UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Politècnica Superior d'Enginyeria de Manresa

TREBALL FINAL DE GRAU

CARTOGRAFIA D'ESPAIS MITJANÇANT UN ROBOT MÒBIL CONSTRUÏT AMB ARDUINO

(Memòria)

Grau en Enginyeria Electrònica Industrial i Automàtica

Curs 2018/2019

Autora: Aina Ibáñez Masanés

Director: Víctor Barcons Xixons

5 de juliol del 2019, Manresa

Especial agraïment a l'espai d'experimentació tecnològica
PensActius pels seus consells i per deixar-me fer ús
de les seves instal·lacions.

RESUM DEL PROJECTE

Aquest projecte consisteix en el disseny i la programació d'un robot mòbil, d'ara en endavant anomenat Carpher, així com un entorn capaç de representar en tres dimensions l'espai on aquest es troba. Pel control a baix nivell (microcontrolador) s'utilitza la plataforma d'Arduino, de software lliure, i per la representació de les mesures dels sensors làser s'utilitza el motor per videojocs Unreal Engine 4 (UE4).

El hardware de Carpher es compon d'un xassís dissenyat en 3D i posteriorment tallat amb làser sobre planxes de metacrilat. Va equipat amb dos motors de corrent contínua complementats amb codificadors i un IMU (unitat de mesura inercial) de 9DOF (9 graus de llibertat) que permetrà, mitjançant el model odomètric, conèixer la posició en (x, y) de Carpher en temps real. Per tal de detallar l'entorn compta amb un total de cinc sensors làser que s'encarreguen de cartografiar l'espai, i un sensor d'ultrasons per tal d'evitar que el robot mòbil xoqui.

El software de Carpher permet a l'usuari controlar-lo mitjançant un controlador tipus PS2 (PlayStation 2) o bé, deixar que el robot treballi de forma autònoma. Les mesures preses pels sensors làser i els càlculs d'odometria són enviats a l'ordinador mitjançant comunicació sèrie remota, i en l'entorn UE4 seran tractades i mostrades en tres dimensions.

PROJECT SUMMARY

This project consists of the design and programming of a mobile robot, hereinafter called Carpher, as well as an environment capable of representing in three dimensions the space where it is located. For low-level control (microcontroller) it's used Arduino platform, which is free software, and by the representation of the measurements of distance the engine for video games Unreal Engine 4 (UE4).

Carpher hardware is made up of a chassis designed in 3D and later laser cut on methacrylate plates. It is equipped with two continuous current motors complemented by Encoders and an IMU (inertial measuring unit) of 9DOF (9 degrees of freedom) that will allow, through the model Odomètric, to know the position in (x, y) of Carpher in real time. To detail the environment has a total of five laser sensors that are responsible for mapping the space, and an ultrasonic sensor to prevent the mobile robot from collide.

The Carpher software allows the user to control it using a PS2-type driver (PlayStation 2) or, let the robot work autonomously. The measurements taken by the laser sensors and odometry calculations are sent to the computer via remote serial communication, and in the UE4 environment they will be treated and displayed in three dimensions.

TAULA DE CONTINGUTS

LLISTA DE TAULES

LLISTA DE FIGURES

INTRODUCCIÓ

Avui dia, els robots formen part elemental de la vida quotidiana. Segons la Real Academia Española (RAE, s.f.a) es defineix robot com "una màquina o enginy electrònic programable capaç de manipular objectes i realitzar diverses operacions". A més, són utilitzats en feines repetitives, difícils i perilloses, i es poden trobar en els àmbits de la fabricació industrial, l'exploració de la Terra i de l'espai, la medicina o la investigació en laboratoris entre molts d'altres.

Un dels camps de la robòtica on hi ha hagut més desenvolupament ha estat en els robots mòbils, aquests es poden trobar tant en ambients domèstics com industrials i la gran majoria es caracteritza per poder moure's de forma autònoma analitzant l'entorn amb l'ajuda de sensors. Carpher forma part d'aquest camp, ja que tenint en compte les seves característiques, encaixa perfectament a la definició de robot mòbil.

En aquest projecte es dissenya i desenvolupa a Carpher, explicant les característiques essencials pel seu funcionament i la programació necessària per moure's de forma autònoma en l'espai. Els càlculs del model odomètric permeten posicionar-lo en x i en y. Aquestes dades, juntament amb les mesures dels sensors de distància, s'envien a un programa capaç de crear una simulació de l'entorn en tres dimensions.

Això permet a l'usuari recórrer, explorar i cartografiar espais de difícil accés, alguns exemples serien forjats sanitaris, mines o coves. També pot tenir un ús domèstic equipant-li un aspirador i, cartografiant la casa, es podria optimitzar el recorregut evitant que el robot aspires dues vegades per la mateixa zona.

CAPÍTOL 1: MARC TEÒRIC I ANTECEDENTS

En aquest capítol s'introdueix al lector al món de la robòtica explicant els seus inicis i la seva evolució, fent un especial èmfasi en els robots mòbil. També es fa una anàlisi de les característiques mecàniques que poden adoptar i els avantatges i desavantatges de cada una d'elles. També s'expliquen els sensors que ha de tenir el mòbil per tal de poder navegar de forma autònoma per l'espai, així com es farà una introducció a l'odometria, model què permet conèixer la posició del robot. Per últim es donen alguns exemples de projectes de robots mòbils per tal de motivar i enriquir la imaginació del lector.

1.1 Història i importància de la robòtica

La robòtica és una ciència o branca de la tecnologia, que estudia, dissenya i construeix màquines capaces de realitzar tasques que requereixen l'ús d'intel·ligència.

La paraula robot té el seu origen en Rossum's Universal Robots o R.U.R (Figura 1), una obra literària teatral de ciència-ficció escrita pel txec Karel Capek l'any 1920. La paraula sorgí a partir del mot txec *robbota*, que significa servitud o treball forçat. Més tard, l'any 1942, l'escriptor i científic Isaac Asimov batejà a la ciència que estudia els robots com a robòtica. També creà les tres lleis de la robòtica, un conjunt de normes aplicades a la majoria dels robots de les seves novel·les, dissenyats per complir les ordes dels humans. Una de les obres més destacades d'Asimov es I, Robot (Figura 2), una recopilació de nou històries curtes de ciència-ficció publicades entre 1940 i 1950.

Figura 1. Fotografia de la representació de l'obra R.U.R. l'any 1922.

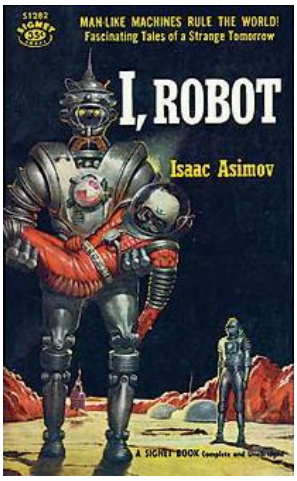


Figura 2. Portada del llibre I, Robot d'Isaac Asimov.

En el món de la ciència-ficció, l'home s'ha imaginat als robots principalment amb forma humanoide, capaços de visitar nous mons, prendre amb el poder dels humans o simplement substituir-los en les feines més quotidianes.

La història de la robòtica moderna va tenir el seu punt de partida l'any 1961 quan George C. Devol Jr. va presentar el model de prova de l'"Unimate" (Figura 3), el primer robot industrial que es va instal·lar en una cadena de muntatge de General Motors. Aquest tenia la forma d'un braç robòtic i permetia transportar i soldar petites peces.

Figura 3. Presentació al 1961 del primer robot industrial anomenat "Unimate".

Des d'aquell moment la robòtica s'ha desenvolupat sense fre i en pocs anys s'han aconseguit metes que a principis de segle XX semblaven ciència-ficció. Actualment es pot trobar en àmbits com la medicina, la indústria, l'educació, l'agricultura, l'exploració especial o la vida domèstica, entre d'altres.

Els robots són capaços de realitzar tasques impossibles per a l'ésser humà, com seria l'exploració continuada de Mart mitjançant el robot *Curiosity* de la N.A.S.A, o bé facilitar tasques repetitives o perilloses com serien tapar milers d'ampolles de plàstic o carregar grans pesos.

1.2 Robòtica mòbil

La robòtica mòbil és un dels camps de la robòtica on la característica principal dels robots és la seva capacitat per desplaçar-se de forma autònoma. Les seves aplicacions són molt variades i usualment estan relacionades amb tasques de risc o nocives per la salut humana en àrees com l'agricultura, el transport de càrregues perilloses, tasques de manteniment en reactors nuclears, l'exploració subterrània o extraterrestre, entre d'altres.

És necessari aclarir que el concepte d'autonomia es refereix a la capacitat de percebre, modelar, planificar i actuar per aconseguir determinats objectius, sense la intervenció (o una intervenció molt petita) de l'operador humà. Per tant, un robot mòbil ha de ser capaç d'assolir certs objectius amb una intervenció mínima o nul·la de l'ésser humà. Per altra

banda, també es troba el concepte de vehicle autoguiat, que es limita a seguir camins preestablerts, com seria línies pintades a terra o bandes reflectores.

Les aplicacions dels robots mòbils van des d'usos domèstics (Figura 4) i de serveis fins a agro-industrials, sense oblidar aplicacions de vigilància (Figura 5) i militars (Figura 6), sectors que més inverteixen en el seu desenvolupament.

Figura 4. Robot domèstic TOYOTA HOME ASSISTANT ROBOT.

Figura 5. Robot de vigilància SUMMIT-XL.

Figura 6. Robot militar TALON pel desarmament d'explosius.

1.3 Morfologia dels robots mòbils

Els robots mòbils tenen unes característiques determinades segons la tasca o tasques que hagin de realitzar. En una primera fase de disseny es decideix el tipus de rodes, el sistema de tracció i direcció i la forma física del robot. En la segona fase es determinen les característiques sensorials.

En el disseny es tenen en compte característiques com la velocitat, la manejabilitat i les particularitats del terreny. La precisió i la rapidesa amb les que el robot és capaç d'aconseguir el seu objectiu, vénen definides per un sistema de tracció confiable i un sistema de direcció que doni manejabilitat al robot. També s'ha de tenir en compte el

nombre de rodes necessàries i el tipus, així com la disposició d'aquestes per tal d'aconseguir una estructura mecànica estable.

1.3.1 Tipus de sistemes de locomoció

El sistema de locomoció és una característica que va relacionada amb l'entorn pel qual es mourà el robot. Es poden diferenciar diversos tipus:

- Terrestres
    - Hexàpode o amb potes (Figura 7)
    - Amb rodes (Figura 8)
    - Amb cadenes o eruga (Figura 9)
    - Humanoide (Figura 10)
- Aquàtics o unmanned underwater vehicles (UUV)
    - Flotants (Figura 11)
    - Submarins (Figura 12)
- Aeris o unmanned aerial vehicles (UUV) (Figura 13)

Figura 7. Robot amb potes SPOT.

Figura 8. Robot amb rodes iRobot Verro 300.

Figura 9. Robot amb cadenes TALON.

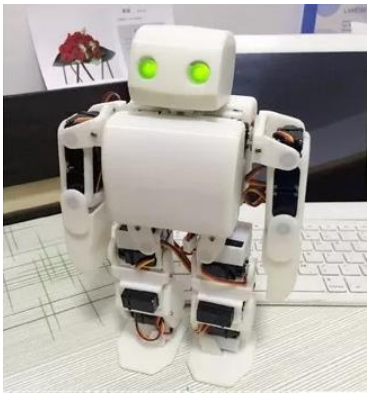


Figura 10. Robot humanoide de SISTEMSOFT.

Figura 11. Robot aquàtic flotant WAVE GLIDER.

Figura 12. Robot submarí RangerBot.

Figura 13. Robot aeri WING.

