és la següent:

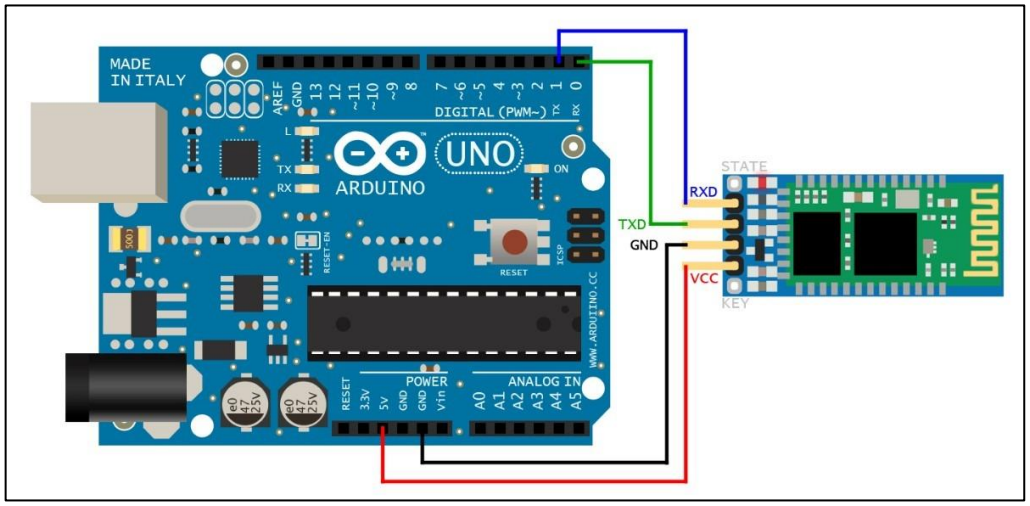


Figura 10. Connexió del mòdul HC-05

A la imatge 10 es mostra que el pin RXD es connecta al TX de l'arduino i el TXD del mòdul es conecta al pin RX de la placa de control. Això és perquè la dada que envia el mòdul l'arduino la rebrà al pin RX i per altra banda les dades enviades per l'Arduino a través del TX les rebrà el mòdul al pin RXD.

Per programar la configuració del mòdul es fan servir les comandes AT. Algunes d'aquestes comandes són: AT+NAMENARCIS per canviar el nom, AT+BAUD4 per establir la velocitat de comunicació de 9600bps i AT+ROLE0 per configurar-lo com a esclau. La configuració de la comunicació Bluetooth a través de l'Arduino.

3. CONTROLADOR

3.1 Adquisició de les dades

Abans de poder realitzar el control del cotxe el primer que s'ha de fer és obtenir un model del sistema. Per fer-ho cal adquirir les dades dels sistemes que es volen controlar, que són la direcció dreta, l'esquerra, el desplaçament cap endavant i endarrere.

Per obtenir una resposta del sistema en aplicar una consigna, s'han assignat dos valors de PWM a l'entrada del sistema en instants de temps aleatoris. Això s'anomena PRBS i és una seqüencia de valors binaris de temps aleatori. D'aquesta manera es pot veure com reacciona el sistema davant del valor d'entrada. Per fer-ho més fàcil d'entendre tot seguit es mostrarà una figura molt representativa:

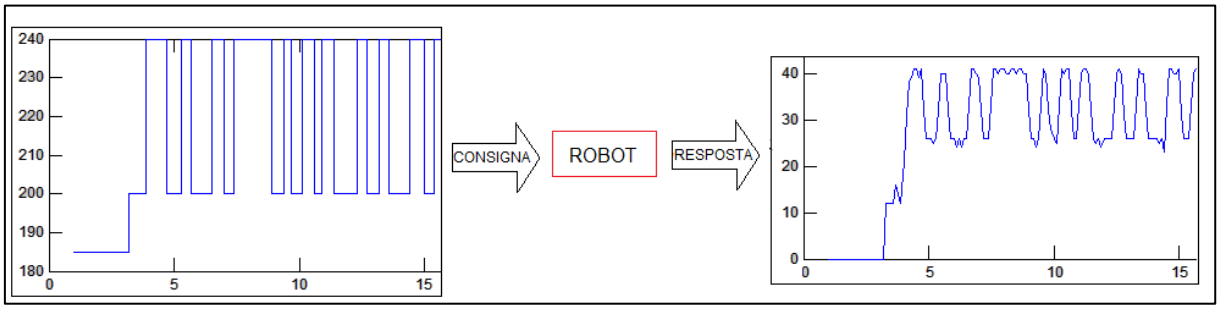


Figura 11. Sistema d'adquisició de dades aleatòries

A l'esquerra de la figura es mostra la consigna d'entrada, com es pot observar se li apliquen dos valors en el qual el temps és diferent en cada instant. D'aquesta manera a la dreta es pot observar la sortida en les unitats físiques corresponents. Així es pot veure la reacció de l'actuador i es podrà aconseguir un model aproximat del sistema.

Per enregistrar les dades s'han utilitzat dos programes, l'Arduino i el Labview, cadascun amb les modificacions pertinents depenent de les unitats de sortida que volem obtenir. L'Arduino s'ha utilitzat per poder llegir les dades i el Labview per processar-les i guardar-les en una taula. A continuació es descriuran les modificacions realitzades per a cada adquisició.

En el cas de la direcció per poder saber els graus de gir s'ha utilitzat un potenciòmetre acoblat al servomotor de la direcció. És important que el potenciòmetre sigui directament lineal així, d'aquesta manera l'angle girat pel servo indica de manera proporcional la posició

de la direcció. L'únic que cal fer és escalar l'entrada analògica amb la funció "map" de l'Arduino, ja que rebem per l'entrada analògica un valor de 0 a 1023 i ho hem de convertir a unitats físiques.

Per l'adquisició de la velocitat del motor s'han de llegir les pulsacions de l'encoder que hi ha integrat al motor EMG30. La dificultat per fer-ho és que l'eix de l'encoder gira tant ràpid que s'obtenen moltes pulsacions (unes 120 pulsacions per segon). Això dificulta la tasca de la lectura ja que no es pot utilitzar el PIN2 que proporciona la placa d'Arduino que genera una interrupció cada vegada que detecta un flanc, ja que sinó empitjoraria molt el rendiment del procés. Per solucionar aquest problema s'ha optat per programar directament sobre la placa d'ATMEGA ja que aquesta disposa de tres pins que es poden utilitzar com a temporitzador i comptador. Aquests pins s'han utilitzat per comptar els flancs d'entrada i també disposa d'una llibreria pròpia amb un llenguatge propi, diferent que el d'Arduino. D'aquesta manera, amb el programa que hi ha l'annex es poden llegir els impulsos de rotació del motor. El següent pas és convertir-los en velocitat angular i posteriorment en velocitat lineal.

Per tal de convertir els impulsos en revolucions per minut primer s'han de saber els pulsos que es realitzen per cada volta del motor, de manera que manualment s'ha comprovat que fent girar el motor obtenim 90 impulsos. Per tant, si coneixem els impulsos obtinguts cada segon es poden saber les voltes per minut. Un cop arribat a aquest punt, per saber la velocitat lineal primer s'ha de tenir en compte la relació de transmissió del cotxe, ja que la força rotativa de l'eix del motor no va directament a les rodes sinó que, com ja s'ha explicat anteriorment, té transmissió diferencial i per tant engranatges que s'han de tenir en compte. Exactament hi ha una corona, un pinyó i la caixa de transmissió de la qual ja en sabem la relació de transmissió entre el seu pinyó i la sortida de les rodes

Per saber la relació de transmissió entre pinyons influeixen dos paràmetres, el diàmetre (D_i) i la velocitat de gir (n_i). I l'expressió que els relaciona és la següent:

$$n_1 * D_1 = n2 * D_2 \qquad \text{(Eq. 1)}$$

Per tant, si tenim el diàmetre i la velocitat de gir del pinyó motriu podem conèixer la velocitat de la corona de la manera que es mostra a continuació:

$$n_1 * 1.5 = n2 * 4.4 \qquad \text{(Eq. 2)}$$

Aïllem la variable n_2:

$$n_2 = \frac{(n_1 * 1.5)}{4.4} \qquad \text{(Eq. 3)}$$

Tot seguit sabem que la relació de transmissió de la caixa diferencial és de 2.75 això significa que per cada dues voltes i tres quarts de la corona d'entrada a la caixa de transmissió la roda farà una volta. Per tant dividint la velocitat n_2 entre 2.75 es pot obtenir la velocitat angular de la tracció de les rodes. Per calcular la velocitat lineal primer cal convertir la velocitat angular en radians per segon multiplicant per 2*π i dividint entre 60. Finalment per obtenir la velocitat lineal cal aplicar la següent expressió:

$$v = w * r \qquad \text{(Eq. 4)}$$

Ara que ja s'ha explicat com adquirir les dades necessàries s'explicarà com s'ha processat a través del software de Labview. Primer de tot cal mostrar la pantalla d'interfície de l'usuari, la qual s'utilitza per guardar les dades en el moment oportú. La interfície consta dues pestanyes, la inicial serveix per configurar les dades de comunicació entre l'Arduino i el PC. La segona és on es visualitzen les dades i on l'usuari es pot comunicar amb el cotxe. A continuació es mostraran les dues pantalles amb la descripció corresponent.

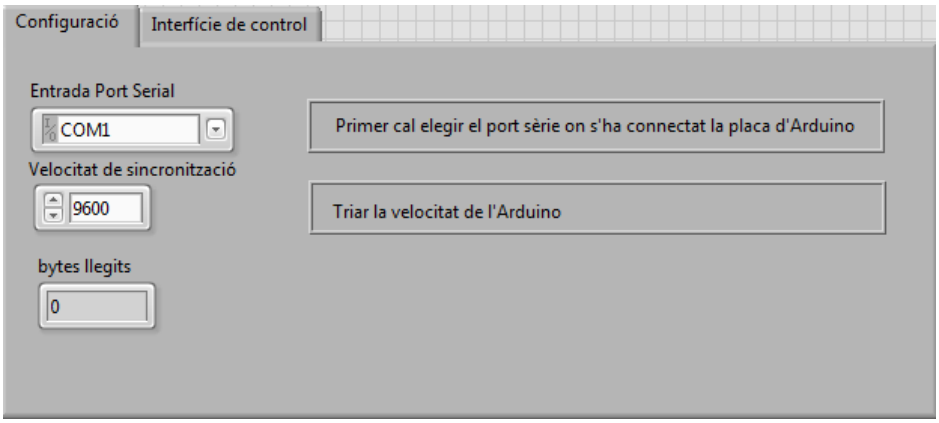


Figura 12. Pàgina de configuració

A la figura 12 es pot veure el menú de configuració de la placa d'Arduino, en la qual s'ha d'escollir el número de port sèrie que s'ha connectat a l'ordinador. Tot seguit s'ha de triar la velocitat de sincronització per enviar i rebre les dades, que en el cas de l'Arduino UNO és de 9600.

Figura 13. Interfície de control

A la figura anterior es pot observar la interfície de l'usuari on es mostren els valors llegits per l'Arduino. També es disposa de dos botons un per parar tot el procés i l'altra per guardar les dades llegides. Per poder-les guardar correctament s'ha de prémer el botó de Grabar i apareixerà una pantalla per triar el lloc on es vol guardar, a partir d'aquest moment el programa comença a emmagatzemar les dades fins que es torna a polsar el botó.

3.2 Identificació del model

Per tal d'obtenir un model pels diferents sistemes dels quals es disposen s'ha utilitzar el programari de Matlab juntament amb alguns dels seus toolkits.

A continuació es mostraran les tres identificacions obtingudes, la primera de les quals serà explicada detingudament, en canvi, les altres només es mostraran els resultats i es farà un breu comentari per tal de no repetir l'explicació del procés.

3.2.1 Identificació de la direcció dreta

El primer que cal fer és carregar la taula de dades del sistema obtingudes amb l'adquisició explicada anteriorment i assignar a cada columna la descripció corresponent per poder-les processar.

Un cop carregades les dades, amb la funció "ident" s'obre la finestra on s'hi introduiran les especificacions corresponents per tal d'obtenir una identificació correcta. A la part esquerra s'hi ha d'importar la taula de dades carregades amb el Matlab de la següent manera:

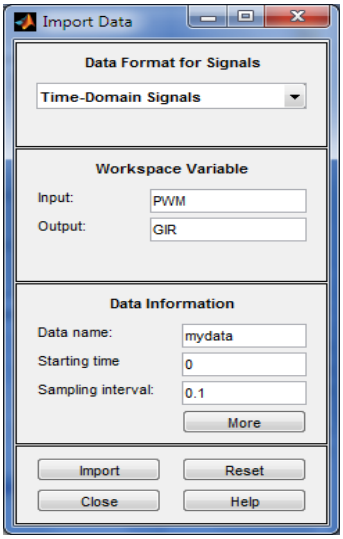


Figura 14. Importació de les dades

Com es pot observar a la figura anterior s'han d'introduir els noms assignats en el Matlab per l'entrada i la sortida. També és important introduir correctament el temps de mostreig que en aquest cas és de 0.1 segons. Una vegada importats els valors, tal i com es pot veure a la següent figura, es poden veure les dues gràfiques creades. En la primera de les quals es mostra la sortida del sistema i la consigna d'entrada a l'altra.

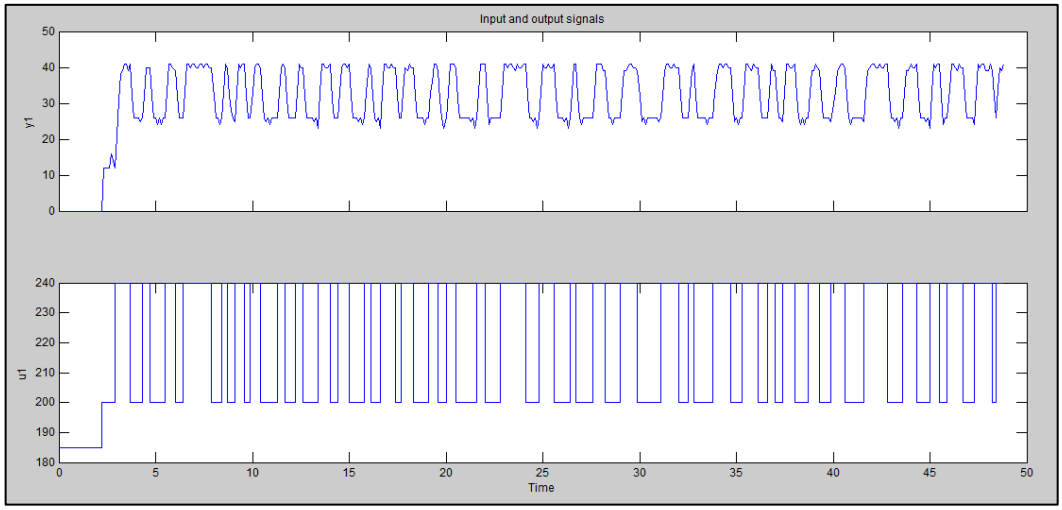


Figura 15. Adquisició de les dades del gir dret

A la part inferior de la figura 15 es pot veure com s'hi entren dos valors de consigna consecutivament en instants de temps diferent i a la part superior la resposta del sistema físic. Es pot observar que durant el primer canvi de 0 a 27º no hi aconsegueix arribar sinó que es queda a 12º però a partir d'aquí la resposta es correspon totalment als valors desitjats

A continuació s'ha de realitzar l'estimació del model paramètric lineal on s'haurà de triar l'ordre de la funció que vulguem obtenir. Per triar el millor hi ha una opció que es diu selecció d'ordre la qual indicarà les millors estimacions de la següent manera:

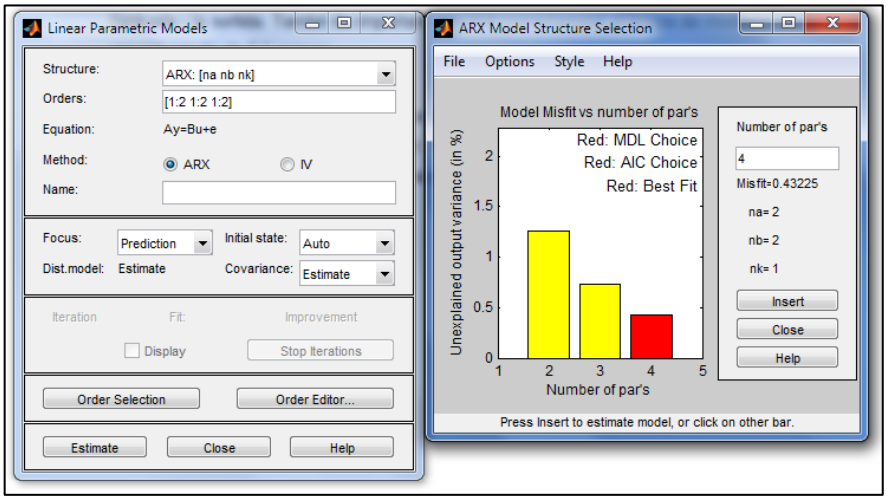


Figura 16. Estimació paramètrica lineal

A l'esquerra de la figura anterior es pot observar la finestra per estimar el model paramètric lineal i a la part dreta les tres opcions més bones. La columna vermella és una equació de segon ordre i es la que s'ha considerat la millor de les que s'han obtingut, per tant la inserim i mirem com és la seva resposta.