1.3.2 Tipus i disposició de les rodes

En el mercat actual existeixen diferents tipus de rodes pels robots mòbils. Se'n diferencien quatres tipus:

- **Roda fixa** (Figura 14): el seu moviment és únicament entorn del seu eix central i aquest està fixat a la plataforma del robot.
- **Roda orientable centrada** (Figura 15): a part de girar sobre el seu eix central també ho pot fer respecte a l'eix vertical, aquest travessa la plataforma i el centre de la roda.
- **Roda orientable descentrada o roda boja** (Figura 16): és similar a la centrada però amb la diferència que l'eix vertical no passa pel centre de la roda.
- **Roda sueca o omnidireccional** (Figura 17): és igual que una roda fixa amb la diferència què té uns petits rodets que li permeten moure en un determinat angle.

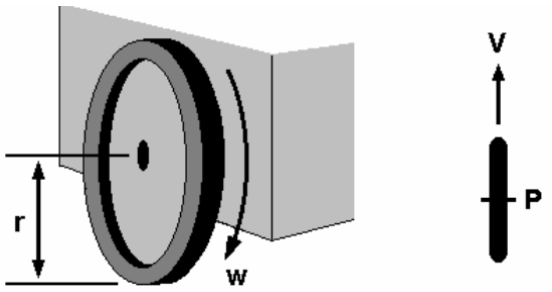


Figura 15. Moviments d'una roda fixa.

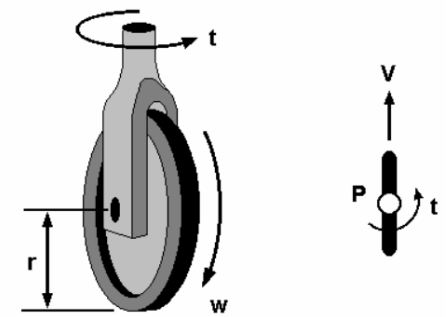


Figura 14. Moviments d'una roda orientable centrada

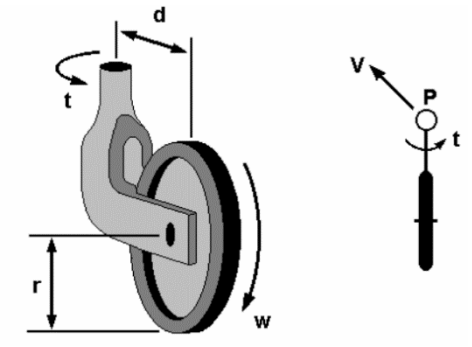


Figura 16. Moviments d'una roda boja.

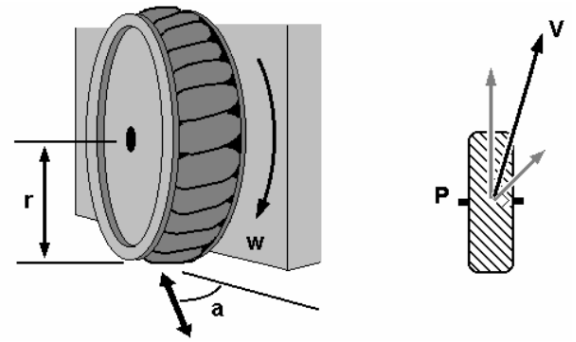


Figura 17. Moviments d'una roda sueca.

El nombre de rodes i la combinació de la disposició d'aquestes, fa que hi hagi una gran varietat de robots mòbils diferenciats pel seu grau de maniobrabilitat. A continuació, es presenten les principals característiques de disseny de diversos tipus de robots mòbils.

1.3.2.1 Robot omnidireccional

Aquests robots tenen una maniobrabilitat màxima en el pla, ja que poden desplaçar-se en qualsevol direcció sense necessitat de reorientar-se. Per contra, són difícils d'implementar. A la Figura 18 és mostra el robot URANUS desenvolupat per la Universitat de Michigan i en la Figura 19 el conjunt de moviments que aquest pot realitzar. D'acord a la rotació de cada una de les rodes, el robot pot avançar, girar o desplaçar-se cap a qualsevol direcció sense necessitat de reorientar-se.

Figura 18. Robot omnidireccional URANUS.

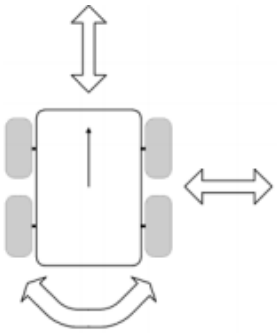


Figura 19. Maniobrabilitat d'un robot omnidireccional.

1.3.2.2 Robot diferencial

Aquesta és la configuració que aporta la cinemàtica més senzilla, es tracta d'una estructura que consta de dues rodes fixes convencionals sobre el mateix eix, controlades de manera independent i una roda boja que li aporta estabilitat. El canvi de direcció es realitza modificant la velocitat relativa de les rodes dreta i esquerra. És un sistema barat i fàcil d'implementar però per contra requereix un control de precisió (per exemple un PID) per tal de mantenir trajectòries rectes. A la Figura 20 és mostra 20 el conjunt de moviments que aquest pot realitzar i en la Figura 21 el robot PIONEER P3-DX desenvolupat per Adept.

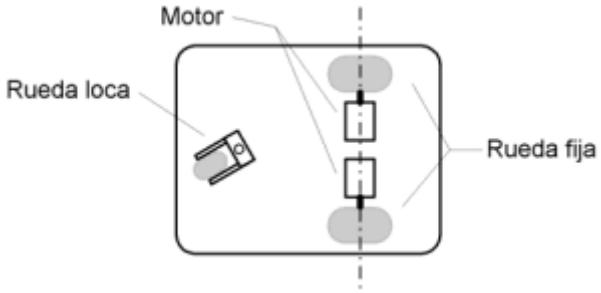


Figura 21. Maniobrabilitat d'un robot diferencial.

Figura 21. Robot diferencial PIONEER P3-DX

1.3.2.3 Robot síncron

Normalment format per tres rodes orientables centrades i acoblades als vèrtexs d'un triangle equilàter sota unes plataformes cilíndriques (Figura 22 i 23). En aquest cas el robot pot canviar la direcció del seu moviment simplement canviant l'orientació de les rodes. Aquest moviment síncron es pot aconseguir mitjançant mètodes mecànics com sistemes de tracció i direcció per corretges o per mitjans electrònics com serien senyals d'accionament simultanis per cada un dels motors.

Com a avantatge, el control en línia recta està garantit de forma mecànica i, en tenir els motors separats per translació i rotació, se simplifica molt. Per contra, la seva implementació i disseny són complicats.

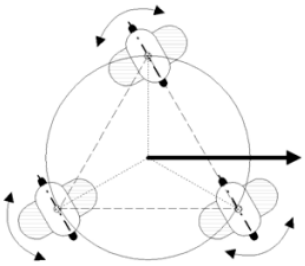


Figura 23. Disposició rodes en un robot síncron.

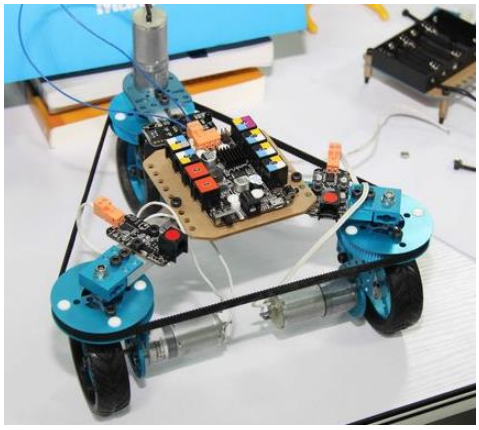


Figura 22. Robot síncron creat amb Makeblock.

1.3.2.4 Tricicle

El robot tipus tricicle està format per dues rodes convencionals fixes sobre un mateix eix i una roda convencional centrada orientable que concentra les funcions de direcció i tracció (Figura 24 i 25). Aquesta configuració aporta una estructura mecànica i un control electrònic senzill que permet el transport de càrregues pesades a baixa velocitat. Per contra, poden sorgir problemes d'estabilitat en terrenys difícils provocant pèrdues de tracció o fins i tot el bolcat.

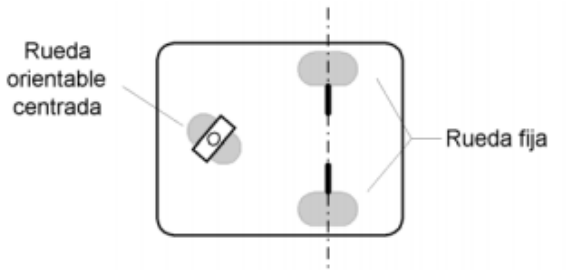


Figura 24. Disposició de les rodes en un robot tricicle.

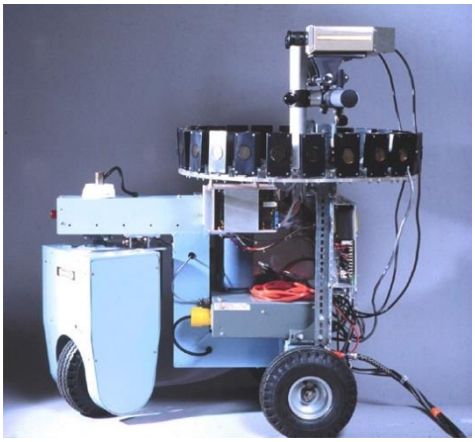


Figura 25. Robot amb locomoció tricicle NEPTUNE

1.3.2.5 Ackerman

El sistema de direcció Ackerman aporta una solució al problema de pèrdua de tracció del tricicle. Com s'observa a la Figura 26, els eixos de les rodes frontals s'intercepten en un punt C que pertany a l'eix comú de les rodes posteriors. Aquesta estructura, a part de brindar més estabilitat, evita lliscaments i per tant redueix els errors de control de posició del mòbil. A la Figura 27 es pot veure un robot tot terreny que utilitza aquest sistema de locomoció.

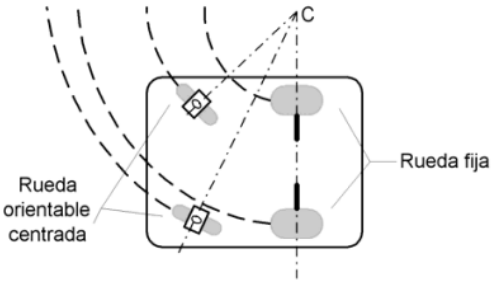


Figura 26. Disposició de les rodes en un robot amb sistema Ackerman.

Figura 27. Vehicle teledirigit amb sistema Ackerman.

1.4 Sensors per robots mòbils

El sistema de percepció d'un robot mòbil permet que s'adapti i faci front a les situacions canviants de l'entorn, així com reaccionar a possibles esdeveniments inesperats. Això exigeix la utilització d'un sistema sensorial que subministri constantment la informació que proporciona l'entorn.

Leonard i Durrant-whyte van resumir el problema de la navegació d'un robot mòbil en tres preguntes: On estic? On vaig? I com haig d'arribar allà? Amb les respostes a les anteriors preguntes, el robot ha de ser capaç de realitzar tres tasques fonamentals: estimar la seva posició i orientació, mantenir-se actualitzat de l'entorn i detectar possibles obstacles.

Per tal que el robot conegui el seu entorn de treball i hi pugui adaptar el seu funcionament, és necessària una recol·lecció i un processament d'informació de diferent

tipus per, posteriorment, utilitzar-ho en el sistema de control. Aquesta informació és generada pels sensors que, segons la necessitat particular del treball a realitzar, poden variar en número, tipus i complexitat. La qualitat i quantitat d'informació obtinguda permetran controlar en el robot trajectòries estables i sense oscil·lacions, el que assegura una arribada al punt de destí amb el mínim error i sense patir cap xoc durant el trajecte.

1.5 Posicionament d'un robot mòbil

És possible calcular la posició i orientació d'un robot mòbil a partir d'una sèrie de mesures internes com les voltes donades per les rodes, les velocitats i les acceleracions en els eixos, els canvis de direcció i sentit, etc. Els sensors integrats en el vehicle són els encarregats de proporcionar aquesta informació, i en funció de la informació utilitzada es poden diferenciar en dos grups. El primer és el sistema odomètric, el qual utilitza la informació donada per codificadors òptics o sensors Doppler. Mentre que el segon utilitza la informació de giroscopis i acceleròmetres.

1.5.1 Model odomètric diferencial

L'odometria té com a objectiu estimar la posició i orientació d'un vehicle a partir del nombre de voltes donades per les seves rodes. La seva idea fonamental és la integració temporal del moviment, el qual porta inevitablement a l'acumulació d'errors. L'avantatge és que, a un baix cost i de forma senzilla, es pot obtenir una freqüència de mostreig molt alta. Tot i això, a més de necessitar un calibratge pel desgast de les rodes, aquesta tècnica és vulnerable al lliscament, les irregularitats del terreny i les variacions en la càrrega transportada.

Per obtenir el punt (x, y) del robot s'han seguit una sèrie de passos descrits a continuació. Es parteix d'obtenir tk (temps en què s'ha pres la mostra), P_d (polos per segon del codificador dret), P_e (polos per segon del codificador esquerre), L (distància del centre de l'eix de les rodes respecte d'una d'aquestes) i r (radi de les rodes).

Obtenir les mesures L i r exactes és complicat, ja que la distància efectiva entre les dues llantes s'ha de calcular des del punt mitjà en què aquestes tenen contacte amb el terra (Figura 28) i, per altra banda, r sempre és més gran que la real perquè no es té en compte la deformació que pateixen les rodes en suportar el pes del robot (Figura 29).

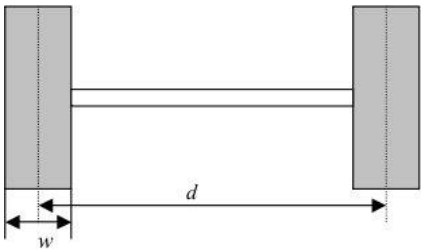


Figura 28. Incertesa en el punt de contacte entre les rodes i el terra.

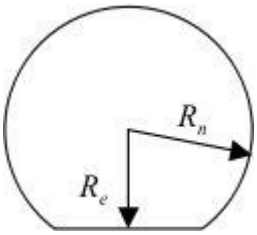


Figura 29. Disminució del radi de la roda provocada pel pes que aquesta suporta

Els càlculs necessaris per obtenir (x, y) del robot mòbil són:

1r.- Càlcul del diferencial de temps, equival al temps entre mostra i mostra:

$$T_s = t_{k+1} - t_k$$

Consisteix a obtenir el diferencial de temps de l'adquisició per tal de tenir una mesura relativa que servirà per realitzar la resta de càlculs.

2n.- Expressar els polsos/segon de les rodes en graus

$$\dot{\theta}_{d\_g} = \frac{P_d \cdot 360}{N_c} \qquad \dot{\theta}_{e\_g} = \frac{P_e \cdot 360}{N_c}$$

3r.- Passar de graus a radians, obtindrem

$$\dot{\theta}_d = \dot{\theta}_{d\_g} \cdot \frac{\pi}{180} \cdot \frac{1}{T_s} \qquad \dot{\theta}_e = \dot{\theta}_{e\_g} \cdot \frac{\pi}{180} \cdot \frac{1}{T_s}$$

Pas necessari per poder establir les fórmules cinemàtiques i obtenir les velocitats lineals i angulars.

4rt.- Càlcul de la velocitat lineal i angular del robot:

$$v = r\left(\frac{\dot{\theta}_d + \dot{\theta}_e}{2}\right) \qquad w = r\left(\frac{\dot{\theta}_d - \dot{\theta}_e}{2L}\right)$$

A partir de les velocitats lineals, angulars i el diferencial de temps de mostreig es poden representar les mesures adquirides pels codificadors. Mitjançant un pla vectorial que tindrà com a eixos X i Y. Per aconseguir-ho s'emprarà la teoria del sinus i el cosinus.

5è.- Càlcul de (x, y) en l'espai, càlcul de la velocitat angular:

$$x_{k+1} = x_k + v_k T_s \cos(\theta_k)$$

$$y_{k+1} = y_k + v_k T_s \sin(\theta_k)$$

$$\theta_{k+1} = \theta_k + T_s w_k$$

A la Figura 30 es pot veure una imatge on hi ha una descripció visual dels resultats calculats amb l'odometria:

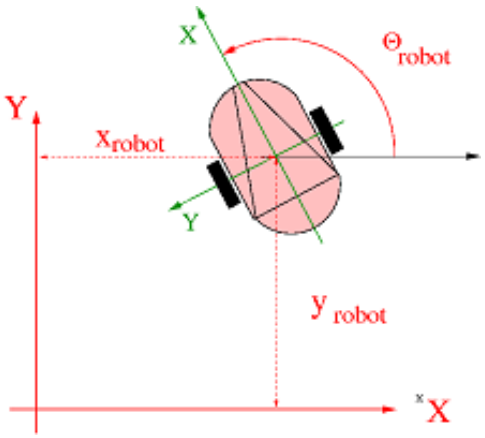


Figura 30. Representació de la posició d'un robot mòbil utilitzant l'odometria.

1.5.2 Navegació inercial

Els IMU (*inertial measurement unit*), en català unitats de mesurament inercial, estimen la posició i orientació del mòbil mitjançant mesures d'acceleració i angles de rotació. Per entendre les dades proporcionades pels IMUs s'ha de parlar de sistemes de coordenades, velocitats angulars i altres conceptes interessants.

Sistemes de referència i coordenades

Per localitzar un punt en un sistema de coordenades, el primer pas és definir un sistema de referència. El més habitual és el que es pot veure a la Figura 31, el sistema cartesià. Per definir un punt en un espai de tres dimensions són necessaris tres valors que definiran les tres coordenades d'aquest punt, s'anomenaran x, y i z.

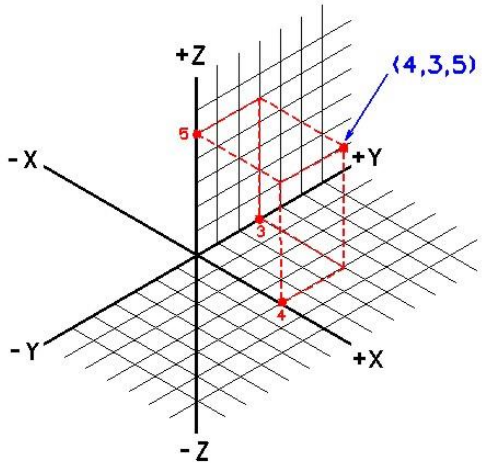


Figura 31. Punt referenciat en un espai x,y i z.

Per exemple, si es posa un avió militar en el sistema de coordenades anterior, es pot fixar el seu centre de gravetat a la vegada que se sap si està ascendent, descendent o realitzant una maniobra de gir sobre si mateix. Per obtenir aquesta informació necessitem els angles marcats per cada un dels eixos de l'aparell respecte a la nostra referència. Aquests angles de desviació s'anomenen en anglès Yaw, Pitch i Roll, i en català corresponen a guinyada, capcineig, i balanceig. A la Figura 32 es poden veure cada un d'aquests angles respecte a un avió.

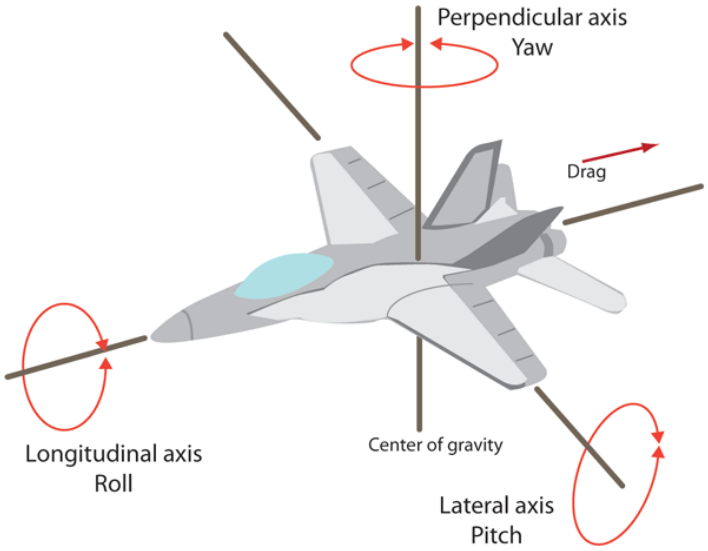


Figura 32. Angles de desviació referenciats en un avió.

En resum, per definir la posició d'un aparell es necessiten tres coordenades que fixin la posició i tres angles que defineixin l'orientació d'aquest respecte els eixos del sistema de referència. Per obtenir aquesta informació s'utilitzen els acceleròmetres, els giroscopis i els magnetòmetres. El seu funcionament es basa en la primera llei de Newton o la llei d'inèrcia, que diu que tot objecte en moviment tendeix a mantenir el seu estat (inèrcia) fins que una força externa el pertorba.

Acceleròmetre

A un sistema de referència que no està sotmès a cap força externa se l'anomena sistema de referència inercial. Si apliquem una força externa al nostre sistema de referència, aquest deixa de ser inercial i es percep una força oposada a les aplicades. Per detectar aquest tipus de força s'utilitzarà un acceleròmetre.

Un acceleròmetre és un dispositiu de massa m que es vincula al robot de massa M a través d'una molla de constant elàstica k. Quan el robot es mogui es complirà quant:

$$F = m \cdot a = k \cdot x$$

On, a => acceleració del mòbil

x => deformació provocada en la molla per la força F.

Sabent el valor de F es poden obtenir els valors de l'acceleració a, la velocitat u i la longitud de la trajectòria curvilínia s.

$$a = \frac{k}{m} \cdot x \rightarrow u = \int a \cdot dt \rightarrow s = \int u \cdot dt$$

A la Figura 33 es pot veure el principi de funcionament de l'acceleròmetre.

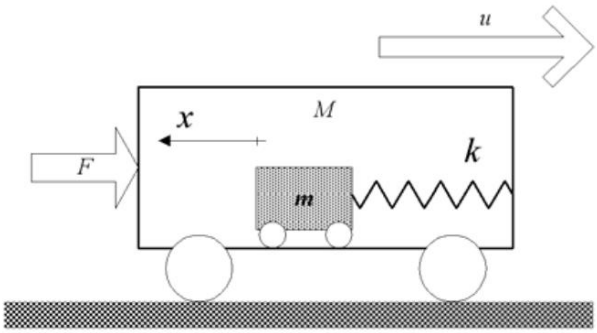


Figura 33. Principi de funcionament de l'acceleròmetre.

Els càlculs explicats anteriorment s'apliquen quan la força actua perpendicular a un eix, però la realitat és més complexa ja que es barreja la informació dels tres eixos per determinar, mitjançant càlculs, la direcció de la força que se li aplica.

Giroscopi

Els giroscopis detecten la força centrifuga que provoca el gir al voltant dels tres eixos de gir principals i les converteix en velocitat (°/seg). D'aquesta forma, i mitjançant càlculs, s'obtenen els angles d'inclinació respecte als eixos de referència.

Magnetòmetre

Els magnetòmetres detecten el magnetisme terrestre en tres eixos (x, y, z). Sense aquests no és podria obtenir el valor de guinyada, ja que la gravetat sempre actua perpendicularment donant un valor constant indecent al moviment.

Els sensors d'orientació són de particular importància en el posicionament de robots mòbils ja què poden ajudar a compensar els errors d'orientació transitoris del sistema odomètric, els quals provoquen un error que va creixent en el temps. Gràcies als IMU l'error es pot detectar i corregir ràpidament.