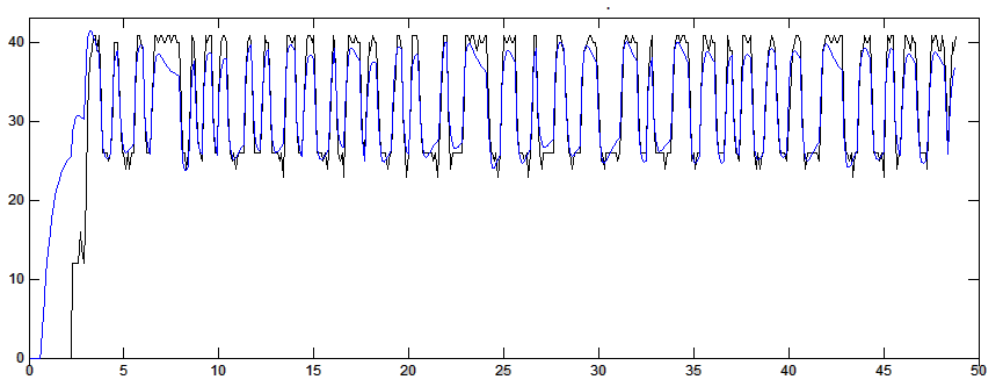


Figura 17. Representació del model

La figura 10 mostra la representació del model que s'ha obtingut fent l'estimació anterior. La línia de color negre és la resposta física del sistema i la de color blau és l'estimació calculada pel model. Podem observar que pràcticament són iguals, nomes difereixen al principi on la resposta del sistema matemàtic és més ràpida que la real però tot i això es pot considerar com una bona identificació.

A continuació si la identificació és correcta s'ha de transferir a l'espai de treball del Matlab per tal de poder treballar amb ella. Per obtenir la funció de transferència en temps continu primer s'ha de passar en temps discret. La funció obtinguda en temps continu és la següent:

$$\frac{2.72s + 1.797}{s^2 + 8.363s + 12.28} \qquad \text{(Eq. 4)}$$

Per comprovar si la funció descriu el sistema físic la podem representar amb el simulink però primer observarem si el guany de la funció quan la s tendeix a 0 coincideix amb el del sistema físic. Si dividim 1.797 entre 12.28 dóna un valor de 0.146 i el quocient entre el valor de la resposta de 40º amb el PWM de 240 obtenim 0.16, per tant el guany és molt similar. Per aquesta funció de transferència s'han obtingut dos pols, l'un a -6.462 i l'altre a -1.90. Com que tots dos estan situats al semiplà esquerra la funció de transferència és estable

Com ja s'ha dit anteriorment el següent pas és dur a terme la representació amb el simulink per tant introduïm els següents blocs per tal d'obtenir la resposta del sistema matemàtic:

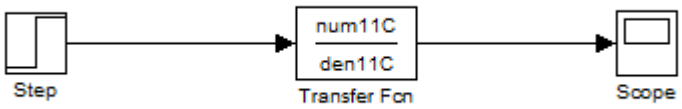


Figura 18. Blocs del Simulink

El conjunt de blocs consta d'una entrada graó el valor del qual es pot modificar, el bloc de "transfer function" on s'hi ha d'introduir el numerador i el denominador de la funció de transferència en temps continu i finalment hi ha l'oscil·loscopi on es podrà veure la representació corresponent

La resposta obtinguda per un graó de 240 (que és un dels valors de PWM utilitzats) és el que es pot veure a continuació:

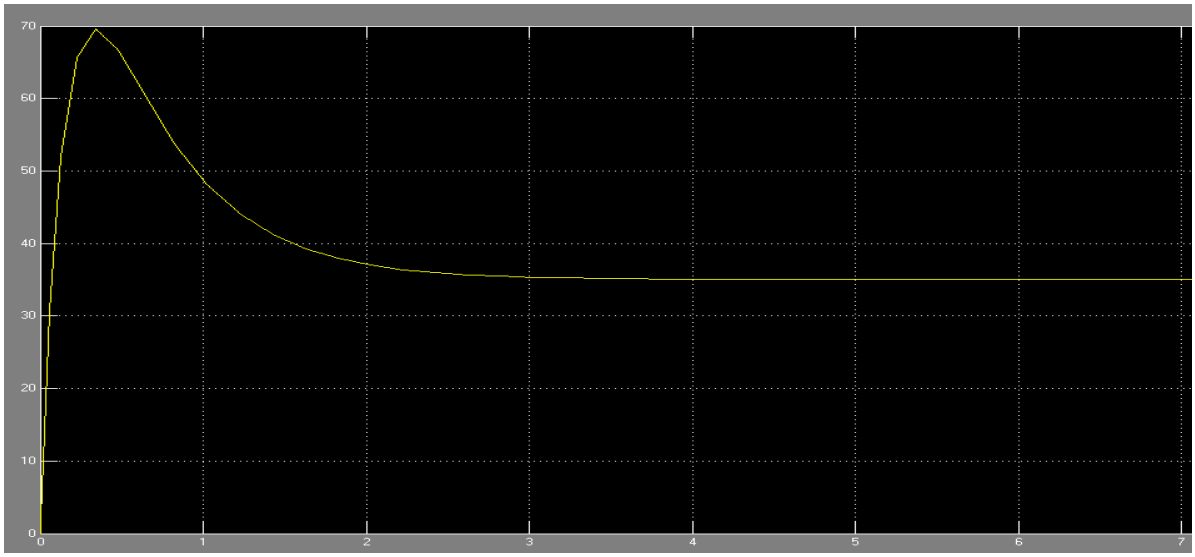


Figura 19. Sortida graó de 240

A la figura anterior es mostra la representació de la funció de transferència obtinguda pel model paramètric lineal escollit. Segons el sistema físic, per un valor d'entrada de 240 s'hauria d'obtenir un angle aproximat de 40º, en canvi segons el model matemàtic veiem que obtenim un valor de 35º. Això és degut a que el guany del sistema físic era de 0.16 i en canvi el del model matemàtic de 0.146, si fem la multiplicació corresponent podem observar que la variació prové d'aquí. Per tant, a l'hora de fer el controlador intentarem corregir aquest error.

3.2.2 Identificació de la direcció esquerra

Tal i com ja s'ha dit anteriorment en aquest cas no es tornarà a explicar el procediment, sinó que es mostraran i s'explicaran els resultats obtinguts.

Per poder diferenciar els graus de gir de la direcció esquerra amb la dreta es consideraran com a nombres negatius. En aquest cas els valors de PWM varien entre 100 i 150, donant una resposta aproximada de -37 i -21 respectivament. El problema és que com més gran és el valor de PWM més petit és l'angle de gir i per fer la identificació no funciona ja que per un augment de l'entrada la sortida també hauria d'augmentar. Un altre problema és que el valor 0 de PWM en la sortida no dóna 0 graus, sinó que per tenir-los s'hi ha d'aplicar un valor aproximat de PWM de 185.

Per tal de corregir aquest problema s'han escalats els valors de PWM col·locant el 0 de PWM com a 0º i d'aquesta manera s'han obtingut els següents resultats:

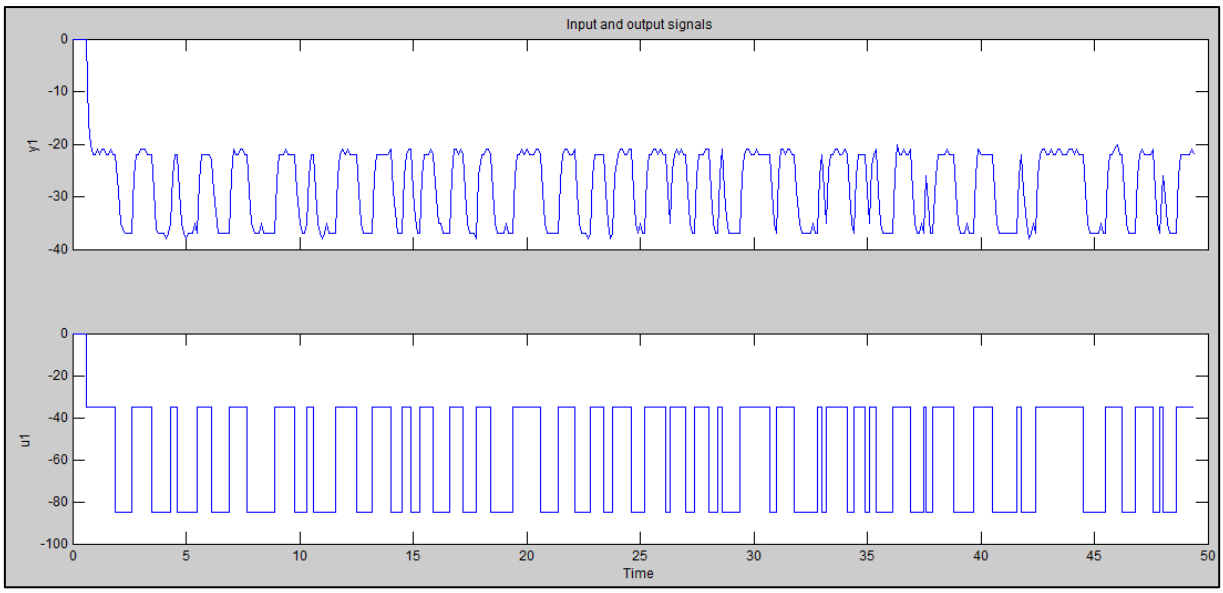


Figura 20. Adquisició de les dades del gir esquerra

Observant la figura 20 podem veure com amb l'escalat que s'ha fet els valors de 100 i 150 de PWM han passat a ser -86 i -36 respectivament. D'aquesta manera amb valor absolut per un augment de PWM també augmenta el gir.

L'equació de transferència resultant de la identificació del model és la següent:

$$\frac{2.91s + 11.76}{s^2+12.75s+25.11} \qquad \text{(Eq. 5)}$$

Els pols corresponent són -10.31 i -2.43, com que tots dos estan situats al semiplà esquerra es pot dir que és un sistema estable. Fent ús de l'equació de transferència s'ha pogut fer una simulació del model identificat.

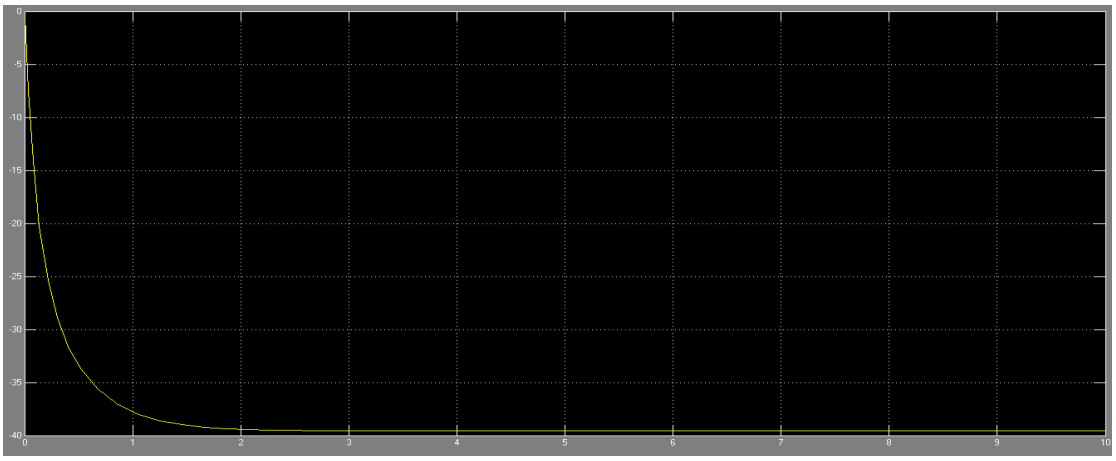


Figura 21. Sortida graó de 100

Com es pot observar per una entrada de PWM de 100 s'hauria d'estabilitzar a -37 graus i podem veure com al cap de dos segons s'estabilitza a -39. Com que l'error és molt petit de es pot considerar un bon model.

3.2.3 Identificació del motor endavant

Pel cas del motor a partir d'un valor de PWM de 90 el motor comença a anar a endavant, de 90 a 60 el motor està parat i per valors més petits va cap endarrere. La resposta del motor endarrere no és lineal, sinó que és tot o res per tant no cal plantejar fer una identificació ja que no es podrà fer cap controlador.

Tot seguit es mostraran les dades adquirides del motor i es comentarà la resposta obtinguda.

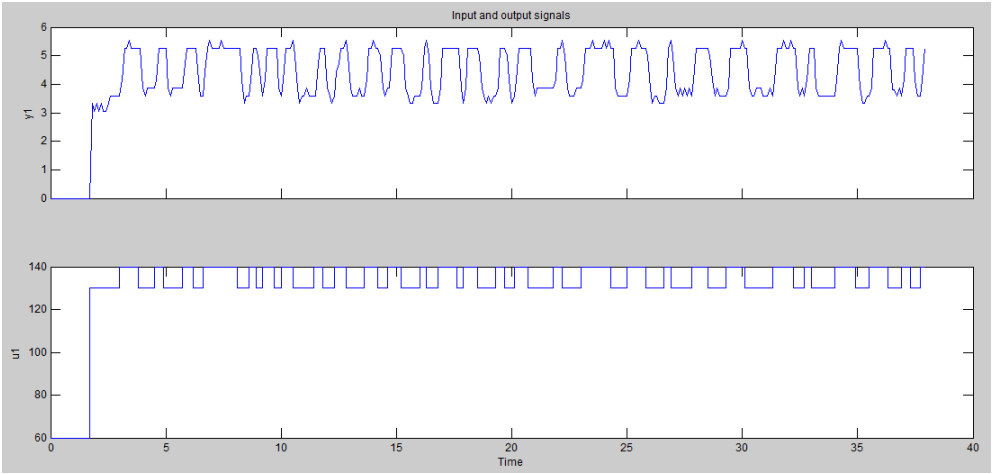


Figura 22. Adquisició de les dades del motor

Com es pot observar a la figura 22, l'entrada de PWM oscil·la entre valors de 140 i 130 donant una resposta de 5.2 i 3.8 cm/s. La velocitat l'obtenim amb cm/s per tal d'obtenir més precisió enlloc de fer-ho amb m/s.

L'equació de transferència resultant de la identificació del model és la següent:

$$\exp(-0.1*s) * \frac{1.17s + 1.68}{s^2+19.44s+51.05} \qquad \text{(Eq. 6)}$$

Els pols corresponents són -16.31 i -3.12 per tant abans de simular el model podem deduir que el sistema serà estable perquè els pols estan situats al semiplà esquerra. Fent us de l simulink hem pogut obtenir la següent simulació de la funció de transferència.

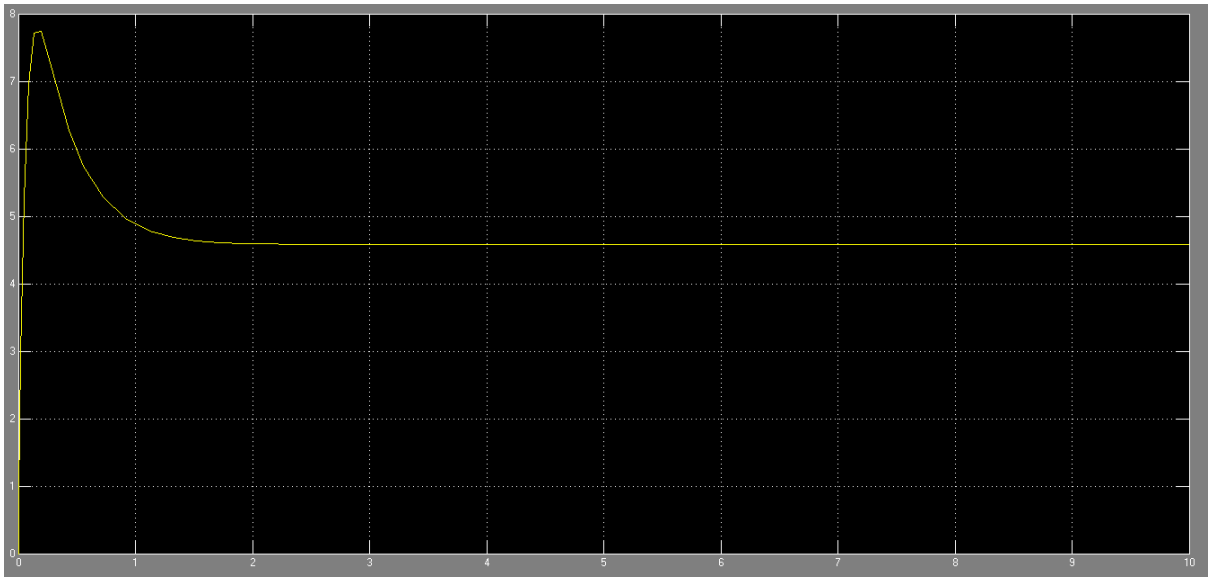


Figura 23. Sortida graó de 140

En aquesta figura podem veure que per una entrada de 140 obtenim un sobrepic de 7.8 en el model obtingut però això no és aplicable al model real ja que el valor màxim de la velocitat no arriba a 6 cm/s per tant quedarà saturat. Tot i això a través d'un controlador s'intentarà corregir aquest sobrepic tant gran perquè no succeeixi per valors més petits de PWM. A més, la resposta s'hauria d'estabilitzar a uns 5.2 cm/s i en canvi ho fa a 4.6 cm/s.

3.3 Controlador de la direcció

Abans de fer el controlador s'ha de mirar si la funció de transferència té algun pol dominant per tal de reduir-la a una de primer ordre. En el cas de la direcció esquerra en tenim un a de 2,43. Per tant la funció seguirà la següent equació:

$$\frac{K'}{s+2.43} \qquad \text{(Eq. 7)}$$

El valor de K' és desconegut però si es divideix el numerador i el denominador entre 2.43 (per deixar el terme independent amb un valor de 1) llavors la K que s'obté és el guany estàtic de la funció de transferència inicial. El guany es troba fent el límit de s cap a 0, és a dir substituint els valors de s per 0. D'aquesta maner obtenim la següent funció:

$$\frac{K'}{s+2.43} = \frac{K}{0.41s+1} = \frac{0.46}{0.41s+1} \qquad \text{(Eq. 8)}$$

Tot seguit cal aplicar el controlador PI a la funció de transferència reduïda i tancar el llaç, per tant obtindrem una funció de segon ordre de les quals en podrem calcular el Kp i Ki, tal i com es pot observar a continuació:

$$G(s)=\frac{1.12K_P s+0.46K_I}{s^2+(2.17+1.12K_P)s+1.12K_I} \qquad \text{(Eq. 9)}$$

Com que és una funció de segon ordre podem deduir les següents expressions:

$$2.17+1.12K_P=2\,\xi\,w_N \qquad \text{(Eq. 10)}$$

$$1.12K_I=w_N{}^2 \qquad \text{(Eq. 11)}$$

Per obtenir un bon esmorteïment definim un valor de $\xi=0.7$ i a continuació juguem amb el temps d'establiment del 5 %. Després de provar diferents valors, s'ha arribat a la conclusió que el més òptim és de 1.7 segons. Una vegada decidit el temps d'establiment podem trobar els valors de K_P i K_I de la següent manera:

$$Ts\ 5\% = \frac{3}{\xi * w_N} \qquad \text{(Eq. 12)}$$

Amb l'equació anterior s'ha pogut trobar el valor de w_N i substituint a les equacions 10 i 11 s'ha trobat kp=1.35 i Ki=5.67. Per tant el controlador en temps continu a aplicar és el següent:

$$C(s)=\frac{1.35s+5.67}{s} \qquad \text{(Eq. 13)}$$

A continuació s'ha de passar el controlador a temps discret per poder obtenir les equacions en diferències, amb la qual s'ha fet sevir una freqüència de mostreig de 0.1segons:

$$C(z)=\frac{1.35-0.78}{z-1} \qquad \text{(Eq. 14)}$$

Com podem observar a la següent figura, si tenim el següent llaç de control tancat, fàcilment podem obtenir el valor de PWM que s'ha d'aplicar al sistema:

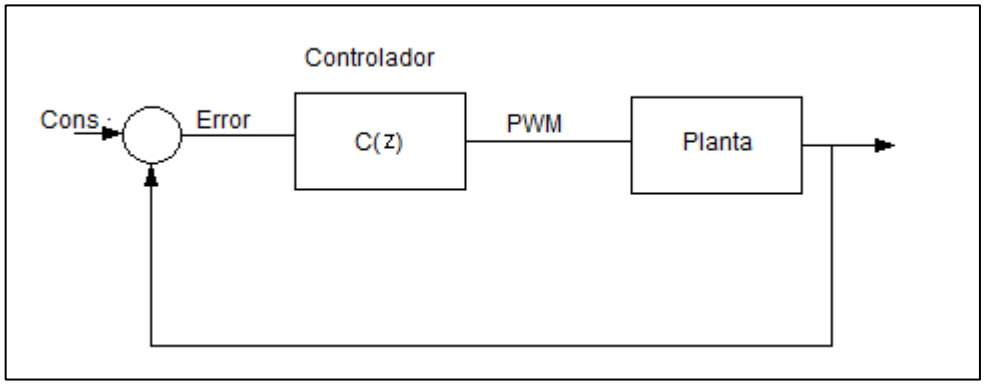


Figura 24. Control llaç tancat

Com podem observar el PWM(z) és el producte entre C(z) i E(z) i d'aquí en podem extreure l'equació en diferències:

$$PWM(t) = 1.35\,E(t) - 0.78\,E(t-1) + PWM(t-1) \qquad \text{(Eq. 15)}$$

Un cop trobada l'equació ja es pot aplicar directament a la programació de l'Arduino, on s'ha comprovat que la mateixa equació serveix per regular la direcció en sentit dret i esquerra. Per tant es pot dir que s'ha obtingut un resultat molt òptim ja que facilita el controlador.

3.4 Controlador del motor

Per fer el control del motor seguint el mateix procediment que en l'apartat anterior vaig tenir dificultosos problemes per obtenir una equació correcta. Per tant vaig intentar trobar el controlador a través del Sisotool amb l'ús del mètode del lloc de les arrels. Malgrat tots els intents, el més favorable aconseguia estabilitzar-se al cap de 1 segon amb un sobrepic de 11.7%. Tot i que el sobrepic sigui superior al 10% es poden considerar uns valors teòrics correcte. El problema va sorgir quan vaig implementar el controlador, ja que no funcionava en absolut. Després de provar altres controladors vaig arribar a la conclusió que potser estava aplicant una teoria de control en temps continu a un sistema que realment no era continu.

Per resoldre aquest dubte vaig mirar el voltatge d'entrada del motor fent augmentar la velocitat progressivament des de 0 fins al seu valor màxim. I efectivament, vaig poder observar que la velocitat no era lineal, sinó que seguia aproximadament la següent gràfica, en relació al voltatge i l'angle del servomotor:

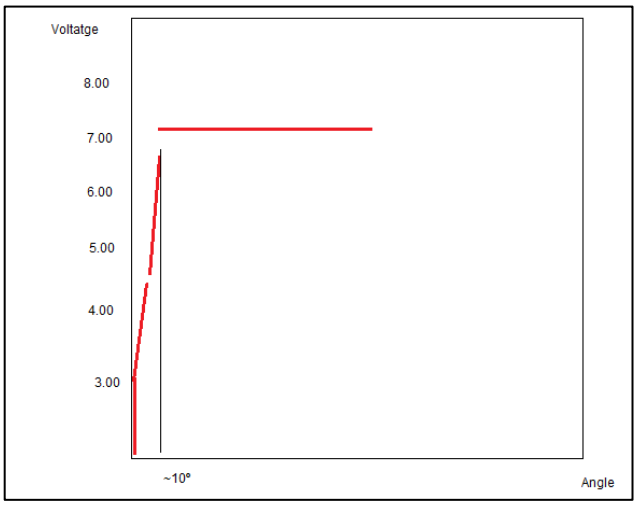


Figura 25. Gràfic voltatge-angle servomotor

Com podem veure per un valor de l'angle molt petit (el PWM és el que fa variar l'angle) el canvi de voltatge al qual alimentem el motor és molt gran, en canvi a partir dels 10º del

servomotor la velocitat s'estableix a un valor constant. Per tant, es fa molt complicat treballar en els trams en què la velocitat varia gradualment ja que per un valor de PWM molt petit de seguida salta al valor màxim. A més, cal afegir que hi ha tres trams discontinus. És per aquests dos motius que no s'ha pogut trobar un controlador pel motor.

4. CONTROL DE TRAJECTÒRIES

Primerament, abans de poder programar les corresponents maniobres en el cotxe primer hem de conèixer la seva posició en cada moment. Per tal de fer-ho possible s'utilitzarà la odometria.

4.1 Odometria

La odometria és l'estudi de l'estimació de la posició relativa del vehicle durant la navegació a partir de la localització inicial. Per realitzar aquesta estimació s'utilitza informació sobre la rotació de les rodes de la direcció i la velocitat al llarg del temps. Això ens proporciona una bona precisió en trams curts on l'acumulació d'errors és petita.

Aquesta es basa en equacions simples fàcils d'implementar i utilitza les dades de l'encoder situat a la sortida del motor i del potenciòmetre lineal de la direcció. Tot i això es podrien produir errors en cas que les rodes patinessin ja que les dades de l'encoder registraria revolucions a les rodes tot i que no es corresponguin a un moviment lineal.

En aquest cas s'estudiarà la odometria en el pla, és a dir el vector (x, y, θ). Abans de realitzar les càlculs és necessari saber les revolucions que s'han obtingut de l'encoder i convertir els pulsos en velocitat lineal tal i com s'ha explicat anteriorment. Pel cas d'un sistema no holonomic tindríem la configuració de coordenades que es pot veure a la figura 26:

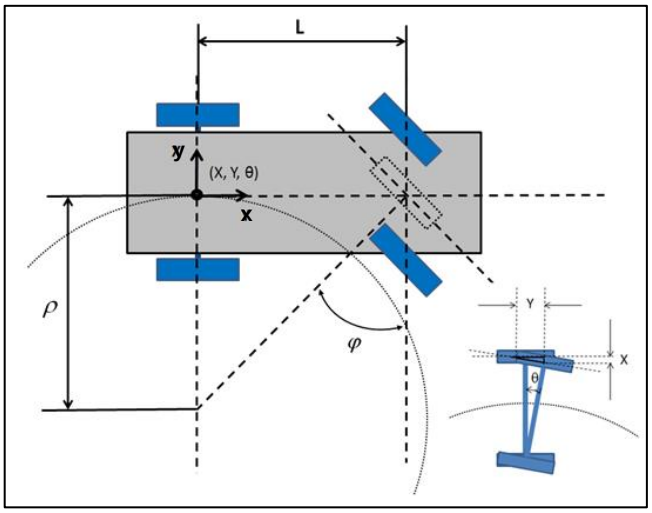


Figura 26. Coordenades sistema no holonomic

Partint de la figura anterior podem observar que θ és la variació de la orientació de l'eix de la direcció respecte una posició inicial. Aquest valor s'obté de la següent equació:

$$\theta = \frac{V}{L} \tan\varphi \qquad \text{(Eq. 16)}$$

On φ és l'angle de la roda respecte l'eix Y. Per tant partint de l'origen (0, 0, 0) obtindrem les següents equacions depenent de la velocitat (V) i la orientació (θ):

$$\theta_i = \theta_{i-1} + \Delta\theta_i \qquad \text{(Eq. 17)}$$

$$x_i = x_{i-1} + \Delta V_i \cos(\theta_i) \qquad \text{(Eq. 18)}$$

$$y_i = y_{i-1} + \Delta V_i \sin(\theta_i) \qquad \text{(Eq. 19)}$$

On x_i, y_i é la posició relativa del punt central del robot en un instant de temps i.

4.2 Maniobres d'aparcament

Primerament abans d'entrar en la programació de les diverses maniobres d'aparcament s'ha de fer un petit anàlisi per observar com es comporten els sistemes d'aparcament en cotxes reals.

Hi ha diferents tipus d'aparcament, hi ha el lateral, en bateria marxa endarrere, en bateria marxa endavant i en diagonal. Dins dels sistemes d'aparcament també n'hi ha de dos tipus que es diferencien en l'autonomia, és a dir un aparca amb el pilot a dins i l'altre és completament autònom.

Per delimitar una mica el projecte he optat per escollir dos d'aquests aparcaments: el lateral i el de bateria marxa endarrere.

4.2.1 Aparcament en bateria

Fixant-nos en com aparquen els sistemes incorporats s'ha vist que amb una sola maniobra aconsegueixen aparcar correctament. Evidentment en aquest projecte no serà possible aconseguir fer-ho d'aquesta manera degut a què el gir de la direcció del robot no ho permet ja que l'angle màxim de gir és de 30º.

Abans de programar l'Arduino s'han de fer diverses proves amb el cotxe en mode manual i adquirint les dades d'odometria. D'aquesta manera es poden captar les coordenades de les posicions inicials i finals de cada maniobra.

Tot seguit es comentarà una de les proves inicials que es van realitzar per fer l'aparcament però que no va ser satisfactòria i es va haver de canviar de sistema d'aparcament. A continuació es pot veure la situació d'aparcament:

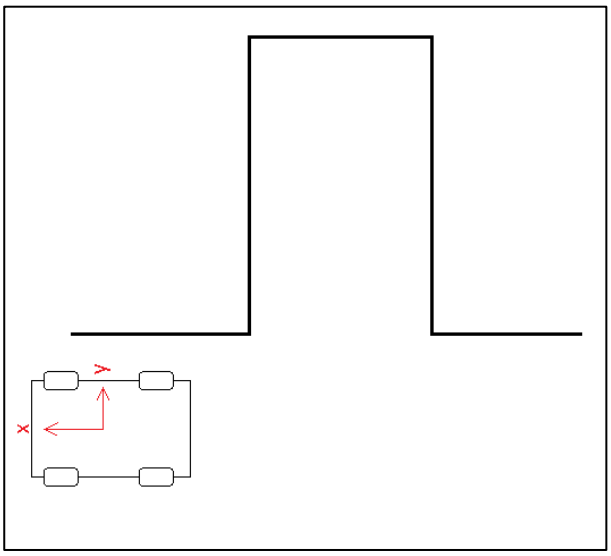


Figura 27. Situació aparcament en bateria 1

Com es pot veure a la figura 27 el cotxe està inicialment per davant de l'aparcament. Les coordenades utilitzades son les que es poden observar en vermell per la x (positiu cap a l'esquerra) i la y (positiu cap a dalt).

Al principi es va intentar aparcar el cotxe de manera que tan bon punt s'ordenés la maniobra d'aparcament el cotxe ja activés la marxa endarrere per aparcar, que és el mode en que ho fan els sistemes d'aparcament reals.

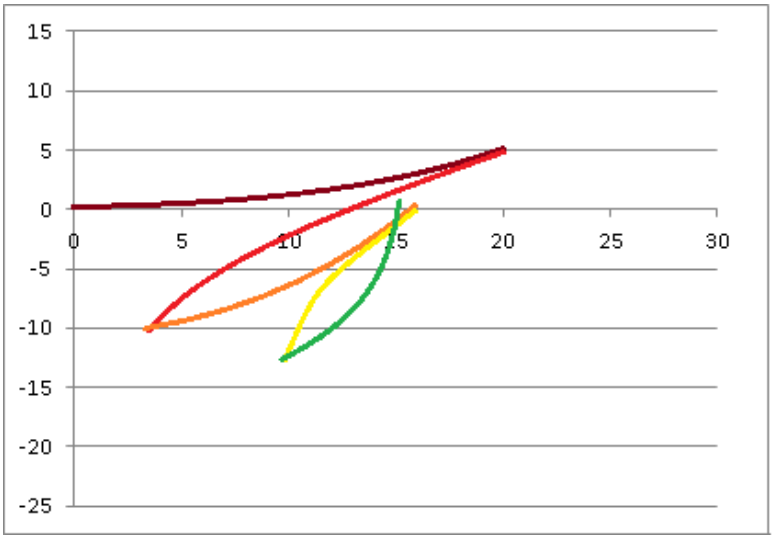


Figura 28. Aparcament inicial en bateria

Aquest va ser la primera maniobra que es va provar però malauradament després d'analitzar-lo es va haver de descartar degut a dos problemes. La primera maniobra és la de color marró amb la qual el cotxe va cap endarrere i aquí és on observem el primer problema ja que necessita molt d'espai per aconseguir el gir la qual cosa és poc pràctica i realista. Seguidament, de color vermell es pot veure com inicia el moviment cap endavant fins a la coordenada (4, -10). Llavors va cap endarrere (taronja) un altre cop cap endavant (groc) i un últim cop cap endarrere. Aquí encara faltaria una última maniobra en què el cotxe aniria cap endarrere de manera recte per entrar el cotxe al fons de l'aparcament. Com es pot observar, els segon problema que es va detectar va ser que feia moltes maniobres per aconseguir aparcar correctament.

Degut a aquests inconvenients es va optar per no fer la primera maniobra marxa endarrere i començar a fer-la cap endavant per tal de disminuir el nombre de maniobres i l'espai necessari ja que en aquest cas el cotxe estarà posicionat de la manera que s'indica a la figura 29.

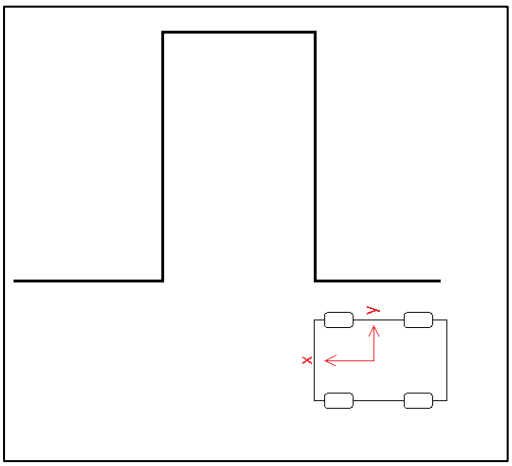


Figura 29. Situació d'aparcament en bateria 2

En aquest cas es pot observar que el cotxe està situat just abans de l'aparcament, d'aquesta manera, com que avancem cap endavant aconseguim reduir la distància necessària.

A continuació es descriurà com s'ha realitzat cadascuna de les maniobres realitzades indicant l'angle de gir, el sentit del motor i les coordenades corresponents.