1.6 Projectes relacionats

Per Internet és poden trobar molts projectes de robots mòbils, ja siguin desenvolupats per grans empreses o de forma individual com a hobby. A continuació s'expliquen molt per sobre dos projectes semblants al que és desenvoluparà en aquest projecte.

1.6.1 ROVINA Project

ROVINA és un projecte nascut de la necessitat de preservar el patrimoni cultural dels espais arqueològics on els investigadors no poden accedir directament. El robot està preparat per moure's en entorns accidentats i per adquirir i digitalitzar dades d'aquest per tal d'aconseguir models texturitzants en tres dimensions. Incorpora una interfície intuïtiva i fàcil d'utilitzar orientada a persones sense coneixements previs de robòtica, que permet supervisar i interactuar amb el robot de forma remota.

A la Figura 34 es pot veure una imatge feta en les primeres proves del robot a les catacumbes de Roma. La Figura 35 mostra el prototip inicial en 3D dissenyat l'any 2016.

Figura 35. ROVINA explorant les catacumbes de Roma.

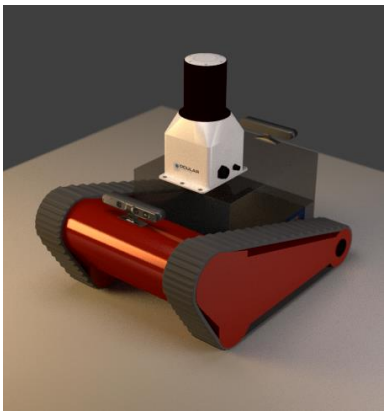


Figura 35. Primer prototip en 3D de ROVINA.

1.6.2 Room mapping Arduino robot with Unity 3D

Aquest és un projecte desenvolupat per l'usuari matthewh8 de la comunitat de Instructables (pagina web en la qual la gent pot penjar els seus projectes maker). És tracta de un robot programat amb Arduino capaç de comunicar-se a través del ESP-8299 NodeMCU amb un telèfon mòbil. El telèfon té instal·lat una aplicació prèviament creada amb Unity 3D, un motor de videojocs. L'aplicació és pot visualitzar simultàniament a un ordinador connectat a la mateixa xarxa Wi-Fi. Permet fer tres coses diferents:

1.- Permet a l'usuari veure des de l'ordinador el què la càmera del telèfon mòbil, ubicada sobre el robot, està gravant en temps real. També a través de les fletxes del teclat pot conduir el mòbil.

2.- Permet que el robot segueix objectes que prèviament se li han ensenyat. És poa l'objecte davant de la càmera i és clica aquest en la pantalla, automàticament el mòbil començarà a seguir-lo.

3r.- Igual què el primer permet conduir el robot amb el teclat. A més, utilitza la realitat augmentada SDK (*Software Development Kit*) per trobar les parets, després envia la informació a l'ordinador que mostra a l'usuari una representació digital del entorn. A la Figura 36 a l'esquerra es pot veure la pantalla de l'usuari al ordinador i a la dreta el robot amb el mòbil detectant el terra.

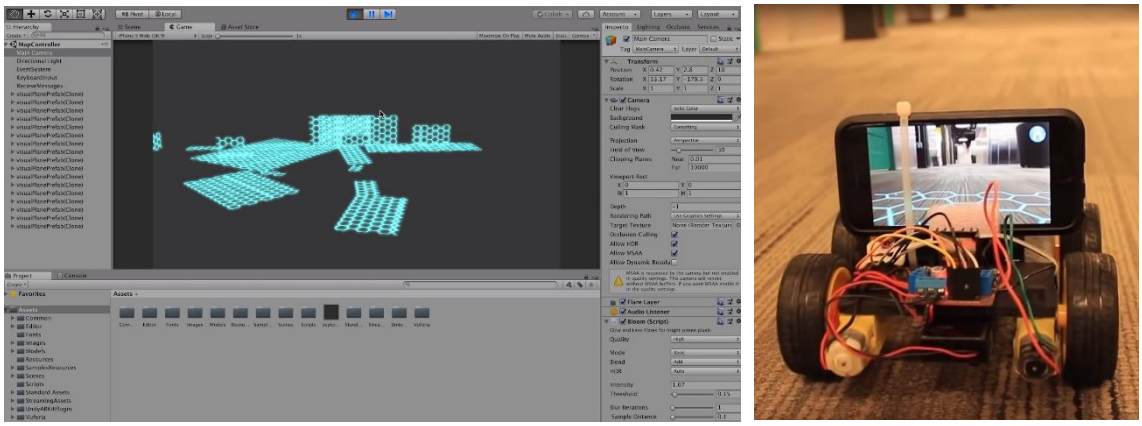


Figura 36. Room mapping Arduino robot with Unity 3D. A l'esquerra la interfície d'usuari, a la dreta el robot.

CAPÍTOL 2: DESENVOLUPAMENT DEL ROBOT

En aquest capítol es descriu el hardware utilitzat per a la construcció de Carpher. Es parla del sistema Arduino, els sensors, els controladors, els mòduls i els perifèrics, així com dels seus fabricants, les llibreries de funcionament, la forma de connectar-los a la placa, la seva programació i qualsevol altre detall important per l'enteniment del sistema complet.

L'objectiu del robot mòbil és poder desplaçar-se en l'espai a la vegada que detecta els objectes que es troben a l'entorn.

2.1 Arquitectura del sistema

Tot robot programable, i per tant Carpher, necessita un microcontrolador (MCU), un petit ordinador especialitzat a controlar circuits electrònics què es troba en un sol circuit integrat. Aquest inclou una CPU (*central processing unit*), una petita memòria volàtil RAM (*random access memory*), unitats d'entrada i sortida on es connectaran els sensors i actuadors i sol portar una memòria ROM (*read only memory*) on és guarda el programa.

Carpher s'equipa amb dos motors per tal de moure's en l'espai, i com el microcontrolador anteriorment mencionat no disposa de suficient potència per fer-los funcionar, es recorre a un mòdul capaç d'alimentar-los.

Per saber la posició de Carpher en l'espai es necessita saber quina distància recórrer en cada moment. Per obtenir aquestes dades s'acobla a cada motor un codificador que indica la velocitat del motor i, mitjançant determinats càlculs, es pot conèixer quina distància s'ha desplaçat i en quin temps. Per aconseguir un control absolut s'utilitza un giroscopi, que permet saber quan gira el robot i en quina direcció, a la vegada que ajuda a detectar si es troba en un terreny amb desnivell.

Amb l'objectiu de detectar els objectes de l'entorn on es troba Carpher, s'utilitzen sis sensors de distància. Les dades resultants s'utilitzen posteriorment per a realitzar el mapa virtual de l'entorn del mòbil.

Per passar les dades del MCU a l'ordinador es fa mitjançant comunicació sèrie, port USB, i s'utilitzen dos mòduls per tal que aquesta es pugui fer sense cables i Carpher guanyi autonomia. Un dels mòduls va connectat amb l'ordinador i l'altre amb el microcontrolador, d'aquesta manera quant l'ordinador vol enviar informació al microcontrolador li diu primer al mòdul que té connectat, i aquest, mitjançant connexió sense fils, li ho envia al mòdul del microcontrolador que al seu torn informa el mateix microcontrolador.

Un dels objectius és que Carpher recorri l'espai de forma autònoma, tot i això, es proporciona a l'usuari un comandament sense cables que permet el control del robot de manera manual en cas de necessitat.

Per tenir més informació de l'entorn, Carpher s'equipa amb un sensor capaç de mesurar la temperatura i la humitat. Si es desitja, també es podrien equipar altres sensors com el de pressió, acústics, de llum... que proporcionarien encara més informació de l'entorn a l'usuari.

Tenint en compte els anteriors elements s'ha creat un diagrama de blocs, Figura 37, amb totes les parts indispensables que ha de tenir el sistema i les comunicacions entre elles per tal de complir els objectius marcats.

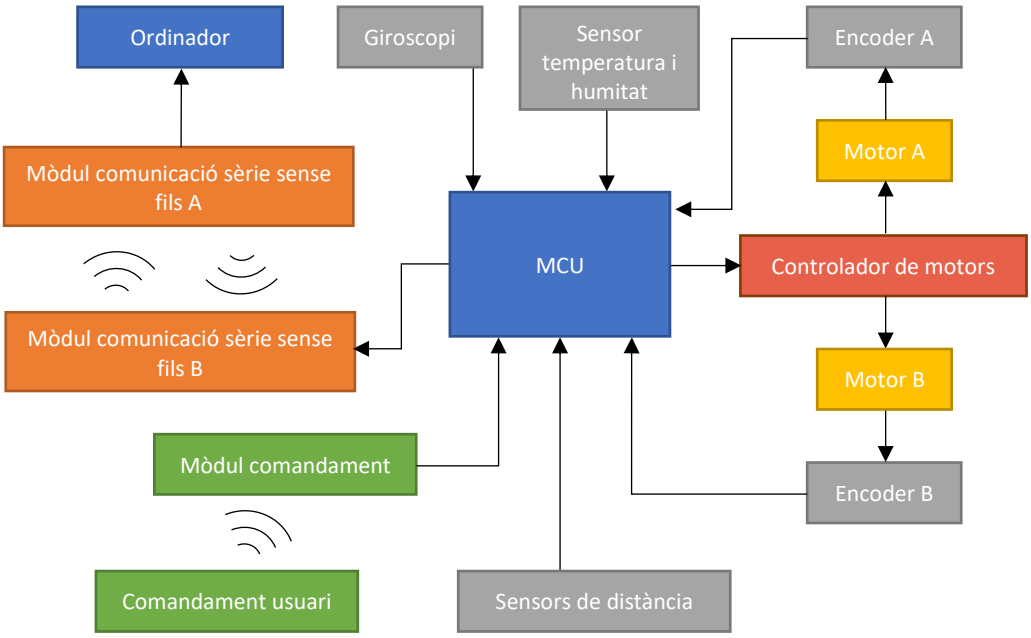


Figura 37. Diagrama de blocs de sistema.

2.2 Part mecànica

En aquesta part s'expliquen les parts mecàniques que té el robot, és a dir, el mecanisme que utilitza per a desplaçar-se.

2.2.1 Sistema de tracció

En desenvolupar un projecte de robòtica que inclou un dispositiu mòbil, una de les primeres decisions és escollir el tipus d'actuadors a utilitzar. S'entén per actuador un dispositiu capaç de generar moviment i que pot ser controlat des d'un processador o un autòmat.

En el mercat es poden trobar molt tipus de motors, a continuació s'exposen els més utilitzats així com els seus avantatges i desavantatges.

Motors de corrent continua o de CC – El seu funcionament és base en l'alineament de dos camps magnètics. Tenen altes velocitats de gir i un parell baix. El control de la posició i la velocitat és dolent. El seu comportament és no lineal i depèn de la càrrega que porta.

Motors *geared down* – Iguals que un motor de CC però incorporant un reductor intern. Això augmenta el parell del motor a la vegada que redueix la seva velocitat. Les velocitats de gir habituals són 60, 120, 240 i 480 rpm.

Motors *brushless* – Són una variació dels motors CC, on es recorre a l'electrònica per realitzar la commutació del camp magnètic. Tenen majors velocitats, menor pes i més durabilitat que els motors CC.

Servomotor o servo – El MCU transmet una posició desitjada i el servo, de manera autònoma, es col·loca a la posició. No pot donar una volta completa, normalment són de 180°. Proporcionen un control total de la posició i gir i una alta precisió. La seva velocitat és baixa però proporciona un parell alt.

Servomotor de rotació contínua – Tal com diu el seu nom són capaços de donar una volta completa. En comptes de controlar-lo per la posició es fa per velocitat. No ofereixen

un control precís de la velocitat de gir, ja que la seva resposta es no lineal i s'han de calibrar.

Motors pas a pas - L'eix gira en un angle fixe anomenat "pas", sent habitual 1,8° (200 passos per volta). És indispensable per al seu funcionament estar controlat per un MCU. El control de posició i la velocitat són absolutes. El parell i la velocitat màxima és intermèdia comparada amb els altres motors.