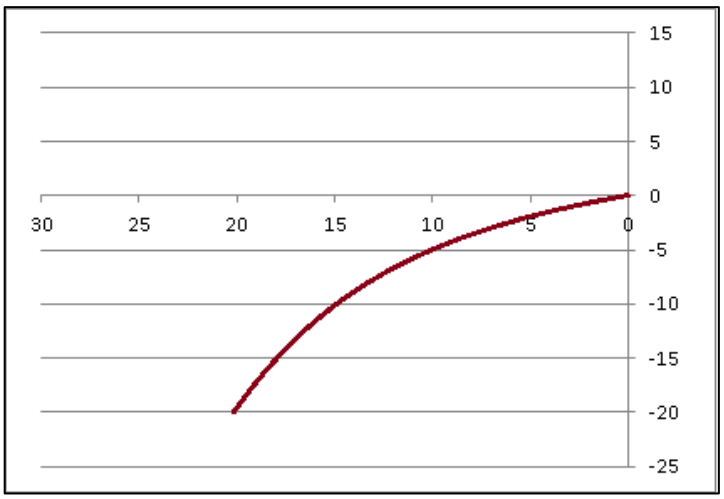


Figura 30. Maniobra 1

A la figura 30 es pot observar la primera maniobra on inicialment el cotxe està a la posició (0, 0) i avança amb la direcció a -30º fins a la coordenada (20, -20). D'aquesta manera el cotxe aconsegueix una inclinació que ens farà estalviar maniobres a l'hora d'aparcar.

La següent imatge mostra la segona maniobra realitzada:

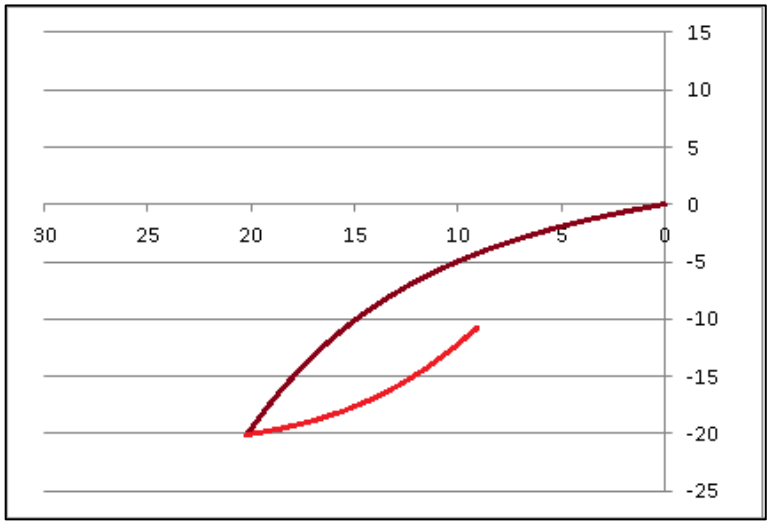


Figura 31. Maniobra 2

A la figura 31 es pot observar de color vermell la segona maniobra la qual consisteix en activar el motor cap endarrere amb un angle de gir de la direcció de 30º fins a arribar a la posició (10, -10).

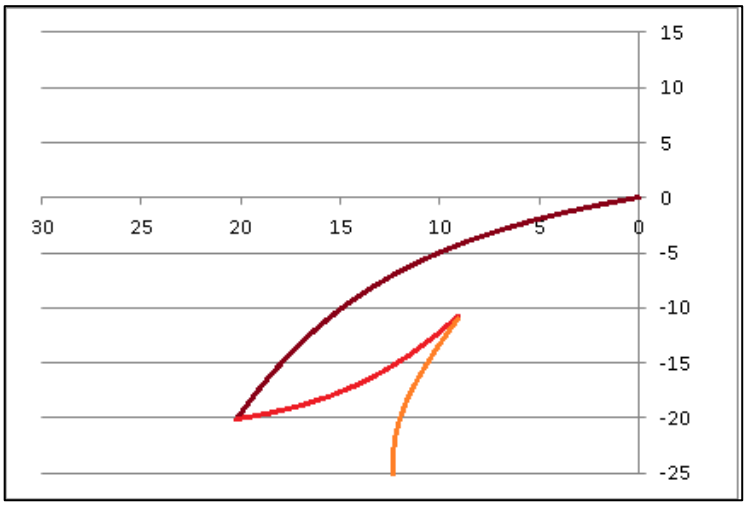


Figura 32. Maniobra 3

A la figura anterior, de color taronja, es pot veure la tercera maniobra en la qual el robot avança amb un angle de -30º fins a les coordenades (12, -25).

Finalment es mostra l'última maniobra realitzada a la següent figura:

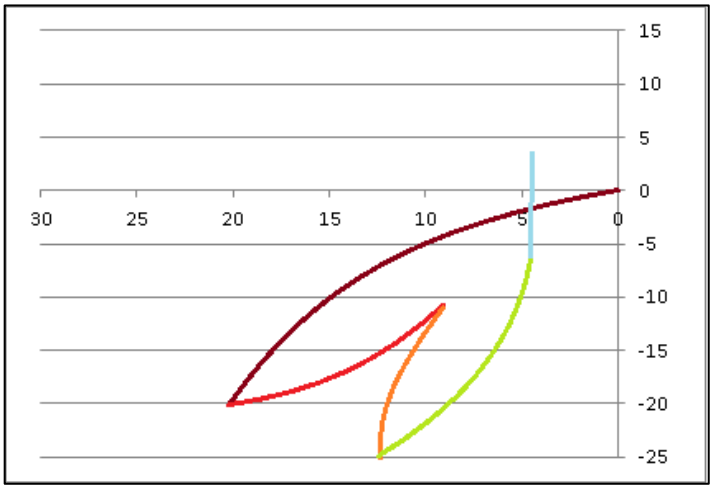


Figura 33. Maniobra 4

A la figura 33 es mostra l'última maniobra, la qual l'he separat en dos colors ja que canvia l'angle de gir. Primerament, de color verd, el cotxe va cap endarrere amb un angle de 30º fins a la posició (5, -5) i llavors, amb el color blau es representa que el robot segueix retrocedint en línia recte fins arribar al fons de l'aparcament.

Per tan, amb aquest segon sistema d'aparcament s'ha aconseguit reduir substancialment el nombre de maniobres i el sistema d'aparcament es bastant més realista tot reduint l'espai d'aparcament.

4.2.2 Aparcament lateral

Per realitzar l'aparcament s'ha seguit exactament el mateix procediment que amb l'anterior, és a dir primer s'han fet les proves amb el cotxe en mode manual i adquirint les dades de posició per obtenir les coordenades necessàries. A diferència que en l'aparcament en bateria no s'explicaran els assajos prèvis degut a que el sistem pel qual es va optar inicialment ja va ser el correcte exceptuant que inicialment també es va haver de fer avançar el cotxe cap endavant a 30º per agafar certa inclinació d'aparcament.

Un cop conegudes les coordenades actuals del posicions del vehicle es poden utilitzar per posicionar-lo en cada cas. Per dur a terme l'aparcament s'ha de partir d'unes condicions inicials de posicionament ja que no es disposa de sensors de proximitat. Per aparcar la part

posterior del cotxe ha d'estar situada a la part frontal de l'aparcament i situat al lateral esquerra aproximadament a uns 3cm de distància.

El cotxe s'ha programat per fer un aparcament lateral a la dreta ja que és el cas més comú en la vida real degut a què poques vegades s'aparca a l'esquerra perquè s'invadeix el carril contrari. Tal i com es pot veure a la figura 27, el sentit de les coordenades son les següents: la x és positiva pel desplaçament lineal del cotxe cap endavant i la y és positiva pel moviment del cotxe cap a la dreta.

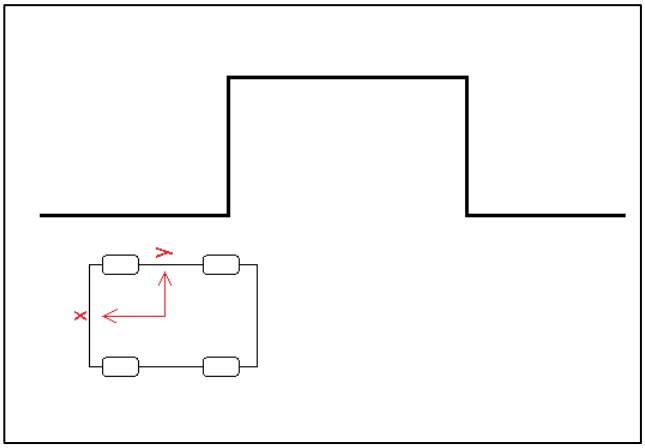


Figura 34. Situació d'aparcament lateral

A continuació es descriuran les maniobres que s'han seguit juntament amb els gràfics corresponents a les coordenades adquirides durant un aparcament.

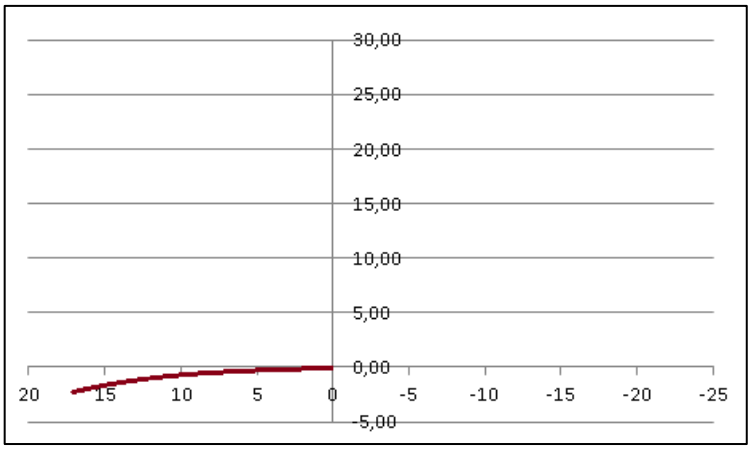


Figura 35. Maniobra 1

A la figura anterior es pot observar que en l'instant de temps 0 el cotxe es troba a les coordenades (0, 0) llavors es fa avançar el cotxe amb un angle de -30º fins a la posició

aproximada de (17, -2.5) per tal d'agafar la inclinació inicial correcte per poder entrar a l'espai d'aparcament. Si no es fes aquest moviment amb un angle de -30º, que és el màxim que permet el cotxe, i es volgués començar amb una marxa endarrere per entrar cap a l'aparcament s'haurien de fer més maniobres ja que els 30 graus són massa poc.

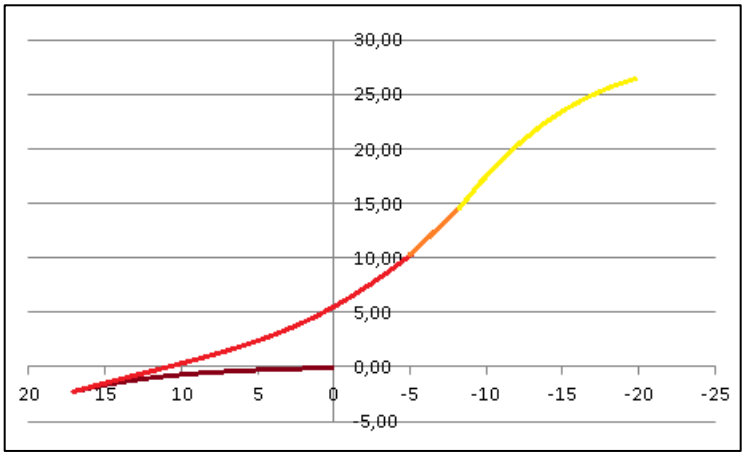


Figura 36. Maniobra 2

En aquesta figura 36 es mostra la maniobra 2, la qual l'he separat en tres colors ja que varia l'angle de la direcció i el cotxe retrocedint. Primer es fa amb un angle de 30º fins a les coordenades (-5, 10), tot seguit es canvia la direcció a 0º fins les coordenades (-7, 15) i finalment es posa a -30 fins a (-20, 26). D'aquesta manera ja tenim el cotxe a dins l'aparcament però amb una inclinació incorrecta, per tant tot seguit es rectificarà anant endavant.

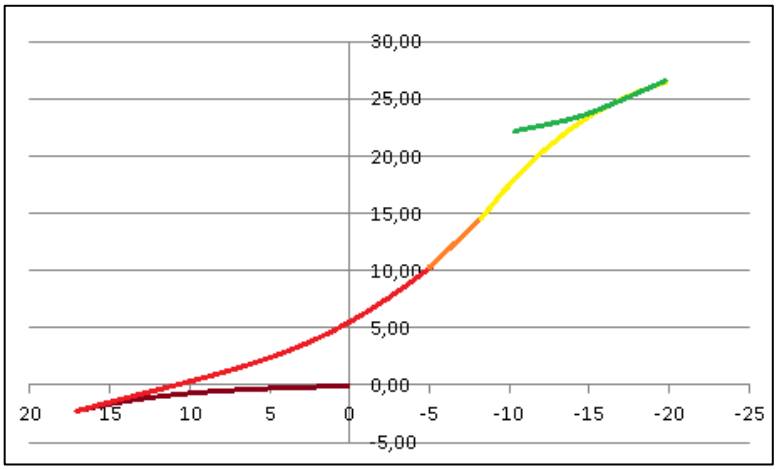


Figura 37. Maniobra 3

Tal i com ja s'ha dit la següent maniobra és cap a endavant amb un angle de 30º fins a arribar al límit de l'aparcament per no tocar amb la paret fins a la coordenada (-10, 22).

Com que encara no s'ha aconseguit tenir el cotxe amb una inclinació de 0º s'ha de tornar a repetir la maniobra 4 amb el qual s'ha obtingut el següent resultat:

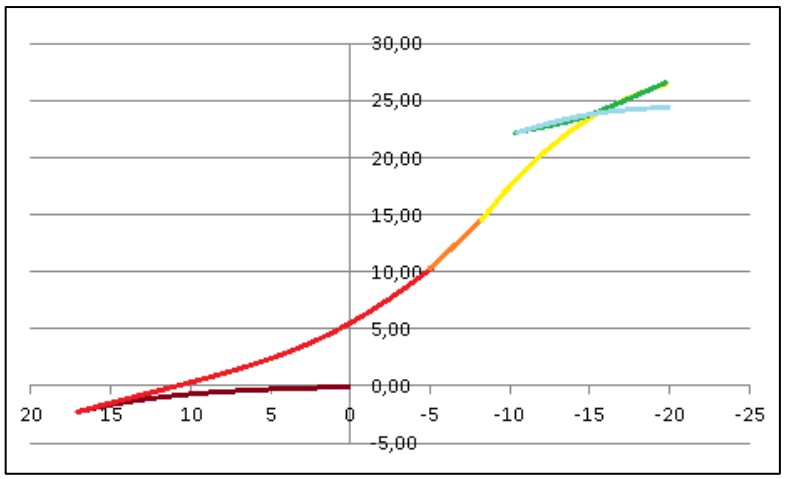


Figura 38. Maniobra 4

A la figura anterior es pot observar que el cotxe està anant endarrere i amb un angle a la direcció de -30º, el qual .va de la posició (-10, 22) fins a la (-20, 24.5) tot respectant el límit on es troba la paret.

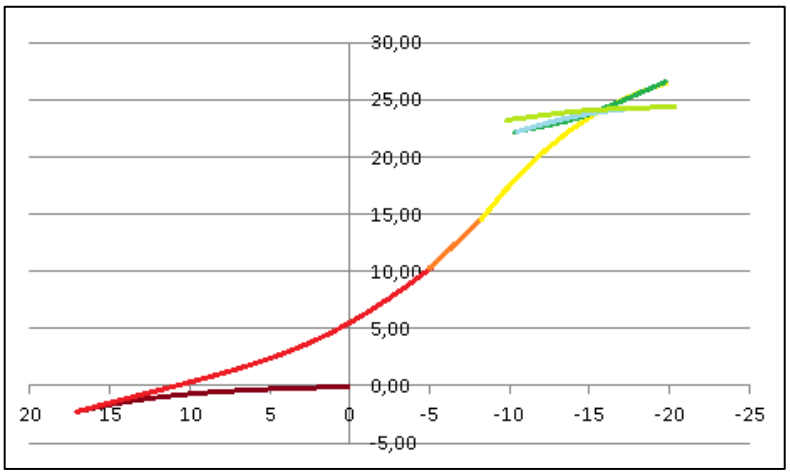


Figura 39. Maniobra 5

Ja per acabar i aconseguir posar el cotxe en paral·lel es fa anar cap endavant amb un angle de 30º aconseguint aparcar el cotxe a la posició (-10, 24). Tal i com es pot observar a la figura 39 podem veure que per aparcar el cotxe s'ha necessitat un marge d'aparcament de 10 cm (de -20 a -10), això em fa dir que els resultat obtinguts han sigut molt bons i s'ha realitzat un bon aparcament.

Resumint, s'han utilitzat 5 maniobres, la primera es fa avançar el cotxe amb un angle concret de gir per tal d'encarar-lo i maniobrar millor. La maniobra 2, que esta dividida en tres i van lligades, s'ha utilitzat per fer entrar el cotxe a l'espai d'aparcament a marxa endarrere. Un cop el cotxe ja és a dins es fan dues maniobres més per deixar el cotxe en paral·lel respecte la vorera. La primer es fa cap endavant i deixar el cotxe recte i la segona cap endarrere per centrar-lo.

5. COMUNICACIÓ ROBOT-USUARI

A la comunicació entre el robot i l'usuari hi intervenen bàsicament tres dispositius: l'Arduino, el mòdul de comunicació i un smartphone.