La informació anterior es resumeix i compara de forma visual en la Taula 1.

	Característica		Control (*)	
	Velocitat	Força/Parell	Posició	Velocitat
Motor de CC	▼Baix	▼Baix	▼Dolent	▼Dolent
Motor *geared down*	– Mitjà	▲Alt	▼Dolent	▼Dolent
Motor *brushless*	▲▲Molt alt	▼Baix	▼Dolent	– Mitjà
Servo	▼Baix	▲Alt	▲▲Absolut	▲▲Absolut
Servo rotació contínua	▼Baix	▲Alt	▼Dolent	– Mitjà
Motor pas a pas	– Mitjà	– Mitjà	▲▲Absolut	▲▲Absolut

(*) Amb un encoder tots passen a control absolut de posició i velocitat

Taula 1. Resum de les característiques dels principals motors de mida petita del mercat.
Font: Luis Llamas (2016)

En aquest projecte s'han utilitzat com a actuadors motors de corrent continu, ja que són els més econòmics i fàcils de controlar de forma indefinida. Els controls de posició i velocitat per si sols són dolents, però acoblant a l'eix un encoder i afegint un giroscopi al sistema se supleix el problema de manera simple i efectiva. A la vegada, les característiques de velocitat i força/parell també són baixes, però Carpher no requereix millores en aquests aspectes, ja que no està pensat per suportar grans carreges ni anar a grans velocitats.

Funcionament dels motors de corrent continua

Els motors de corrent continua, es basen en l'alimentació de dos camps magnètics. Com es pot veure en la Figura 38, l'estator, la part fixa del motor, disposa d'un imant permanent que genera un camp magnètic a l'interior del motor.

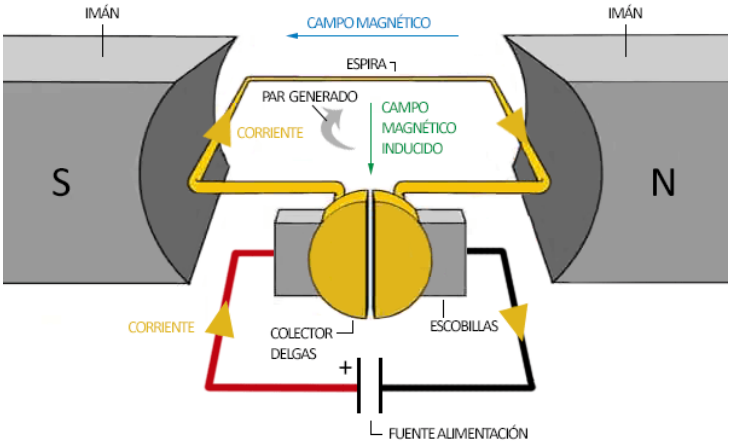


Figura 38. Funcionament intern d'un motor.

En fer circular un corrent elèctric es genera un camp magnètic i el desfasament entre els dos camps magnètics genera un parell de gir, que fa que el rotor giri fins que els dos camps magnètics s'alenin.

Per fer que roti contínuament s'ha d'invertir el corrent que travessa l'espira. Per aquest motiu, els contactes que alimenten l'espira estan compostos per un anell partit que llisca sobre uns contactes elèctrics que freguen amb el mateix anell, transmeten l'electricitat. A l'anell dividit se l'anomena col·lector de delgues, mentre que als contactes lliscants se'ls anomena escombretes. En passar un cert angle, les escombretes passaran d'una delga a la següent, provocant la inversió de corrent a l'espira. D'aquesta forma el col·lector de delgues actua com un inversor mecànic i permet la rotació continua del motor.

En un motor real no es disposa d'una sola espira sinó de bobines formades per múltiples espires. A la Figura 39 es poden veure les parts d'un motor de corrent continua.

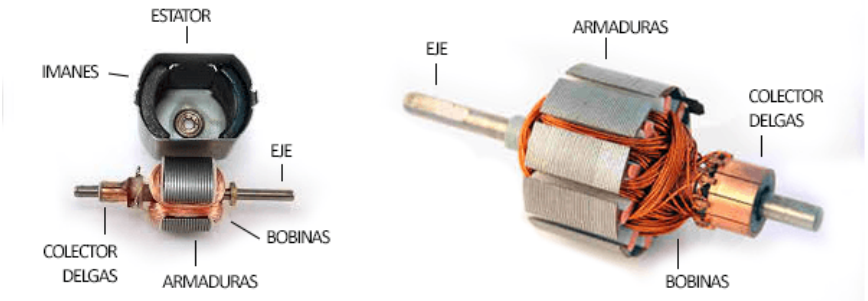


Figura 39. Parts internes d'un motor de corrent contínua.

Les bobines s'enrotllen sobre nuclis ferromagnètics anomenats armadures, aquestes augmenten la potència del motor i disminueix les pèrdues per dissipació magnètica.

L'avantatge d'aquest sistema és que la sincronització és sempre perfecte, independentment de la velocitat i el parell, ja que és el mateix angle de gir del motor el que marca la inversió del corrent. El desavantatge és que el fregament suposa una pèrdua d'eficiència i redueix la vida útil del motor.

Els motors tenen altes velocitats de gir i un parell baix. Es poden utilitzar sortides PWM (*pulse width modulation*) d'un microcontrolador per variar la velocitat, mantenint sempre el parell màxim. Per controlar l'actuador és necessària una etapa d'amplificació, com un transistor bipolar o BJT (*bipolar junction transitor*), o un transistor MOSFET (*metal oxide semiconductor field effect transitor*). Si a més es vol invertir el sentit de gir, es necessitarà un controlador amb pont-H.

El motor que escollit per a la creació de Carpher és el que és mostra en la imatge 40. Com ja s'ha dit, funciona amb corrent continua i té les següents especificacions:

- Voltatge de funcionament: 3-12 V DC (se suggereix utilitzar-los entre 6-8 V).
- Consum de corrent: 70 mA (3 V) (250 mA MAX) .
- Parell màxim: 800 gf cm min.
- Velocitat de rotació: 1:48.
- Velocitat màxima del motor 130 rpm.

Figura 40. Motors CC utilitzats en la construcció de Carpher.

Un cop definit el tipus de motor s'ha de decidir el nombre de rodes, que variarà en funció del seu model:

Model 2WD (*two-wheel-drive*)

- Ús de dos motors encarregats de controlar la tracció i la direcció.
- Requereix una roda boja, exemple a la figura 41, que funciona com un tercer punt de suport per tal de mantenir l'estabilitat. Aquestes tenen la capacitat de girar en totes direccions.

Figura 41. Roda boja.

Model 4WD (four-wheel-drive)

- Ús de quatre rodes convencionals fixes que permeten una major tracció i moviments més sofisticats.
- La seva programació és més complexa que el model 2WD.

Com Carpher no necessita moviments complexos, s'ha optat pel model 2WD, més fàcil de programar que el 4WD. Les rodes que s'utilitzen són de plàstic i amb la part exterior de goma dura. A més, tenen un relleu que permet una millor tracció amb la superfície. Es poden veure en la figura 42.

Figura 42. Rodes de plàstic utilitzades en el projecte.

2.3 Part electrònica

Aquesta part és, sens dubte, la més important en el disseny del projecte. És la base sobre la qual es desenvolupa tot el sistema de mapatge que es vol implementar.

2.3.1 Microcontrolador Arduino

Arduino va néixer l'any 2005 al Interaction Design Institute Ivrea d'Itàlia amb l'objectiu de fer més accessible, simple i econòmic el disseny de circuits electrònics amb microcontroladors per tal que totes les aules de l'escola poguessin practicar l'activitat.

La idea original era fabricar una placa per ús intern de l'escola, però durant el seu desenvolupament, l'escola es va veure obligada a tancar per falta de recursos el mateix any 2005. Davant la perspectiva de perdre tot el procés del projecte, es va decidir fer-lo lliure i obrir-lo al públic perquè tot el món pugues participar en la seva millora i evolució.

Amb els anys, aquest alliberament ha propiciat una gran comunitat que genera diàriament una extensa quantitat de projectes i documentació, cobrint pràcticament qualsevol necessitat. Actualment Arduino és una plataforma lliure, educativa i de desenvolupament que és utilitzada a escala global per a persones de totes les edats.

Generalment, quan es parla d'un Arduino, es fa referència a l'element físic de la plataforma Arduino, el seu *hardware*, també conegut com a placa o PCB (*printed circuit board*).

Arduino disposa d'un entorn de programació integrat conegut com a IDE Arduino (*integrated development environment*), figura 43. Aquest és multiplataforma i es pot instal·lar i executar en sistemes operatius com Windows, Mac OS i Linux.

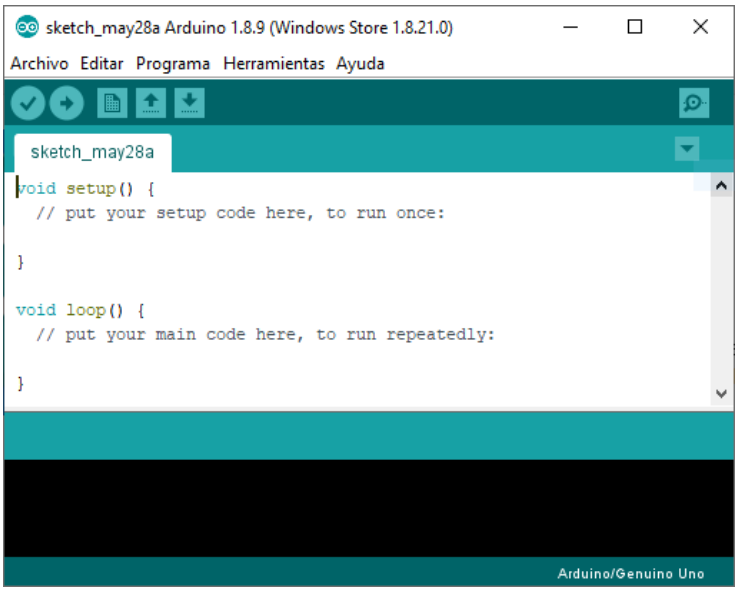


Figura 43. IDE Arduino.

Aquest entorn permet escriure programes, verificar-los i carregar-los a l'Arduino. Disposa d'una eina com és el monitor sèrie (Figura 44) que permet enviar o rebre informació a través del port USB de l'ordinador. Des d'aquest es poden enviar caràcters al microcontrolador, com serià una "A" per encendre un dispositiu LED (*light-emitting diode*), o bé rebre les dades obtingudes per un sensor, tot això programant-lo adequadament.

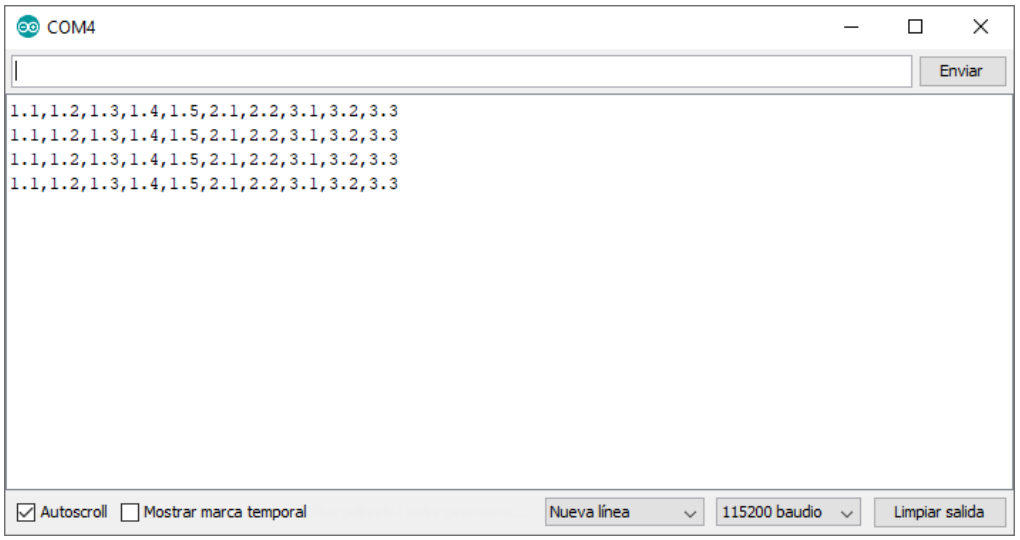


Figura 44. Monitor sèrie IDE Arduino.