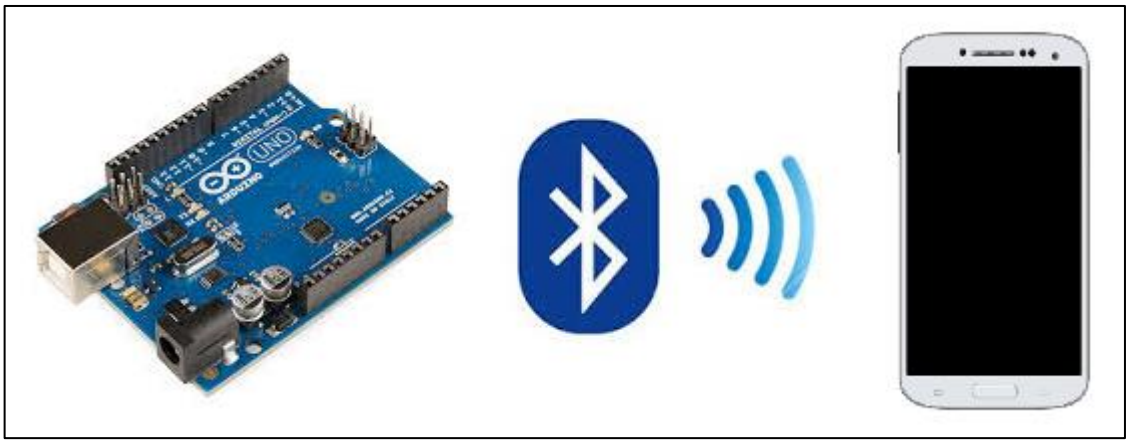


Figura 40. Trajectòria d'aparcament

La placa Arduino s'encarrega de rebre les dades corresponents enviades a través de del telèfon mòbil i gestionar-les segons el programa ques'hi ha carregat.

La interfície utilitzada per tal d'establir una comunicació amb el robot és a través d'una aplicació Android de telèfon mòbil. Per tal de realitzar-la s'ha utilitzat el framework Appinventor, un framework és un espai de treball en el qual es pot realitzar un esquema pel desenvolupament i implementació d'una aplicació.

Appinventor és un entorn integrat de desenvolupament de Google que permet crear aplicacions mòbils senzilles que es poden utilitzar amb telèfons mòbils amb sistema operatiu Android i com a mínim 250 MB de memòria RAM. Ens proporciona una eina online fàcil d'utilitzar i molt intuïtiva amb uns coneixements bàsics de programació.

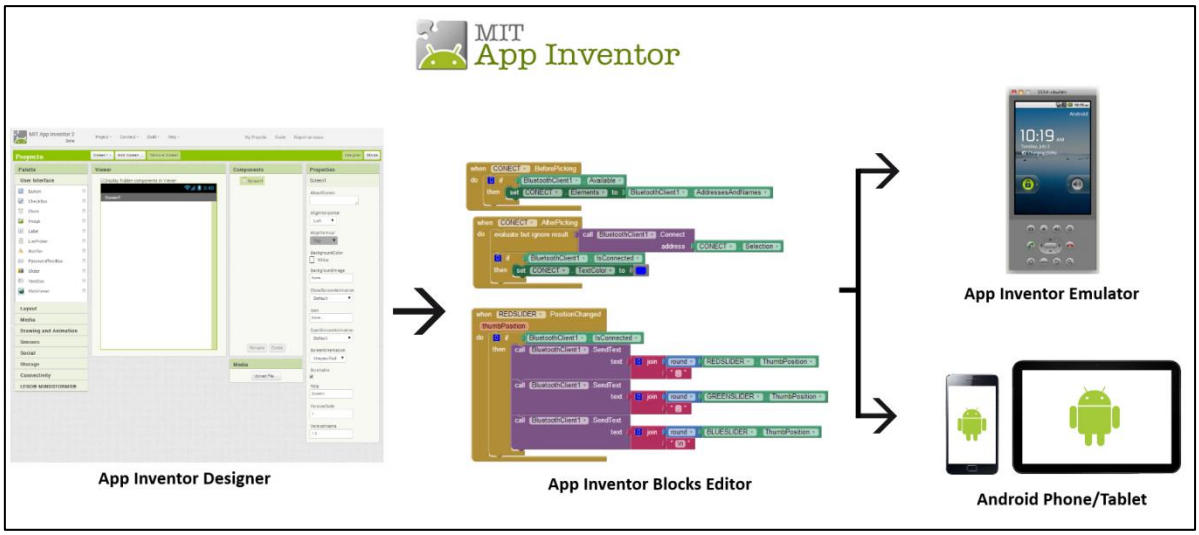


Figura 41. Construcció d'una l'aplicació

A la imatge anterior podem observar un organigrama del procés utilitzat poder crear qualsevol aplicació. Primer de tot cal construir el disseny gràfic, que es la part que es visualitzarà a la pantalla del dispositiu. Un cop s'ha creat la distribució dels diferents elements visuals es passa a l'editor, on a través de blocs es programarà el software de l'aplicació. Finalment, si tot està correcta es pot utilitzar l'aplicació en un mòbil o tauleta. Abans de transferir l'aplicació al telèfon es pot fer servir un emulador proporcionat per AppInventor per poder-lo simular i comprovar si tot funciona com s'havia desitjat.

5.1 Editor gràfic

El disseny de l'aplicació és la interfície que es mostrarà a l'usuari i la qual podrà manipular directament per comunicar-se amb el cotxe. Quan creem un nou projecte amb App Inventor ens apareixerà l'App Inventor Designer tal i com es pot observar a la següent imatge.

Figura 42. Pantalla de disseny gràfic

A la part central de la figura anterior es mostra la pantalla del dispositiu Android, és el lloc on s'han d'organitzar els diferents elements per obtenir la composició visual desitjada de la pantalla. Aquests components els podem trobar a la part esquerra de la pantalla, on hi tenim per una banda els botons, etiquetes de textos i imatges entre d'altres. Per altra banda també s'hi poden incorporar elements del mateix dispositiu Android com ara el bluetooh, el sensor d'orientació, la càmera... Això serà essencial per poder-nos comunicar amb el robot. Un dels elements essencials que s'ha utilitzat és el Canvas i és un requadre transparent ideal per posar-hi algun objecte o imatge al seu interior i poder-lo moure. Per afegir un element a la pantalla de visualització hem d'anar a la part esquerra i arrastrar-lo fins al lloc on vulguem. A la dreta de la imatge es pot observar una columna molt llarga, aquí és on es pot configurar cada component afegit a la pantalla com ara canviar el color, el tipus de lletra, la posició... També es poden afegir arxius de música, vídeo o gràfics.

Finalment, un cop aconseguit el disseny desitjat de l'aplicació, a l'extrem superior dret s'hi pot observar un botó per anar a l'editor de blocs on ens permet programar l'aplicació.

5.2 Editor de blocs

Per facilitar la programació del software s'utilitza un llenguatge visual a partir de blocs Java. Les llibreries d'aquests blocs han sigut desenvolupades i distribuïdes per l'Institut de Tecnologia de Massachusetts (MIT) i són de llicència lliure. En aquesta llibreria podem trobar eines per crear variables, funcions matemàtiques, lògiques o de control entre d'altres.

Gràcies a això es poden crear aplicacions de manera poc complexa i intuïtiva només utilitzant uns coneixements bàsics de Java.

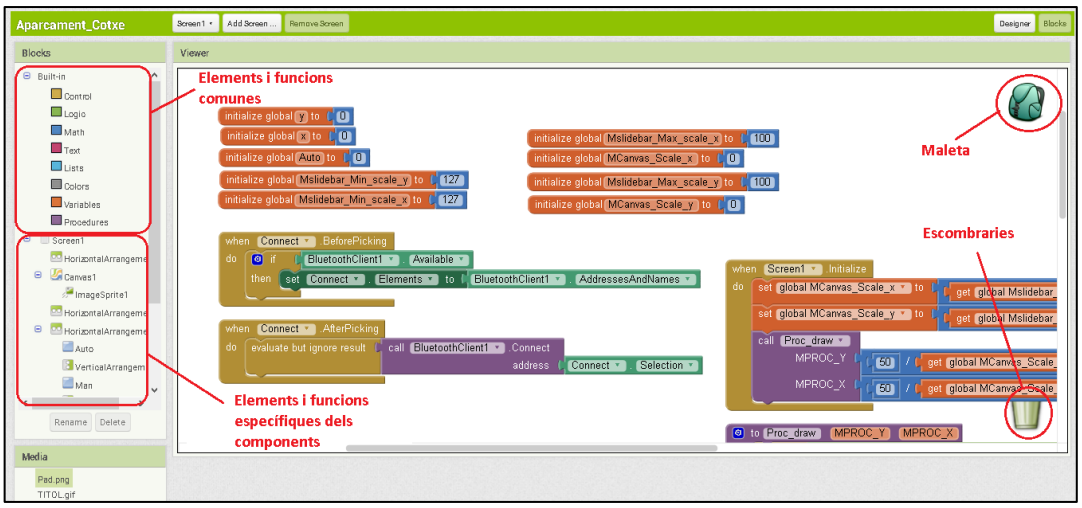


Figura 43. Editor de blocs

A la part esquerra de l'editor hi podem trobar els diferents blocs agrupats en diferents seccions: control, lògic, matemàtic, text, variables... A més també hi ha els diversos components que s'han afegit a la part del disseny gràfic per poder interactuar entre ells i configurar les diverses opcions de les quals disposen. Si hi ha alguns blocs o parts d'ells que són molt utilitzats en el programa es poden guardar a la maleta situada a l'extrem superior dret per poder-los copiar i enganxar.

Al centre de la figura s'hi poden veure una part dels blocs utilitzats per a la programació de l'aplicació, els quals s'aniran explicant detalladament a continuació. El primer que cal fer abans de començar en tot programa es declara les variables i inicialitzar-les.

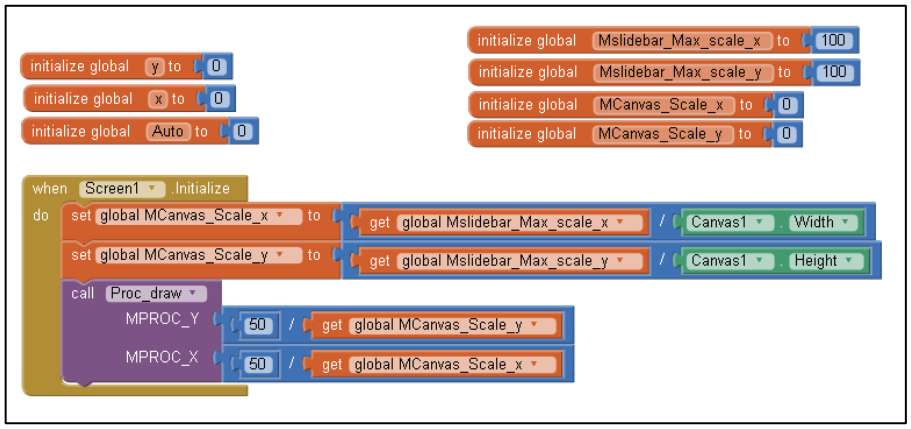


Figura 44. Declaració de variables

La figura 44 la separarem en dues parts per poder-la explicar millor, a dalt hi ha la declaració de les variables i a baix la inicialització de la pantalla. Les variables "Mslidebar_Max_scale_x" i "Mslidebar_Max_scale_y" de la columna de la dreta són per determinar els límits màxims de les coordenades del requadre per on podem moure el cursor. Les altres dues serviran per determinar la posició actual del cursor les quals s'aniran explicant amb més detall en el bloc corresponent.

Tant bon punt s'executa l'aplicació i s'obre la pantalla es pot utilitzar la funció d'inicialització la qual només serà executada una vegada. Per una banda, es fa servir per definir l'escala del Canvas de tal manera que si hem especificat que el límit màxim és de 100 i tenim que l'amplada del component és de 200 l'escala amb la que treballa és de ½. És a dir, si ens situem a la posició x de 150 el valor que ens donarà serà de 75. Per altra banda, la segona part del bloc fa una crida a la funció Proc_draw i s'assigna la posició 50 a MPROC_Y i MPROC_X per centrar el punt inicial al centre de la pantalla segons l'escala explicada anteriorment.

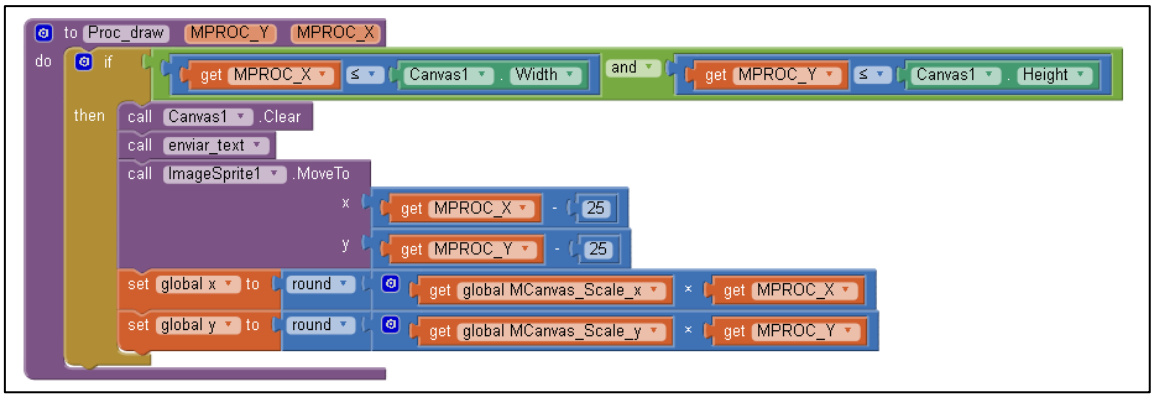


Figura 45. Funció Proc_draw

Aquesta funció està creada amb un bloc Procedure (procediment) que serveix per reunir una sèrie de de components per tal de crear una subrutina i poder-la cridar vàries vegades a dins del mateix programa. En aquest cas s'utilitza principalment per situar la bola o cursor a la posició que es defineix amb les variables MPROC_Y i MPROC_X sempre i quan estigui dins dels límits del Canvas. A més, també crida la subrutina d'enviar text que serà explicada més endavant.

La manipulació de la pantalla on hi ha el cursor es fa amb els següents blocs:

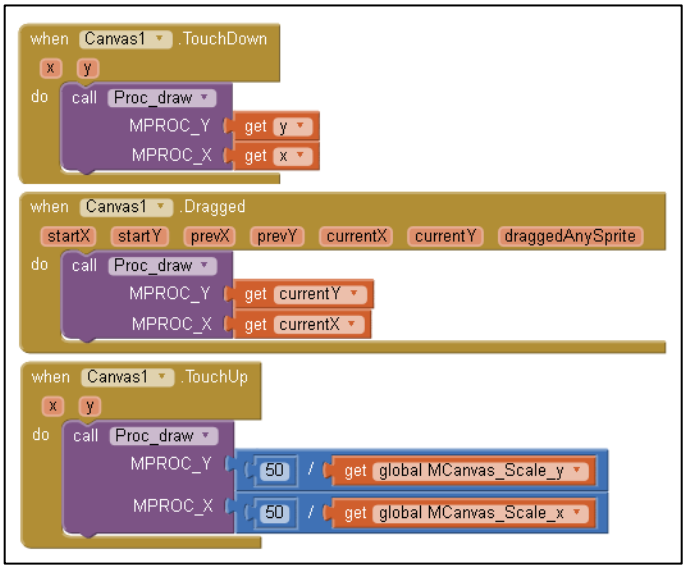


Figura 46. Manipulació del Canvas

Tal i com es pot veure en el primer bloc la subrutina Proc_draw es crida sempre que es toca la pantalla dins del requadre del Canvas donant la posició x i y de la pulsació. A més, en el bloc central, si mantens el cursor polsat i el vas movent per la pantalla actualitza el valor de les coordenades a través de les variables currentX, currentY i torna a cridar la funció Proc_draw. Finalment, en el moment que es deixa de mantenir el dit a la pantalla crida la subrutina per última vegada enviant-li la posició de repòs.

Una part imprescindible del programa és la comunicació amb l'Arduino per poder controlar els moviments del cotxe. Per fer-ho es fa amb Bluetooth i per tant també s'ha d'incorporar a la programació de l'aplicació.

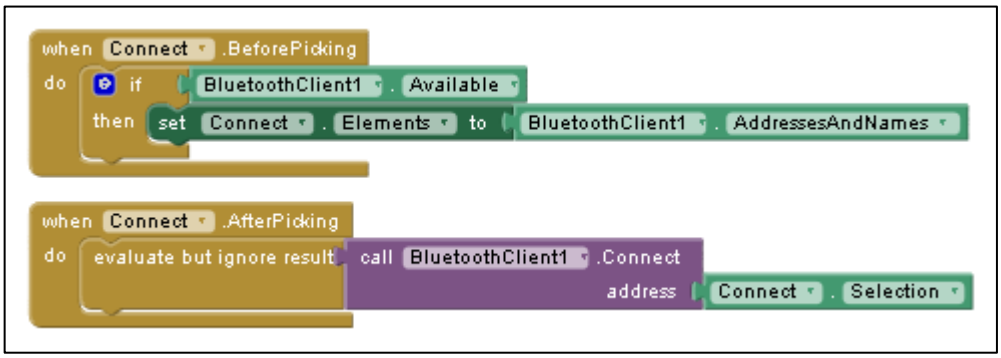


Figura 47. Connexió bluetooth

Per activar la connexió de Bluetooth es fa a través d'un botó creat al dissenyador gràfic. Abans que es premi, el programa comprova si hi ha connexió disponible amb algun dispositiu i les emmagatzemarà en una llista. Sempre i quan el Bluetooth del mòbil estigui activat, un cop premem el polsador es mostra la llista de disponibilitat on es podrà seleccionar el dispositiu desitjat.