El seu llenguatge de programació està basat en el C, de fàcil comprensió i que permet una entrada senzilla als nous programadors a la vegada que permet als programadors experts esprémer tot el seu potencial i adaptar-lo a qualsevol situació.

El preu d'una placa Arduino estàndard oficial és de 20 euros, però en ser *hardware* lliure es poden trobar plaques de fabricants xinesos per menys de 3 euros.

2.3.1.1 Elecció de la placa Arduino

Arduino ha desenvolupat de forma oficial més de deu tipus de plaques. La placa original i més utilitzada és l'anomenada UNO Rev3 (Figura 45), recomanada per iniciar-se en el món Arduino, ja que en ser la primera desenvolupada, és de la que es pot trobar més informació a Internet.

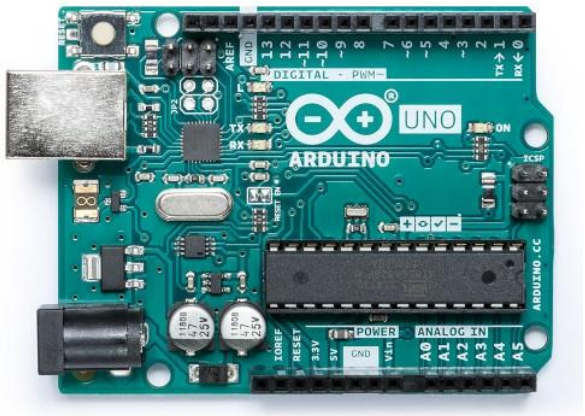


Figura 45. Placa Arduino UNO Rev3.

Arduino UNO Rev3 és una placa electrònica basada en el microprocesador ATmega328P. Té 14 pins d'entrada/sortida digital, dels quals 6 poden utilitzar-se per sortides PWM (*pulse-width modulation*). També compta amb 6 entrades analògiques, una velocitat de rellotge de 16MHz, una connexió USB, un connector d'alimentació i un botó de reinici.

Igual que amb la resta de microcontroladors utilitzats ens les plaques Arduino, l'ATmega328P té una arquitectura desenvolupada per Atmel i, d'alguna mesura,

"competència" d'altres arquitectures com per exemple la PIC del fabricant Microchip. Més concretament, l'Atmega328P pertany a la subfamília de microcontroladors "megaAVR",

Altres famílies de l'arquitectura AVR són les "tinyAVR" (Figura 46), amb els microcontroladors més limitats i que s'identifiquen amb el nom ATiny. I la "XMEGA", amb microcontroladors més capaços i que s'identifiquen amb el nom d'ATxmega (Figura 47).

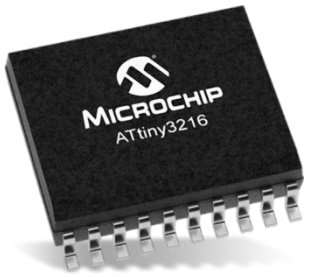


Figura 47. MCU ATiny3216 de Microchip.

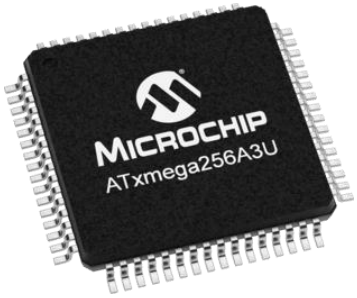


Figura 47. MCU ATxmega256A3U de Microchip.

Tot i que la placa UNO és molt completa per aquest projecte, es necessiten més entrades i sortides digitals que les que aquesta proporciona, de manera que s'utilitza un model superior: la placa Arduino MEGA 2560 Rev3 (Figura 48). Aquesta utilitza el microcontrolador ATmega2560 i està dissenyada per projectes més complexos, proporcionant un nombre més gran d'entrades i sortides, 4 ports sèrie i més memòria de programa.

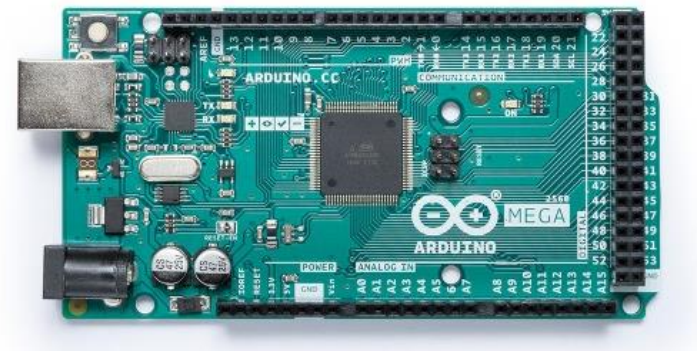


Figura 48. Placa Arduino MEGA 2560.

A la següent taula és fa una comparació entre les especificacions de les dues plaques.

Taula 2. Comparació de les especificacions tècniques entre les plaques UNO i MEGA2560 d'Arduino.

Microcontrolador	ATmega328P	ATmega2560
Voltatge de funcionament	5V	5V
Pins E/S digitals	14 (6 PWM)	54 (15 PWM)
Pins entrada analògiques	6	16
Corrent DC per pin d'E/S	20 mA	20mA
Memòria Flash	32 KB	256 KB
SRAM*	2KB	8 KB
EEPROM**	1KB	4KB
Velocitat rellotge	16 MHz	16 MHz
Dimensions	68.6 x 53.4 mm	101.52 x 53.3 mm

* SRAM (Static Random Access Memory): Memòria que utilitza el MCU per treballar amb les dades temporals durant l'execució del programa, les dades és perdran al apagar la placa.

** EEPROM (Electrically-erasable programable read-only memory): Memòria que permet emmagatzemar dades, per exemple les instruccions del programa, i les quals no és perdran al apagar la placa. Està limitada a 100.000 escriptures.

Tot i que l'Arduino MEGA 2560 és superior en les especificacions electròniques que la UNO, té un gran desavantatge a l'hora d'implementar-la, i és que la seva dimensió, sent casi el doble de gran que l'Arduino UNO, és difícil d'encabir en alguns projectes.

2.3.2 Controlador motors

L'Arduino, i en general tots els autòmats, no disposen de potència suficient per moure actuadors. De fet, la funció d'un processador no ha de ser executar accions sinó fer que altres realitzin aquest treball pesat.

Per aquest motiu s'utilitza un controlador de motors L298N (Figura 49). Aquest permet engegar i controlar dos motors de corrent continua des de l'Arduino, variant tant la direcció com la velocitat de gir.

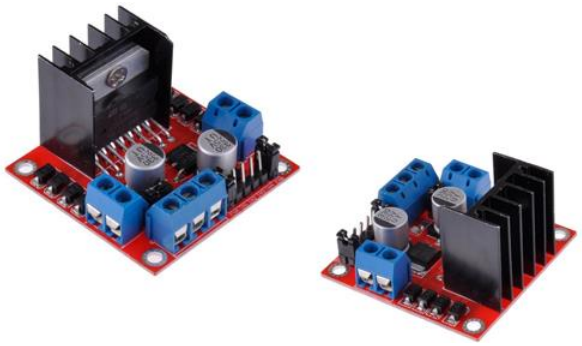


Figura 49. Controlador de motors L298N.

Funcionament d'un L298N

El L298N consisteix en dos ponts-H, un per la sortida A (motor 1) i l'altre per la sortida B (motor 2). El L298N és un component àmpliament utilitzat en electrònica per tal d'alimentar una càrrega de forma que es pugui invertir el sentit del corrent que el recorre.

Actuant sobre els 4 transistors, activant els transistors oposats en diagonal de cada branca, podem variar el sentit del corrent que travessa la carrega, aquest efecte es mostra a la Figura 50. En la imatge de la dreta el motor giraria cap endavant, mentre que a la de l'esquerra ho faria cap endarrere.

Connectant simultàniament els transistors superiors o inferiors, podem posar la carrega Vcc o GND respectivament, configuració que s'utilitza com a fre.

Mai es pot encendre els dos transistors d'una mateixa branca, ja que es produiria un curtcircuit.

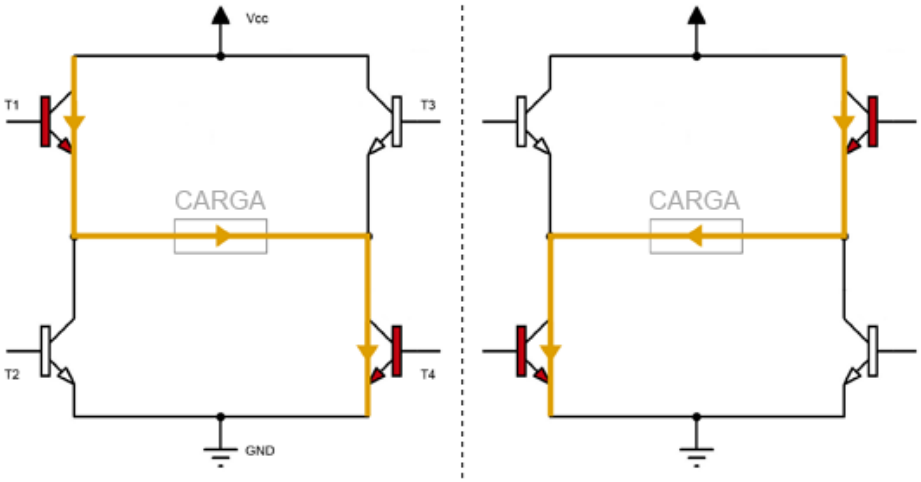


Figura 50. Funcionament pont-H.

Esquema de muntatge

La placa de connexió del L298N incorpora una entrada de voltatge, una sèrie de *jumpers* per configurar el mòdul, dues sortides (A i B), i els pins d'entrada que regulen la velocitat i el sentit de gir. A la Figura 51 es mostra les entrades i sortides del mòdul.

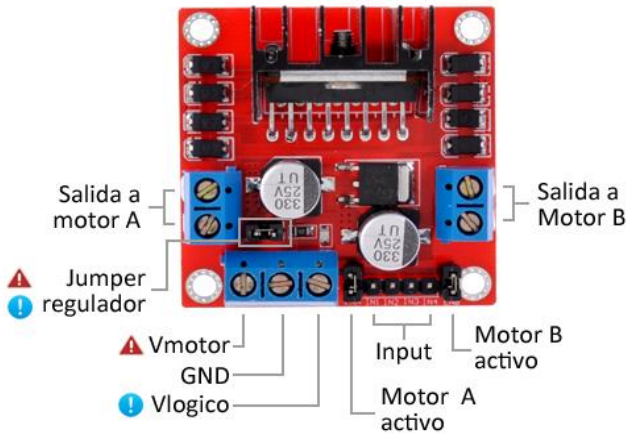


Figura 51. Connexions de la placa del driver L298N.

El voltatge que alimentarà els motors és proporcionat per l'entrada de tensió.

- Si el regulador està actiu (*jumper* tancat), a Vmotor se li haurà de subministrar un voltatge d'entre 6V i 12V, i Vlògic serà una sortida de 5V que podem utilitzar per alimentar altres dispositius.

- Si el regulador està desactivat (*jumper* obert), a Vmotor se li haurà de subministrar entre 12V i 35V, i Vlògic serà una entrada a la qual se li haurà de subministrar entre 4.4V i 5V.