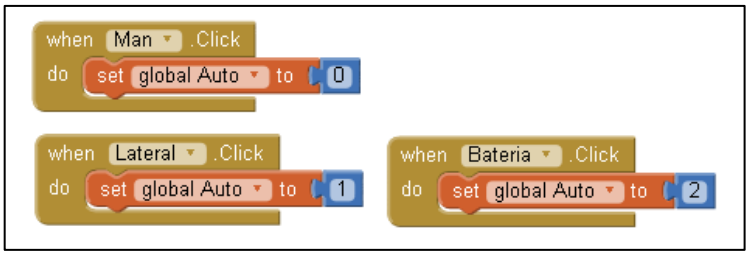


Figura 48. Polsadors Manual, lateral i bateria

A la figura 48 hi podem observar els tres blocs que s'utilitzen per seleccionar si volem controlar el cotxe manualment o activar la funció d'aparcament. Per fer-ho s'utilitzen tres polsadors i la variable Auto declarada a l'inici del programa enviant 0, 1 o 2 depenent del botó que es premi.

Per últim falta enviar la informació obtinguda de la manipulació de la pantalla a l'arduino per tal d'executar les operacions corresponents.

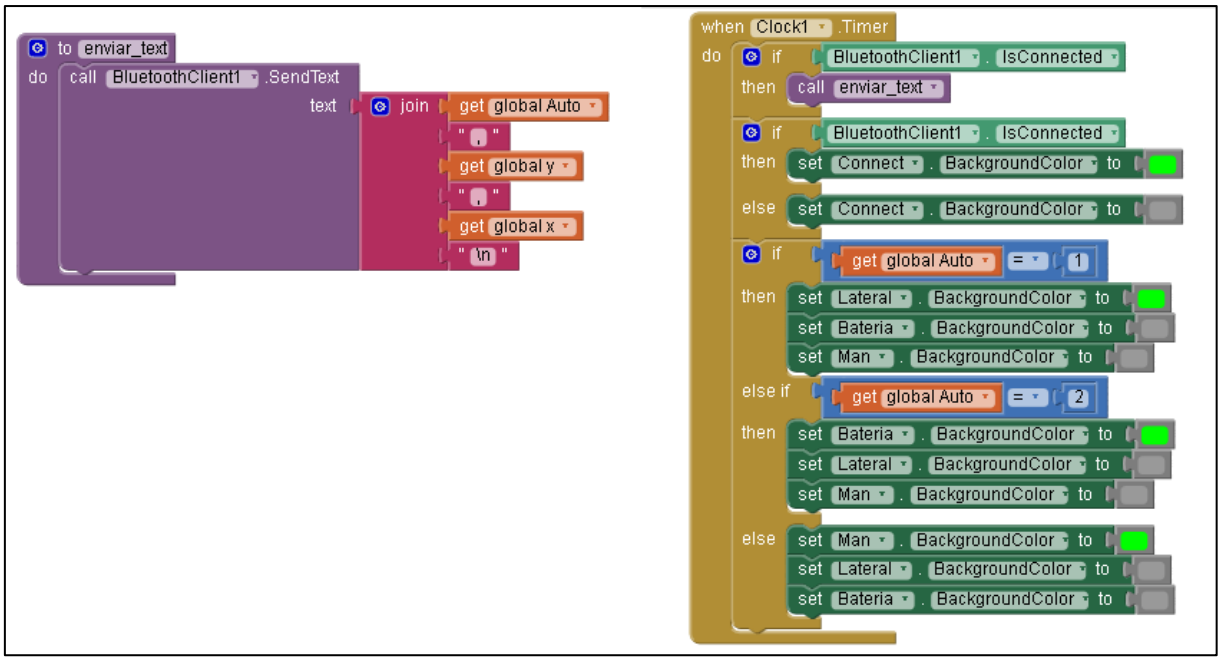


Figura 49. Enviar el text

A la imatge de la figura 49 hi ha dos blocs, una subrutina per enviar les dades a l'Arduino i l'altre està relacionat amb un temporitzador. El primer envia un 0, 1 o 2 depenent dels polsadors manual, lateral i bateria respectivament i les posicions y i x. La posició y serà la que controlarà el motor i la x servirà per posicionar el servo de la direcció.

El segon bloc utilitza el component Clock que és un temporitzador programat cada 500ms. Quan s'activa sempre i quan el Bluetooth estigui connectat crida la funció d'enviar el text per transferir les dades explicades en el paràgraf anterior. A més també s'encarrega de modificar el color dels botons depenent del seu estat, el color verd significa que està activat i el gris desactivat.

5.3 Compilació i instal·lació

Un cop finalitzada l'aplicació només falta posar-la a prova la qual cosa es pot fer de diverses maneres. Es pot fer servir un emulador de App Inventor o instal·lar-la directament al dispositiu Android. Per emular-la amb l'ordinador primer s'ha d'instal·lar el programa aiStarter i executar-lo, llavors a la pagina web de App inventor al menú superior hi ha un desplegable que hi posa Connect i s'ha de triar la opció Emulator. Una altra manera per fer la simulació és a través d'un dispositiu Android on també caldrà instal·lar una aplicació anomenada MIT ai2 Component. Un cop instal·lada s'ha d'executar i triar la opció Scan QR i també s'ha d'anar a la web i clicar la opció AI Companion del desplegable Connect. Després només caldrà escanejar el QR amb el dispositiu Android i s'executarà.

Si el que es desitja és instal·lar l'aplicació creada al mòbil o tauleta Android també es pot fer de dues maneres: utilitzant l'aplicació citada anteriorment per Android o bé compilant i creant un arxiu d'instal·lació. Tant si volem utilitzar el primer mètode com el segon s'ha d'utilitzar la opció desplegable de la web on hi posa Build, llavors es podrà triar si es desitja utilitzar un codi QR o bé guardar l'arxiu amb extensió apk.

Finalment a continuació es mostrarà l'aplicació instal·lada al meu dispositiu Android i s'explicarà com s'utilitza la interfície.

Figura 50. Interfície de l'aplicació

A la figura 50 podem observar que al centre de la imatge hi ha una bola negra amb quatre fletxes en diferents direccions a dins d'un requadre gris. Aquesta s'utilitza com un joystick, el qual es pot desplaçar per dins del requadre. Per tant si es mou la bola a l'extrem superior dret el cotxe es desplaçarà endavant amb la direcció màxima cap a la dreta.

Tant bon punt s'inicia l'aplicació s'ha de pressionar el polsador de connectar que es pot veure a la part inferior dreta per poder establir una comunicació a través de Bluetooth amb el dispositiu. Cal dir que s'ha de tenir la opció de Bluetooth connectada en mòbil, en aquest cas al polsar el botó s'obrirà una pantalla amb tots els dispositius detectats. Si la connexió s'ha establert correctament el botó quedarà de color verd. En el cas que en qualsevol moment es perdi la connexió el botó tornarà en estat gris i es mostrarà un error per pantalla.

Llavors a la part inferior esquerra podem observar dos polsadors, un anomenat lateral i l'altre bateria. No es poden tenir els dos botons activats a la vegada, en tot moment es mostra de color verd el que està activat. Amb el polsador manual pressionat es pot controlar el cotxe manualment amb el joystick anomenat anteriorment. En canvi quan es polsi l'aparcament lateral o bateria el joystick queda inhabilitat i el robot comença a realitzar la trajectòria d'aparcament programada en cada cas. En qualsevol moment de la maniobra es pot canviar de mode d'operació sense cap problema, quan es pressioni de nou es reiniciarà l'aparcament des de la posició inicial.

6. PROGRAMARI

A continuació es mostraran els organigrames dels programes utilitzats en Arduino i Labview juntament amb una breu explicació per tal de fer-lo mes entenedor.

6.1 Arduino

El programa de l'Arduino s'ha escrit amb el llenguatge del mateix dispositiu que parteix de la base del C. Aquest dispositiu, és l'element principal de control del robot, amb el qual s'ha utilitzat com a tarja de sensorització al PC a través del port sèrie i poder així tenir un sistema d'adquisició de dades, amb les quals posteriorment s'ha pogut realitzar el propi controlador.

6.1.1 Adquisició de dades

Per l'elaboració del projecte s'han utilitzat diversos programes diferents, els dos primers que es mostraran són dos programes senzills que s'han utilitzat per l'adquisició de les dades de velocitat i de direcció en aquest mateix ordre.

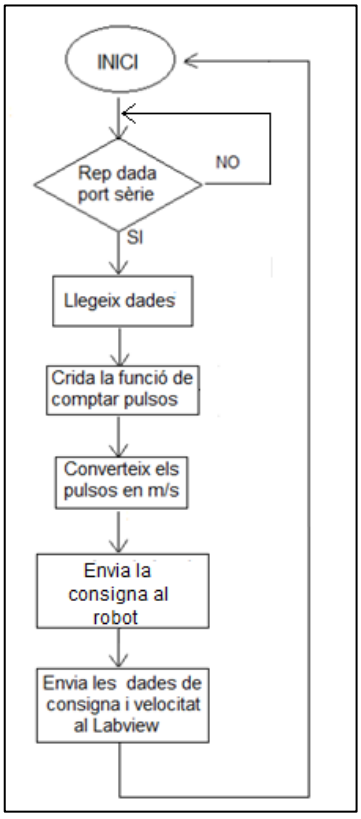


Figura 51. Organigrama d'adquisició de velocitat

Tal i com es pot observara la figura 51, primerament, rebem les dades del port sèrie si aquestes són captades prosseguirà a llegir les dades de l'encoder, en cas contrari restariaen aquest bucle fins que s'aconseguí establir connexió.

Un cop finalitzada la subrutina fa la conversió dels pulsos a velocitat lineal en metres per segon i envia la consigna al motor. Per últim transmet les dades de consigna de PWM i de velocitat del cotxe al Labview per tal de poder-les enregistrar.

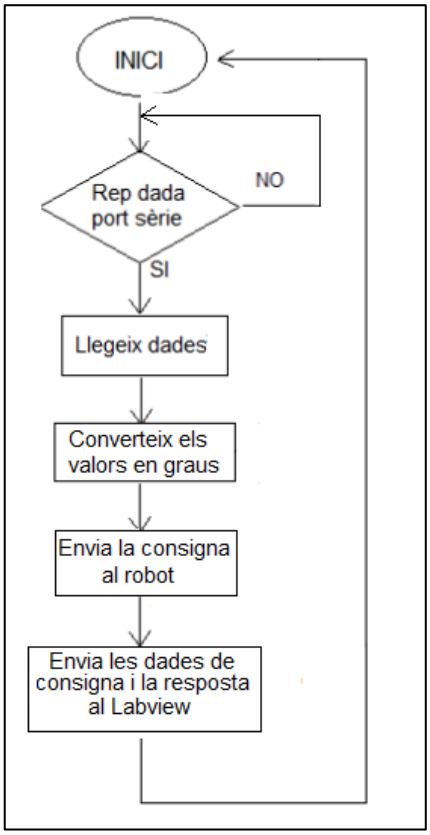


Figura 52. Organigrama d'adquisició dels graus de gir

A l'organigrama de la figura 52 es pot veure el procés realitzat pel programa d'aquisició dels graus de gir de la direcció. Primer, si rebem les dades del port sèrie llegirà el valor analògic de l'entrada, sinó esperarà que la connexió sigui correcta.

A continuació, fa la conversió del valor rebut a graus i escriu sobre la sortida analògica el valor de la consigna que serà enviada cap al servomotor i finalment envia les dades de consigna i la resposta en graus de gir al Labview.

6.1.2 Control de maniobres

Tot seguit es passarà a explicar esquemàticament el programa final que s'ha bolcat a l'Arduino a fi efecte de poder controlar els moviments del cotxe radio-control.

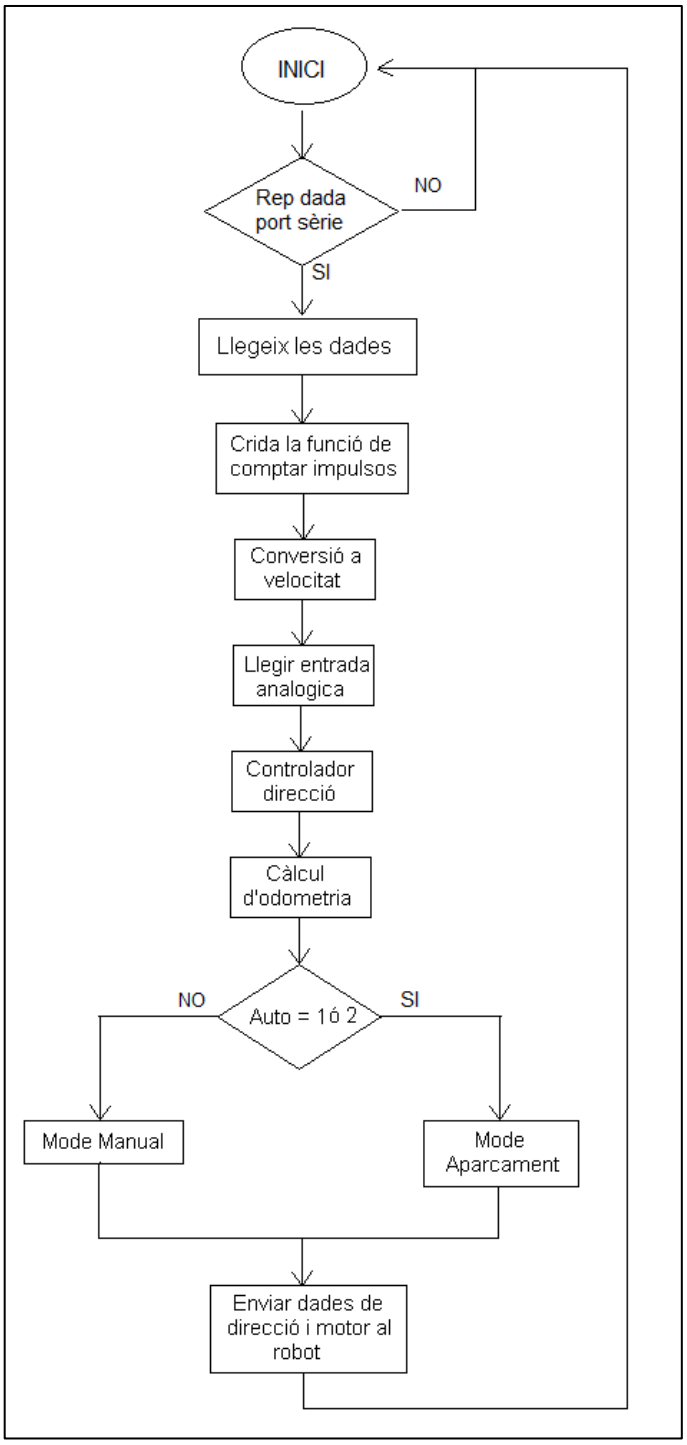


Figura 53. Organigrama de control cotxe RC

Passem ha explicar l'organigrama de la figura 53 per tenir-ne una idea de com poder realitzar el programa d'Arduino a partir d'aquest.

Primerament, rebem les dades del port sèrie si aquestes són captades pels sensors i l'Arduino les pot llegir sinó tornarà anar a l'inici ja que no començaríem cap procés. A continuació crida la funció encarregada de comptar els impulsos de l'encoder durant 100ms. Aquí es treballa amb dos temporitzadors de l'ATMEGA un per comptar els flancs dels impulsos i l'altre per fer un càlcul de temps amb un interval de 1 ms creant una interrupció. Aquesta és l'encarregada d'assignar el valor del comptador un cop passat els 100ms i reiniciar els temporitzadors. Tenint en compte el cicle de comptatge i els impulsos per volta, juntament amb la relació de transmissió ja explicada, es converteix el valor a velocitat.

Posteriorment amb el valor obtingut de l'entrada analògica del potenciòmetre situat al servomotor de la direcció s'executa el controlador PI. Un cop obtinguts els resultats de la velocitat i la direcció del cotxe es calcula la posició del cotxe a través de la odometria. Un cop això, hi ha dos branques el mode manual i el d'aparcament.

Per una banda, si el valor Auto rebut des de l'aplicació d'Android és igual a 0 s'executa el cicle manual, de manera que els valors enviats al servo del motor i de la direcció seran els que es reben via Bluetooth assignats per l'aplicació (després de processar-los). D'altra banda, quan es rep un 1 s'executa el mode d'aparcament seguint les maniobres explicades a l'apartat corresponent. Finalment, s'enviaran les dades cap als servos del corresponents del robot mòbil.