A cada una de les bandes del mòdul trobem uns terminals de connexió (A i B) que subministren la sortida als motors.

Per últim, es té els pins d'entrada que controlen la direcció i la velocitat de gir.

- Els pins IN1, IN2, i IN3 i IN4, controlen la direcció de gir de la sortida A i B, respectivament.

- Els pins IEA i IEB desactiven la sortida. Es poden connectar permanentment mitjançant un *jumper*, o connectar un senyal PWM per controlar la velocitat de gir.

En el cas de Carpher s'utilitzen dues fases per tal de poder escollir tant el gir com la velocitat. El mòdul s'alimenta amb una bateria de 7,2V. A la Figura 52 es pot veure l'esquema de connexió del mòdul.

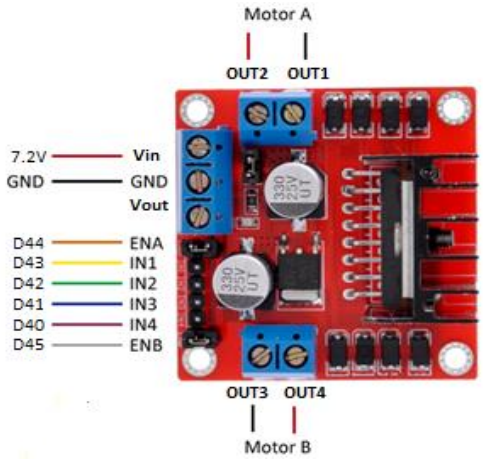


Figura 52. Esquema de connexió del mòdul L298N.

A la Figura 53 es veu el resultat de realitzar la connexió en provar el funcionament dels components.

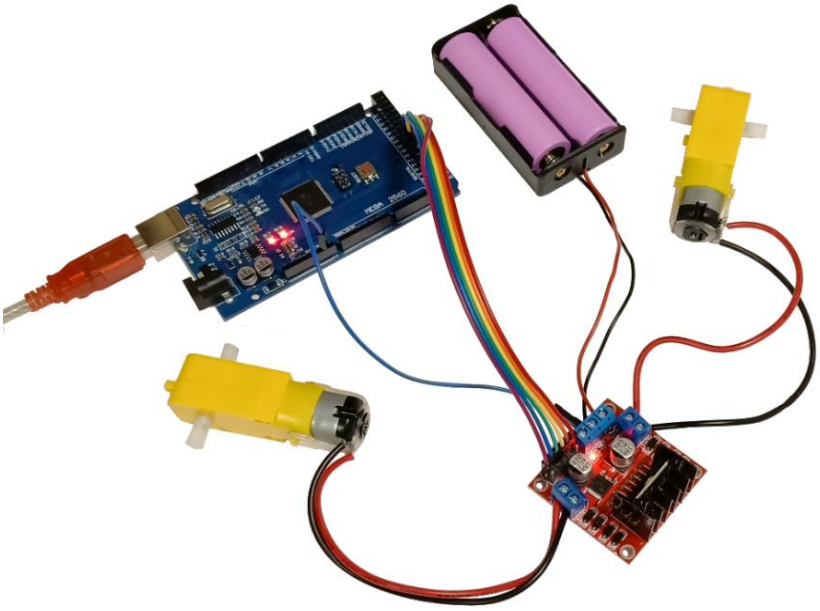


Figura 53. Muntatge de prova del controlador de motors L298N.

Programació del mòdul amb Arduino

El primer que s'ha de fer és indicar a la placa els pins què s'utilitzen per controlar els motors, aquests es declaren com a sortides al `setup()` del programa. A continuació, en forma de constant, s'agrupa en dues llistes diferents els pins que corresponen al control del motor A i els de motor B.

```
const int pinMotorA[3] = { pinENA, pinIN1, pinIN2 };
const int pinMotorB[3] = { pinENB, pinIN3, pinIN4 };
```

Ara que ja estan definits els pins dels motors es creen tres funcions, la primera fa que el motor giri endavant, la segona que giri endarrere i la tercera que s'aturi en sec.

```
void moveForward(const int pinMotor[3], int speed){

   digitalWrite(pinMotor[1], HIGH);
   digitalWrite(pinMotor[2], LOW);
   analogWrite(pinMotor[0], speed);
}


void moveBackward(const int pinMotor[3], int speed){

   digitalWrite(pinMotor[1], LOW);
   digitalWrite(pinMotor[2], HIGH);
   analogWrite(pinMotor[0], speed);
}

void fullStop(const int pinMotor[3]){

   digitalWrite(pinMotor[1], LOW);
   digitalWrite(pinMotor[2], LOW);
   analogWrite(pinMotor[0], 0);
}
```

Com es pot veure, el pinIN1 i el pinIN2 corresponen a les dues branques del pont-H, i per tant, posant el primer en HIGH i el segon en LOW, farem que el motor giri endavant. En canvi, si invertim l'estat lògic, IN2 a HIGH i IN1 a LOW, el sentit de gir també s'invertirà fent que el motor giri endarrere. Per altra banda, en posar els dos pins IN a LOW es deixarà de passar senyal al motor, el què farà que s'aturi en sec.

El `pinEnX` (*enable*) serà el que controli la velocitat, sent un enter entre 0 i 255.

En el `loop()` simplement és crida a les funcions indicant-li'ls el motor què es vol controlar i la velocitat de gir, seguit del temps que ha de mantenir aquest estat. A continuació es mostra un exemple en què el Carpher gira cap a la banda (dreta o esquerra) on se situa el motor durant un temps de cinc segons.

```
void loop(){

   moveForward(pinMotorA, 255);
   moveBackward(pinMotorB, 125);
   delay(5000);
}
```

2.3.3 Placa de prototipat (Protoboard)

Una protoboard, Figura 54 i 55, és una eina que constà de forats que permeten fixar qualsevol component amb pins. Serveix per fer circuits temporals que no requereixen soldadura. Existeixen molts models de plaques i es poden diferenciar pel nombre de forats que tenen.

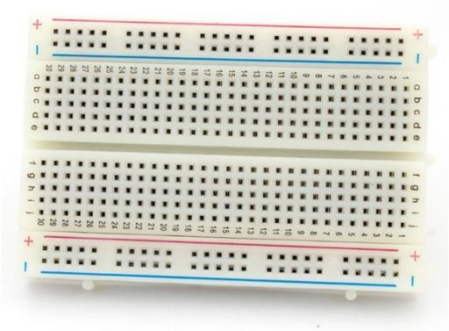


Figura 54. Protoboard de 400 punts comercialitzada per AFE.

Figura 55. Miniprotoboard de 170 punts comercialitzada per TBem.

Com es veu en la Figura 56, les protoboard tenen tres zones. Les línies blaves i vermelles corresponen a la zona d'alimentació i estan compostes per una sèrie de forats col·locats en horitzontal i connectats elèctricament entre si. Per altra banda, les zones de connexió estan compostes per columnes de forats connectats elèctricament entre si. Cada columna és independent a les altres.

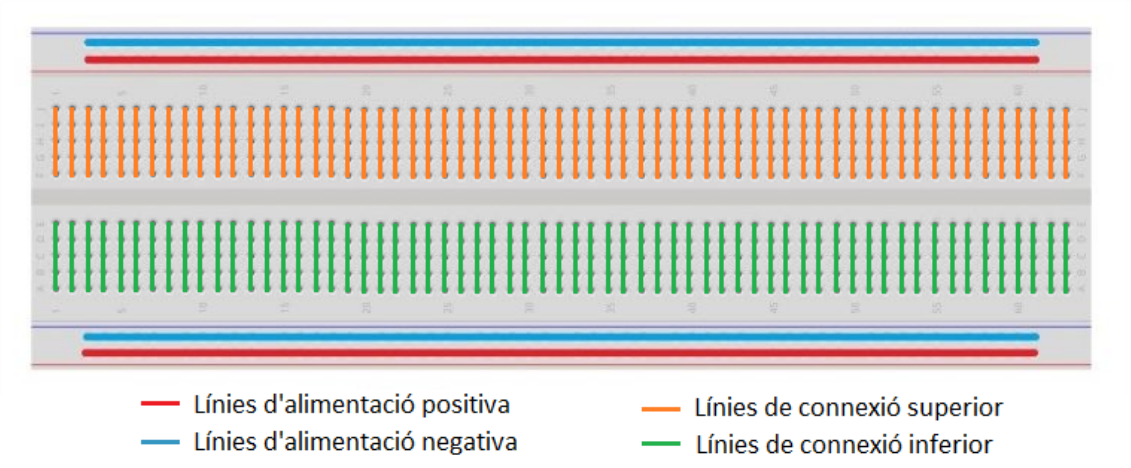


Figura 56. Esquema de connexions elèctriques per zones d'una protoboard de 830 punts.

2.3.4 Sensor de distància

Com diu el seu nom, els sensors de distància mesuren la proximitat a un objecte. Són molt utilitzats en els robots mòbils, principalment per evitar xocs.

En aquest projecte s'utilitzen dos sensors que funcionen amb diferents tecnologies. El primer és d'ultrasons i permet evitar xocs. El segon és làser i proporciona les dades pel posterior mapatge de l'entorn.

2.3.4.1 Sensor ultrasons HCSR04

Funcionament del sensor d'ultrasons

El seu funcionament, tal com es pot veure a la Figura 57, es basa a enviar un pols d'alta freqüència no audible per l'ésser humà que rebota en els objectes propers i es reflecteix de nou cap al sensor. Aquest, disposa d'un micròfon adequat per la freqüència i la capta. Mesurant els temps entre polsos i coneixent la velocitat del soroll, es pot estimar la distància de l'objecte.

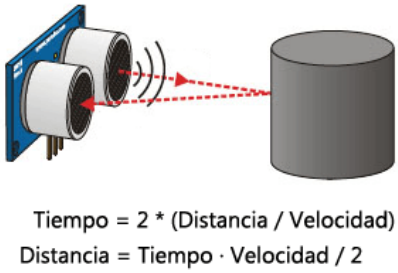


Figura 57. Funcionament d'un sensor d'ultrasons.

Tot i ser el sensor de distància més utilitzat en l'entorn d'Arduino, són els que tenen més baixa precisió perquè el rang de l'ona és bastant gran i per tant, és susceptible a rebotar entre objectes i crear eco falsejant el resultat. Això es pot veure com un desavantatge o com un avantatge, ja que aquests rebots permeten evitar, per exemple, el xoc amb la pota d'una cadira en el cas d'un robot que tingui la part frontal més amplia que el rang del mateix sensor.

Característiques del sensor escollit

S'ha escollit el sensor HC-SR04, Figura 58, pel seu preu econòmic i a la seva accessibilitat. Aquest s'ha instal·lat a la part frontal del robot mòbil i la seva única funcionalitat és evitar xocs amb els objectes de l'entorn.

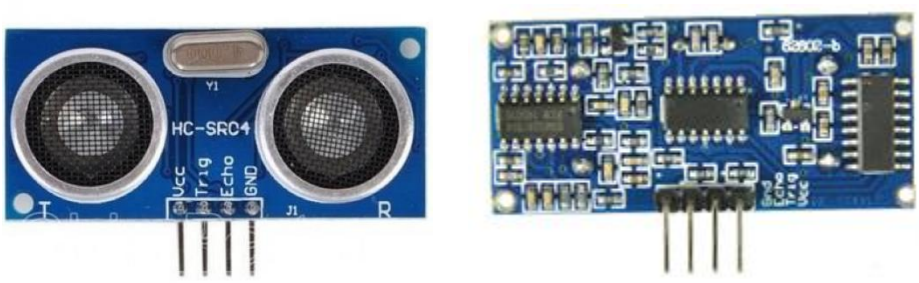


Figura 58. Sensor d'ultrasons HC-SR04.