6.2 Labview

A continuació tal i com s'ha fet pel programa de l'Arduino tenim l'organigrama del programa del Labview a fi d'efecte de poder-ne obtenir una idea més clara i precisa per a partir d'aquest poder realitzar el mateix programa. A posteriori el passarem ha explicar:

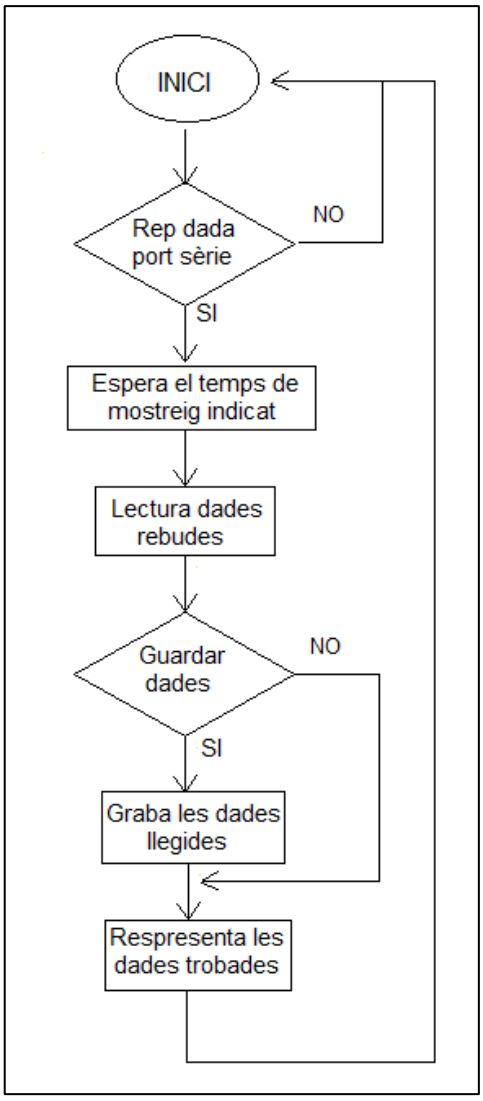


Figura 54. Organigrama Labview

Primer de tot com s'ha pogut observar en l'organigrama anterior es rep primer la dada del port sèrie si aquesta no ha arribat torna a l'inici. Seguidament llegeix les dades rebudes per l'Arduino segons el temps de mostreig i les processa fent les conversions de cadenes de nombres pertinents. Si s'ha polsat el botó de guardar les dades començarà a emmagatzemar i sinó seguirà el procés mostrant les dades de recepció en pantalla.

7. RESUM DEL PRESSUPOST

El present projecte del control de les maniobres d'un robot mòbil està valorat amb un pressupost total de dos mil cinc-cents trenta-quatre euros amb vint-i-cinc cèntims, sense IVA.

8. CONCLUSIONS

Amb la construcció d'aquest projecte de robòtica s'han pogut posar en pràctica molts dels coneixements assolits durant la carrera i assentar-los. A més, com que la robòtica és molt transversal enllaça tots aquests coneixements relacionant-los entre ells per aconseguir un mateix objectiu.

Totes les expectatives a realitzar per aquest projecte s'han complert correctament tot i les limitacions amb les que s'han topat. S'ha de tenir en compte que el robot utilitzat és una maqueta de joguina d'un cotxe radio-control i això ha fet que no es pogués exprimir tant com es volia. Per exemple no s'ha pogut realitzar el controlador de la velocitat. A l'hora de realitzar les maniobres d'aparcament des d'un principi es desitjava utilitzar sensors de proximitat, per fer-ho més similar als sistemes d'aparcament implementats als cotxes reals. Malgrat a això, es va haver de desistir degut a la complexitat i cost del sistema complert.

Pel que fa a la programació d'Arduino s'han adquirit nous coneixements com ara la programació directe de l'Atmega per comptar els flancs de pujada dels impulsos de l'encoder.

Per tant, finalment es pot afirmar s'han assolit els objectius inicials correctament i s'ha aconseguit realitzar l'aparcament del cotxe radio-control a través d'un dispositiu Android, tot simulant la tecnologia explicada inicialment del nou sistema d'aparcaments dels cotxes d'Audi.

Narcís Casellas Arbat
Grau en Enginyeria Electrònica i Automàtica

La Bisbal d'Empordà, 28 de desembre de 2015

9. RELACIÓ DE DOCUMENTS

El projecte consta de cinc documents: Memòria, Plànols, Plec de condicions, Estat d'amidaments i Pressupost.

10. BIBLIOGRAFIA

ARDUINO. Diagrama pinout Arduino UNO. (http://forum.arduino.cc/index.php?topic=146315.0, 15 de març de 2015).

ARDUINO. Descripció de les funcions completes del llenguatge d'Arduino (https://www.arduino.cc/en/Reference/HomePage, 25 d'octubre de 2014).

ATMEL. Microcontrolador ATmega . (http://www.atmel.com/images/Atmel-8271-8-bit-AVR-Microcontroller-ATmega48A-48PA-88A-88PA-168A-168PA-328-28P_datasheet_Complete.pdf, 20 de març de 2015).

BYCMO. Pinyó z-24. (http://www.bycmo.com/catalog/product/view/id/776/s/pi-on-z-24-1u/, 18 de setembre de 2014).

ELECTRONICA60NORTE. Mòdul bluetooth HC-05. (http://www.electronica60norte.com/mwfls/pdf/newBluetooth.pdf, 24 de setembre de 2014).

FORTINO. Timer 1 del AVR del ATmega32. SCRIBD. (http://es.scribd.com/doc/19073773/Capitulo7-Timer1-del-AVR-del-ATmega32-espanol, 25 d'agost de 2015).

GAMMON, NICK. Frequency Counter sketch for Atmega328. GAMMON FORUM. (http://www.gammon.com.au/forum/?id=11504, 12 de març de 2015).

PACHECO, LLUÍS. Apunts de Sistemes robotitzats, Robots mòbils. Publicació Udg de 2014.

PÉREZ ANTONIO. Motor EMG30. SCRIBD. (http://es.scribd.com/doc/167830327/Motor-EMG30, 13 de desembre de 2014).

PFAFFENDORF, FEDERICO. Alimentació de l'Arduino a 9V. WORDPRESS. (https://fpfaffendorf.wordpress.com/2013/07/18/arduino-uno-alimentado-a-bateria-de-9v/, 18 de juliol de 2015).

SOLOELECTRONICOS. Tutorial App Inventor (http://soloelectronicos.com/2014/04/21/como-se-programan-apps-moviles-con-mit-app-inventor-bloques-logicos/, 21 d'abril de 2015).

SPHINX. Ús de controladors i timers de ATmega. (http://sphinx.mythic-beasts.com/~markt/ATmega-timers.html, 25 de març de 2015).

VIME, JAVIER. Variadores de velocidad en Radiocontrol. Radiocontrol. Vol. 1. Fascículo 45. p. 17-18. 2002

11. GLOSSARI

MIT: Massachusetts Institute of Technology

PWM: Pulse Width Modulation

SO: Sistema Operatiu

USB: Universal Serial Bus

A. PROGRAMA ARDUINO

En aquest capítol de l'annex es mostren tots els codis dels programes Arduino realitzats en el transcurs del projecte i es donaran les taules amb totes les variables utilitzades en cadascun d'ells.

A.1 Programa de configuració del Bluetooth

El programa que es mostra a continuació és el que es va utilitzat per configurar el mòdul de comunicació de Bluetooth HC-05. El que es fa és crear un segon port sèrie de manera que un és per enviar dades al dispositiu i l'altre és per rebre la resposta. Llavors mitjançant el monitor sèrie del programa Arduino et permet comunicar-te amb el mòdul mitjançant comandes AT.

La connexió realitzada entre el mòdul i la placa és el següent:

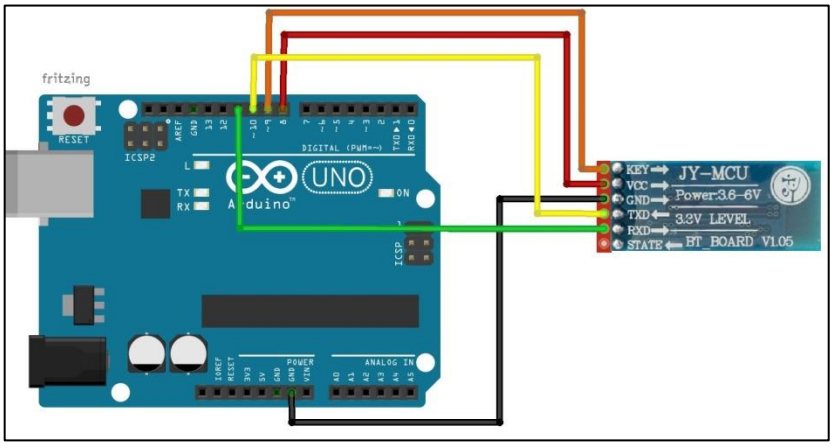


Figura 55. Connexió Arduino-Bluetooth

En primer lloc, per tal que el mòdul entri en el mode de comandes AT, cal que el pin KEY estigui HIGH. És per això que l'alimentació s'ha connectat al pin 8 i no directament a 5V. De tal manera que primer posarem a HIGH el pin 9 i després el 8 per alimentar-lo i entrar en el mode de comandes.

Els pins TXD i RXD es col·loquen als pins 10 i 11 consecutivament. Per tal de crear dos ports de comunicació s'utilitza la llibreria SoftwareSerial com es pot observar en el següent programa:

```
#include <SoftwareSerial.h>  //Llibreria de comunicació sèrie

SoftwareSerial serial2(10,11);  //Crea un port sèrie entre el pin 10
//i 11

void setup(){
Serial.begin(9600);  //Establim la velocitat del primer port
serial2.begin(38400);  //Establim la velocitat del segon port
pinMode(8,OUTPUT);  //Pin d'alimentació
pinMode(9,OUTPUT);  //Pin KEY
digitalWrite(9,HIGH);  //Activem el pin KEY
delay(100);  //Esperem 100 ms
digitalWrite(8,HIGH);  //Activem l'alimentació del mòdul

Serial.println("Inici de comandes AT: ");  //Notificació per saber
//que ja estem comunicant
}

void loop()  //Inici del loop
{

  if(Serial.available())
    serial2.write(Serial.read());  //Envia pel port 2 el que hem
//escrit al monitor sèrie

  if(serial2.available())
    Serial.write(serial2.read());  //Escriu al monitor sèrie la
//resposta rebuda pel mòdul

}  //Fi del loop
```

A.2 Programa d'adquisició de la direcció

El codi del programa que es mostra ja esta comentat a cada línia d'execució i el funcionament global ja s'ha explicat a l'apartat de programari, per tant aquí tant sols es donarà el programa juntament amb les entrades i sortides.

Pin	Tipus	Nom variable	Descripció
A0	Analog IN	lecturaPoten	Lectura del valor del potenciòmetre
~11	Digital OUT	direccio	Sortida en PWM del valor de gir del servo de la direcció

Taula 1. Entrades i sortides

Com podem observar s'utilitza una entrada i una sortida, l'entrada serveix per llegir el valor que ens proporciona el potenciòmetre, el qual posteriorment s'ha d'escalar per saber els graus. El valor de sortida és de PWM per tant un valor de 0-255 per posicionar el servomotor a la posició donada per la consigna. Tot seguit es dóna el codi del programa utilitzat en aquesta part.

```
//Declarem les variables:
int lecturaPoten = A0; //Pin d'entrada analògica per la lectura del
potenciometre del servo
int valorPoten = 0, direccio;
float comptador, valor, valor_ant;
boolean flanc;

void setup()
{

  // Establim la velocitat per transmetre dades pel port sèrie
  Serial.begin(9600);

}
```

```
void loop()
{
//S'encarrega de canviar la consigna cada "x" temps
comptador = random(0,3000); //El tems és aleatori de 0 a 3 segons
valor=millis();
if (valor-valor_ant > comptador) //Condició per fel el canvi de
//consigna segons el temps
{

    if (flanc == 0)
        {
         direccio = 100; //Consigna 1
         flanc = !flanc; //Alterna el valor de flanc que per fer el
//canvi de consigna
    }else
       {
         direccio=150; //Consigna 2
         flanc = 0;
        }
   valor_ant=valor; //Es passa el valor de tems actual a la variable
//del valor anterior
}

// Llegim el valor analògic del potenciòmetre
valorPoten = analogRead(lecturaPoten);
int graus= map(valorPoten,0,1023,-120,120); //Escalem de -120 a 120°

analogWrite(11,direccio); //Escriu el valor a la sortida

//Escribim en el port sèrie els valors que ens interessen
Serial.print (";");// Limita els nombres per poder-los diferenciar
//amb el labview
Serial.print (direccio);// Consigna del servo
Serial.print (";");// Limita els nombres per poder-los diferenciar
//amb el labview
Serial.println(graus);// Lectura dels graus de la direcció
delay (100); }
```

A.3 Programa d'adquisició de velocitat

Aquest programa té el mateix objectiu que l'anterior però enlloc de llegir els valors de la direcció es volen llegir el de la velocitat. Tot i això és bastant diferent a l'hora de processar el valors d'entrada ja que aquí llegim pulsacions i per tant s'ha d'utilitzar un comptador d'impulsos.

Pin	Tipus	Nom variable	Descripció
D5	Digital IN	---	Compte els flancs de l'encoder
~11	Digital OUT	motor	Sortida de PWM del servomotor

Taula 2. Entrades i sortides

A la taula 2 es pot veure que hi ha una entrada digital (D5) per comptar els flancs de pujada dels impulsos de l'encoder. No té una variable assignada perquè s'assigna directament amb llenguatge de l'Atmega. Igual que en el programa anterior tenim una sotida de PWM encarregada d'aplicar la consigna assignada al servo del motor. Tot seguit es mostra el codi del programa.

```
// Input: Pin D5
//Declaració de variables:
boolean flanc;
float comptador,valor, valor_ant, velocitat;
int motor;

//Declaració de variables del comptador de pulsos:
volatile unsigned long timerCounts;
volatile boolean counterReady;
float count;
long pulsos;
unsigned int timerTicks;
unsigned int timerPeriod;

//Funció de comptatge de pulsos de l'encoder del motor
void startCounting (unsigned int ms)
```

```
  {

  counterReady = false;          // temps inferior a 100 ms
  timerPeriod = ms;              // quants ms ha de comptar
  timerTicks  =  0;                          // reinicia el comptador
d'interrupció

  // reset Timer 1 i Timer 2
  TCCR1A = 0;
  TCCR1B = 0;
  TCCR0A = 0;
  TCCR0B = 0;



  // Timer 1 - compte els flancs del pin D5
  // Timer 2 - ens dóna un interval de 1ms
  // El rellotge és de 16 MHz  (62.5 nS per tic) - prescaled de 64
  // El comptador incrementa cada 8 µS.
  // per tant nosaltres comptem 250 vegades per tenir 1000 µS (1 mS)
  TCCR0A = bit (WGM01) ;   // CTC mode, és un mode de comparació
  OCR0A  = 249;              // compte fins a 250  (zero relatiu) és
el valor que ha de comparar, és a dir cada 1ms
  // Timer 2 - interromp cada 1 ms
  TIMSK0 = bit (OCIE0A);   // habilita la interrupció del Timer 2

  TCNT1 = 0;  // Inicialitza els dos comptador a 0
  TCNT0 = 0;

  // Reinicia els prescalers
  GTCCR = bit (PSRASY);
  // Inicia el Timer 2
  TCCR0B =  bit (CS01) | bit (CS00) ;  // Prescaled de 64
  // Comença el Timer 1
  // Configura el pin (D5) com una entrada que incrementa a cada
flanc de pujada
  TCCR1B =  bit (CS10) | bit (CS11) | bit (CS12);
  }  // fi de startCounting
```

```
//*****************************************************************
//  La interrupció del Timer2 és cridada cad 1ms
//  16Mhz / 64 / 250 = 1000

ISR (TIMER0_COMPA_vect) // (Rutina d'interrupció) cada 1ms
  {
  // Asigna el valor del comptador
  unsigned int timer0CounterValue;
  timer0CounterValue = TCNT1;  // li dóna el valor del comptador


  // Mira si s'ha arribat al temps de 100ms
  if (++timerTicks < timerPeriod)
    return;  // si encara no hi ha arribat retorna



  // Quan ja hi ha arribat es paren els Timers per veure els pulsos

  TCCR1A = 0;    // stop timer 1
  TCCR1B = 0;

  TCCR0A = 0;    // stop timer 2
  TCCR0B = 0;

  TIMSK1 = 0;    // Desactiva la interrupció del Timer1
  TIMSK0 = 0;    // Desactiva la interrupció del Timer2


  // Calcula el valor

  timerCounts = timer0CounterValue;  // valor actual del comptador

  counterReady = true;               // set global flag for end count
period
```

```
  }  // end of TIMER2_COMPA_vect


void setup ()
  {
  Serial.begin(9600);
//  Serial.println("Frequency Counter");

  } // end of setup

void loop ()
  {
  //Crida la funció de comptar els impulsos
  startCounting (100);  // cada 100 ms crida la funció

  while (!counterReady)
     { }  // Crea un bucle mentre no ha acabat de llegir els pulosos
que dura 100 ms

//S'encarrega de canviar la consigna cada "x" temps
comptador = random(0,3000);
valor=millis();
if (valor-valor_ant > comptador)   //Condició per fel el canvi de
consigna segons el temps
{

    if (flanc == 0)
        {
         motor = 140; //Consigna 1
         flanc = !flanc; //Alterna el valor de flanc que per fer el
canvi de consigna
    }else
       {
         motor = 130; //Consigna 2
         flanc = 0;
        }
```

```
   valor_ant=valor; //Es passa el valor de tems actual a la variable
del valor anterior
}

   //Converteix el valor del comptador de pulsos a velositat
   pulsos = timerCounts;
   velocitat                                                    =
(((((((pulsos*10*60)/90)*1.5)/4.4)/2.75)*2*PI)/60)*0.032*100;
   //Tenim els pulsos cada 0.1s per tant, multiplicant per 10 tenim
els pulsos/segon per 60
   //segons tenim els pulsos per minut i si 90 pulsos son una volta
tenim rpmi despres fem
   //les relacions de transmissió i ho passem a cm/s

   //Escribim en el port sèrie els valors que ens interessen
   analogWrite(11,motor);
   Serial.print   (";");//   Limita   els   nombres   per   poder-los
diferenciar amb el labview
   Serial.print(motor);
   Serial.print   (";");//   Limita   els   nombres   per   poder-los
diferenciar amb el labview
   Serial.println (velocitat);




  // let serial stuff finish
  //delay(100);


  } // Fi del loop
```

A.4 Programa de control de maniobres

En aquest capítol es mostra el programa d'Arduino final que s'ha utilitzat per controlar el robot mòbil. Hi ha integrat el controlador, el comptador de pulsos, la comunicació via bluetooth i el control de trajectòries .

Pin	Tipus	Nom variable	Descripció
A0	Analog IN	lecturaPoten	Lectura del valor del potenciòmetre
D0	Digital IN	---	Recepció de dades
D1	Digital OUT	---	Transmissió de dades
~3	Digital OUT	direccio	Sortida en PWM del valor de gir del servo de la direcció
D5	Digital IN	---	Compte els flancs de l'encoder
~11	Digital OUT	motor	Sortida de PWM del servo del motor

Taula 3. Entrades i sortides

A la taula anterior es poden observar les entrades i sortides que s'han utilitzat en el programa de control del robot mòbil. Les entrades i sortides A0, D5 i ~11ja s'han xomentat en els dos apartats anteriors i el ~3 és el mateix que s'havia utilitzat com a sortida del primer programa però s'ha canviat el pin ja que en aquest programa hi ha dues sortides PWM. Els pins D0 i D1 no surten específicament en el programa d'arduino però son els encarregats de la comunicació amb el dispositiu de Bluetooth. El D0 és l'encarregat de rebre les dades del dispositiu i en canvi el D1 és el que les envia.

```
//**Programació de les maniobres d'aparcament**//


//Declaració de les variablesdel comptador de pulsos

volatile unsigned long timerCounts;
volatile boolean counterReady;
float pulsos;
unsigned int timerTicks, timerPeriod;
```

```
//Variables generals
float velocitat, pwm_direccio, pwm_ant=0, error, error_ant, graus;
int motor, Recepcio, sentit, lecturaPoten = A0, valorPoten = 0,
direccio, Auto, pwm, pwm2;

//Variables d'odometria
float teta, x, y, L, x_ant, y_ant, teta_ant, omega;
int posicio=-1, estat;

//Funció de comptatge de pulsos de l'encoder del motor
void startCounting (unsigned int ms)
  {

  counterReady = false;          // temps inferior a 100 ms
  timerPeriod = ms;              // quants ms ha de comptar
  timerTicks = 0;                          // reinicia el comptador
d'interrupció

  // reset Timer 1 i Timer 2
  TCCR1A = 0;
  TCCR1B = 0;
  TCCR0A = 0;
  TCCR0B = 0;



  // Timer 1 - compte els flancs del pin D5
  // Timer 2 - ens dóna un interval de 1ms
  // El rellotge és de 16 MHz  (62.5 nS per tic) - prescaled de 64
  // El comptador incrementa cada 8 µS.
  // per tant nosaltres comptem 250 vegades per tenir 1000 µS (1 mS)
  TCCR0A = bit (WGM01) ;   // CTC mode, és un mode de comparació
  OCR0A  = 249;             // compte fins a 250  (zero relatiu) és
el valor que ha de comparar, és a dir cada 1ms
  // Timer 2 - interromp cada 1 ms
  TIMSK0 = bit (OCIE0A);   // habilita la interrupció del Timer 2

  TCNT1 = 0;  // Inicialitza els dos comptador a 0
```

```
  TCNT0 = 0;

  // Reinicia els prescalers
  GTCCR = bit (PSRASY);
  // Inicia el Timer 2
  TCCR0B =  bit (CS01) | bit (CS00) ;  // Prescaled de 64
  // Comença el Timer 1
  // Configura el pin (D5) com una entrada que incrementa a cada
flanc de pujada
  TCCR1B =  bit (CS10) | bit (CS11) | bit (CS12);
  }  // fi de startCounting


//****************************************************************
//  La interrupció del Timer2 és cridada cad 1ms
//  16Mhz / 64 / 250 = 1000

ISR (TIMER0_COMPA_vect) // (Rutina d'interrupció) cada 1ms
  {
  // Asigna el valor del comptador
  unsigned int timer0CounterValue;
  timer0CounterValue = TCNT1;  // li dóna el valor del comptador


  // Mira si s'ha arribat al temps de 100ms
  if (++timerTicks < timerPeriod)
    return;  // si encara no hi ha arribat retorna



  // Quan ja hi ha arribat es paren els Timers per veure els pulsos

  TCCR1A = 0;    // stop timer 1
  TCCR1B = 0;

  TCCR0A = 0;    // stop timer 2
  TCCR0B = 0;
```

```
  TIMSK1 = 0;    // Desactiva la interrupció del Timer1
  TIMSK0 = 0;    // Desactiva la interrupció del Timer2


  // Calcula el valor

  timerCounts = timer0CounterValue;  // valor actual del comptador

  counterReady = true;               // set global flag for end count
period
  }  // end of TIMER2_COMPA_vect

// inici del setup
void setup ()
  {
  Serial.begin(9600);



  }
// fi del setup

//inici del Loop
void loop ()
  {

while (Serial.available() > 0) //Quan hi ha recepció de dades les
llegeix
    {
       int Mode = Serial.parseInt(); //Llegim el primer valor enter
(0-1) i el guardem a la variable

       int y_pant = Serial.parseInt(); //Llegim el primer valor
enter (0-100) i el guardem a la variable

       int x_pant = Serial.parseInt(); //Llegim el primer valor
enter (0-100) i el guardem a la variable
```

```
        if (Serial.read() == '\n') //Si rebem \n significa que la
primera sèrie s'ha acabat
        {
         Auto = Mode;
         pwm = y_pant;
         pwm2 = x_pant;
        }

    } //Fi de la recepció de valors

//Crida funció comptatge
  startCounting (100);    // durant 100 ms crida la funció de
comptatge de pulsos de l'encoder
  while (!counterReady)
     { }   // crea un bulce fins que el temporitzador no hagi
finalitzat

//Conversió dels pulsos de l'encoder a velocitat
   pulsos = timerCounts;

//Determinem la velocitat del vehicle
   if(motor < 175){sentit = -1;} //Determina el sentit de gir
   else if (motor > 175){sentit = +1;}
   /* Tenim els pulsos cada 0.1s per tant, multiplicant per 10 tenim
      els pulsos/segon per 60 segons tenim els pulsos per minut i si
90 pulsos
      són una volta tenim rpm; després fem les relacions de
transmissió i ho passem a cm/s
    */
   velocitat                                                    =
((((((pulsos*10*60)/90)*1.5)/4.4)/2.75)*2*PI)/60)*0.031*100*sentit;


//Lectura analògica dels graus de la direcció
   valorPoten = analogRead(lecturaPoten);
```

```
   graus=   map(valorPoten,0,1023,-120,120);   //Escalem   el   valor
analogic a graus de -120 a 120°

//CONTROLADOR DIRECCIÓ
  error = direccio - graus; //L'error és la consigna menys l'angle
de gir actual
  pwm_direccio = 180; //inicialitzem
  pwm_direccio=(error)*1.35-(error_ant)*0.78+pwm_ant;     //Regulador
dels graus de direcció

  //Limitació dels limits superior i inferior del controlador
  if(pwm_direccio>255)
    pwm_direccio=255;
  if(pwm_direccio<50)
    pwm_direccio=50;

  //Actualitzem els valors anteriors d'error i de pwm
  error_ant=error;
  pwm_ant=pwm_direccio;

  //Odometria:
  L=25.5; //distancia entre l'eix de la direcció i el de la tracció
  omega= (graus*3.15)/180; //Conversió de graus a radiants
  teta = (0.1*velocitat/L)*tan(omega)+teta_ant;
  x = 0.1*velocitat * cos(teta)+ x_ant;//multiplica per 0.1 perquè
mostreja cada 100ms i en canvi la velocitat és cm/s
  y = 0.1*velocitat * sin(teta)+ y_ant;
  //Actualitzem els valors anteriors de x, y i teta
  x_ant = x;
  y_ant = y;
  teta_ant = teta;


 if (Auto == 0) //si Auto és 0 significa que estem en mode manual
         {
          //Assignem a les variables del motor i direcció els valors
enviats des de l'aplicació Android
```

```
         motor= map(pwm,100,0,130,225); //Com que els valros rebuts
van de 0-100 els hem d'escalar
         direccio=  map(pwm2,0,100,-30,30);//Com  que  els  valros
rebuts van de 0-100 els hem d'escalar
         x=0; //inicialitzem les posicions d'aparcament a (0,0)
         y=0;

          }

//Màquina d'estats per realitzar la maniobra d'aparcament
 else if (Auto == 1) //Si Auto és 1 inicia l'aparcament lateral
  {
  motor=100;
  direccio=0;
  switch(posicio)
    {
       //Estat inicial de repòs.
       case -1:
        direccio = -30;
        posicio = 0;
       break;

       case 0:
         motor = 225; //Endavant
         direccio = -30;

          if( x < 15 )
            {
             posicio = 0;
            }
          else if (x >= 15)
            {
              posicio=1;
            }
        break;
```

```
//Posició 1
case 1:
 motor = 100;
 direccio = 30;
   if( x > -5 )
    {
     posicio = 1;
    }
  else if (x <= -5)
    {
      posicio = 2;
    }

 break;

//Posicio 2
case 2:
 motor = 100;
 direccio = 0;

  if( x > -15 )
    {
     posicio = 2;
    }
  else if (x <= -15)
    {
     posicio = 3;
    }

 break;

//Posició 3
 case 3:

 motor = 100;
 direccio = -30;
```

```
 if( x > -20 )
   {
    posicio = 3;
   }
 else if (x <= -20)
   {
     posicio = 4;
   }

break;

//Posició 4
case 4:

motor = 225;
direccio = 30;

  if( x < -10 )
     {
      posicio = 4;
     }
  else if (x >= -10)
     {
       posicio = 5;
     }

break;

//Posició 5
case 5:

motor = 100;
direccio = -30;

   if( x > -18 )
     {
      posicio = 5;
```

```
            }
          else if (x <= -18)
            {
              posicio = 6;
            }

        break;

         //Posició 6
        case 6:
        motor = 225;
        direccio = 30;

         if( x < -10 )
           {
            posicio = 6;
           }
         else if (x >= -10)
           {
             posicio = 7;
           }

        break;

          //Posició 7 - Estat de repòs -> Cotxe aparcat
        case 7:
        motor = 175;
        direccio = 0;

        break;


   } // Fi maniobra d'aparcament

 }// Fi Auto

else if (Auto == 2) //Si Auto és 2 inicia l'aparcament en bateria
```

```
{
motor=175;
direccio=0;
switch(posicio)
  {
     //Estat inicial de repòs.
     case 0:
      direccio = -30;
      posicio = 1;
     break;

      case 1:
       motor = 225; //Endavant
       direccio = -30;

        if( x < 20 )
        {
         posicio = 1;
        }
        else if (x >= 20)
        {
         posicio=2;
        }
       break;

      //Posició 1
      case 2:
       motor = 100;
       direccio = 30;
        if( y < -10 )
        {
          posicio = 2;
        }
        else if (y >= -10)
        {
          posicio = 3;
        }
```

```
  //Posició 2
  break;

   case 3:
     motor = 225;
     direccio = -30;
      if( y > -25 )
        {
          posicio = 2;
        }
      else if (y <= -25)
        {
          posicio = 3;
        }

   break;

  //Posició 3
   case 4:

    motor = 100;
    direccio = 30;
      if( y < -5 )
        {
         posicio = 4;
        }
      else if (y >= -5)
        {
          posicio = 5;
        }

   break;

//Posició 4
 case 5:
```

```
          motor = 100;
          direccio = 0;
            if( y < 4 )
              {
               posicio = 5;
              }
            else if (y >= 4)
              {
                posicio = 6;
              }

           break;

           //Posició 5 - Estat de repòs -> Cotxe aparcat
           case 6:
           motor = 175;
           direccio = 0;

           break;

}
}

//Escrivim les sortides
   analogWrite(3,pwm_direccio);
   analogWrite(11,motor);

  } // Fi del loop
```

B. PROGRAMA LABVIEW

Al llarg del projecte ja s'ha explicat i ensenyat la interfície del programa del Labview utilitzat per adquirir les dades i s'ha explicat el seu organigrama. Per tant, aquí s'ensenyarà l'editor dels blocs amb les explicacions corresponents.

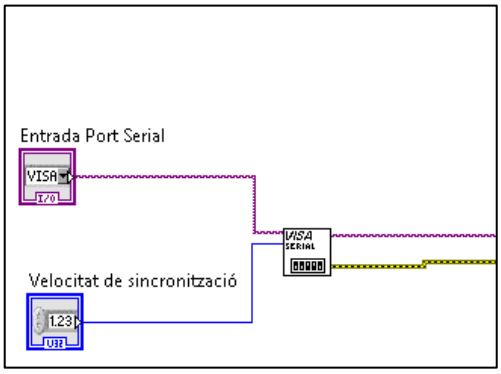


Figura 56. Comunicació Software

A la figura 56 es pot veure la comunicació realitzada entre el Labview i l'Arduino. Després d'haver fet la comunicació vam realitzar dins el *While Loop* una seqüència on hi posaríem les diferents parts del disseny del control.

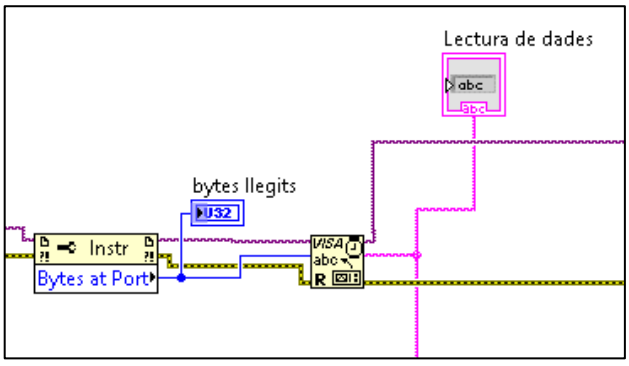


Figura 57. Lectura de les dades

A la figurar anterior es mostra la part on hi llegirem les diferents dades a fi d'efecte de poder guardar resultats, representar-les en un gràfic o bé en diferents indicadors numèrics.

En la figura que tenim seguidament es pot veure com separem les dades que ens arriben de l'Arduino per després poder-les dividir en cada apartat bé si és la consigna, o el valor de la sortida.

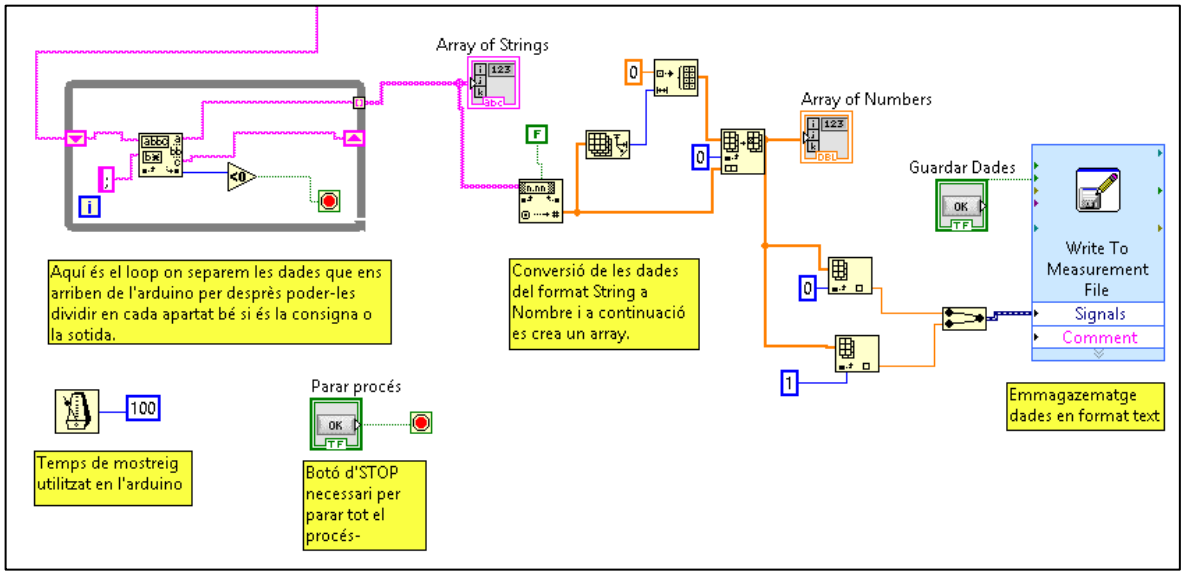


Figura 58. Processament de les dades

Per tractar les dades que llegim primer les hem de seperar les unes de les altres, això ho realitza al loop que podem veure. Cada cop que rep un ";" separa la dada anterior de la següent.

L'Array of strings que podem observar crea una matriu de cadenes. Un cop ha fet això reemplacem un element o subconjunt d'una matriu en el punt que especifiquem amb l'índex i retornem l'element o subconjunt de la matriu a l'índex, després hi tenim els valors de La consigna o la sortida. Finalment, hi ha el bloc per a guardar les dades en un arxiu de text.