A continuació es llisten algunes de les característiques del sensor:

- Alimentació a 5 VDC.
- Angle d'acció menor a 15°.
- Mesura de distàncies de 2 a 400 cm.
- Consum de corrent 15mA.
- Resolució d'1 cm.

Càlcul de la distància mitjançant polsos

Com ja s'ha explicat el sensor HC-SR04 està compost per un emissor de soroll d'alta freqüència i un receptor. Per iniciar la mesura, es genera un *trigger* (tret) de 10 microsegons. Un tret equival a una transmissió de 8 polsos de soroll a 40kHz. Quan el sensor detecta el soroll retornat per algun objecte proper a través del micròfon, el pin ECHO es col·loca a 5V i es manté així per un temps proporcional a la distància. A la Figura 59 es pot veure la seqüència de polsos.

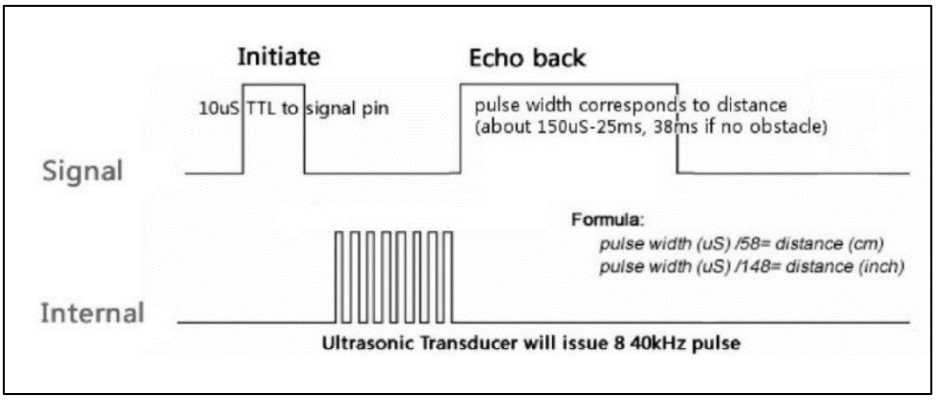


Figura 59. Diagrama de sincronització de polsos del sensor d'ultrasons HC-SR04.

Per saber la distància a partir d'aquest temps s'utilitza la següent fórmula:

$$distancia = (temps\ ECHO\ [s]) \cdot \frac{velocitat\ soroll\ \left[\frac{m}{s}\right]}{2}$$

On,

$temps\ ECHO$, donat per la funció `ping()` (Aquesta s'explicarà a més tard) en µs.

$Velocitat\ soroll$, serà 343 m/s en condicions de temperatura a 20ºC i 50% d'humitat i pressió atmosfèrica a nivell del mar.

Es divideix el temps en dos perquè s'ha de tenir en compte que el temps obtingut és el que tarda el pols a anar i tornar. Per tant, la fórmula per obtenir cm a partir de l'ús de la funció `ping()` quedarà de la següent forma:

$$distancia\ [cm] = (temps\ ECHO\ [\mu s]) \cdot \frac{0{,}0343\ \left[\frac{cm}{\mu s}\right]}{2} = \frac{temps\ ECHO\ [\mu s]}{29{,}2 \cdot 2}$$

Connexió del sensor a la placa Arduino

A la Figura 60 és mostra la connexió del sensor amb la placa d'Arduino. El pin Vcc va connectat a la sortida de 5V del controlador. I l'Echo i el Trigger van a pins digitals.

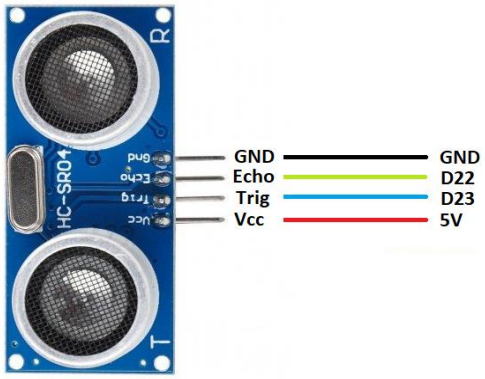


Figura 60. Esquema de connexió del sensor HC-SR04.

A la Figura 61 és pot veure el sensor connectat i funcionant.

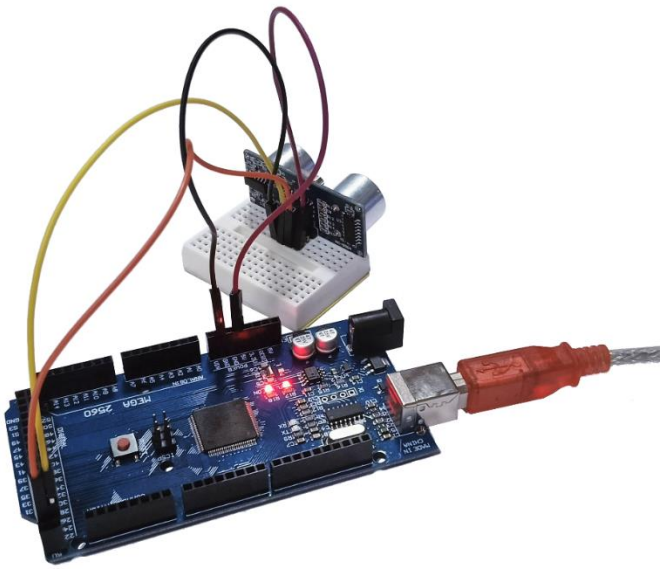


Figura 61. Muntatge de prova del sensor de distància HC-SR04.

Programació

En el `setup()` simplement s'ha de declarar el pin del trigger com a sortida i el de l'eco com a entrada.

Per facilitar l'ús del sensor en un sistema més gran, s'ha creat una funció que s'encarrega dels passos necessaris per retornar el valor en cm de la distància de l'objecte més proper.

```
int ping(int TriggerPin, int EchoPin) {

   unsigned long duration, distanceCm;

   digitalWrite(TriggerPin, LOW);
   delayMicroseconds(4);
   digitalWrite(TriggerPin, HIGH);
   delayMicroseconds(10);
   digitalWrite(TriggerPin, LOW);

   duration = pulseIn(EchoPin, HIGH);

   distanceCm = duration * 10 / 292/ 2;
   return distanceCm;

}
```

La funció anterior produeix un trigger de 10 microsegons i és guarda, a la variable `duration`, el temps que s'ha tardat a rebre l'ona. Després, mitjançant els càlculs explicats anteriorment, s'obté la distància de l'objecte.

Quan es vulgui obtenir la mesura del sensor en el `loop()` senzillament s'ha de cridar la funció assignant-la a una variable.

```
int cm = ping(TriggerPin, EchoPin);
```

2.3.4.2 Sensor làser VL53L0X

S'han implementat cinc sensors làser al voltant del robot Carpher, cobrint la part frontal i els laterals i un altre a la part superior de forma que pugui detectar l'alçada del sostre. Aquest últim sensor només s'activa si l'usuari, persona que controla el robot, assegura que el mòbil no està a la vista de cap ésser viu, ja que el làser pot resultar nociu per la

vista. El conjunt de sensors permeten a posteriori enviar les dades recollides al PC i, d'aquesta manera, recrear virtualment l'espai on es troba el robot.

Característiques del sensor escollit

S'ha escollit el sensor VL53L0X, Figura 62, ja que té una gran precisió i un angle de mesura molt estret. És capaç d'operar en condicions d'elevada llum ambiental infraroja i incorpora un sistema de compensació del mesurament que permet que funcioni darrere d'un cristall protector.

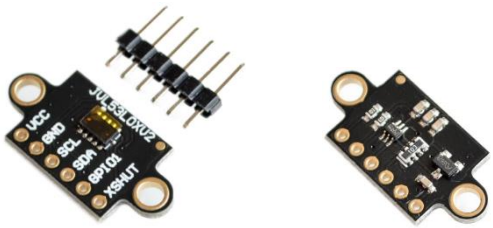


Figura 62. Sensor làser VL53L0X.

A continuació es llisten algunes de les característiques del sensor:

- Alimentació a 5 VDC.
- Consum de corrent 15mA.
- Angle d'acció de 25º.
- Mesura de distàncies de 5 a 120 cm.
- Precisió del 4% al 12%, depèn del color del objecte i llum ambient.

Funcionament del sensor ToF (*Time of Flight*)

Consisteix en enviar un pols làser de llum infraroja i mesurar el temps que tarda el feix en retornar al sensor.

L'integrat incorpora un emissor làser de 940nm VCSEL (*vertical cavity surface-emitting laser*), un detector SPAD (*single photon avalanche diodes*) i l'electrònica interna, anomenada FlightSense™, encarregada de realitzar els càlculs necessaris.

Protocol de comunicació I2C

I2C es un protocol de comunicació en sèrie de curta distància on les dades són transferides bit a bit per la línia d'SDA.

En el cas de Carpher s'ha de connectar més d'un sensor a la placa Arduino, el que significa haver de connectar més d'un dispositiu al bus I2C.

El bus només necessita dos cables per funcionar, un pel senyal del rellotge (SCL) i un altre per enviar dades (SDA). També necessita cables d'alimentació, el GND i el Vcc entre +5V i 3.3V. Tant la línia SDA com SCL són síncrones, bidireccionals i connectades amb resistències *pull-up*. A la Figura 63 es pot veure l'estructura en tenir connectats més d'un dispositiu.

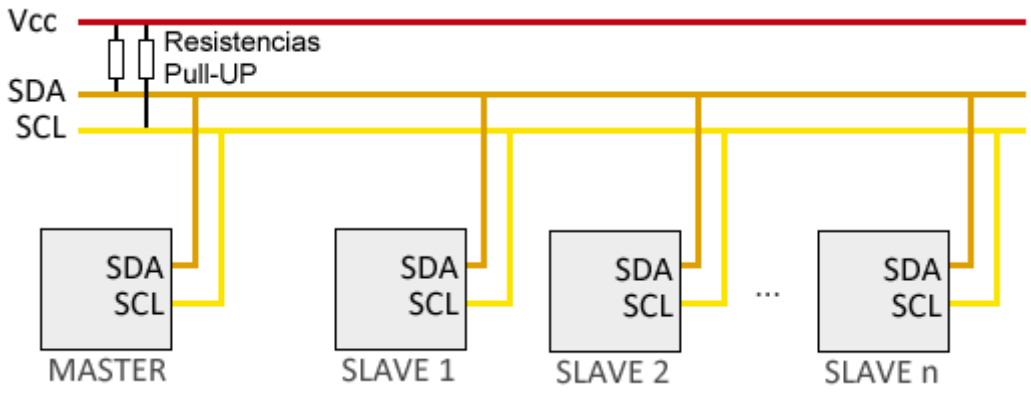


Figura 63. Esquema de connexió comunicació I2C.

Cada dispositiu ha de tenir una direcció única que s'utilitzarà per accedir al dispositiu de forma individual. Es disposa de 7 bits de direcció, el que significa què és possible accedir a 112 dispositius. Les 15 adreces restants estan reservades.

El bus té una arquitectura de tipus mestre-esclau. Significa que els esclaus no poden enviar dades pel bus a no ser que el mestre hagi iniciat la comunicació amb ells. Els esclaus no és poden comunicar entre si. L'enviament de bits es sincronitza mitjançant el senyal de rellotge (SCL), compartida entre el mestre i l'esclau i sempre controlada pel mestre.

La velocitat del bus I2C és reduïda, l'estàndard és de 100MHz, amb un mode d'alta velocitat de 400MHZ.

Per utilitzar el bus I2C a l'Arduino, l'IDE Standard proporciona la llibreria "`Wire.h`", que conté les funcions necessàries per controlar el hardware integrat.

Algunes de les funcions bàsiques i que s'usen en el projecte són les següents:

```
Wire.begin()  // Inicialitza el hardware del bus
Wire.beginTransmission(address); //Comiença la transmisió
Wire.endTransmission(); // Finaliza la transmisió
Wire.requestFrom(address,nBytes);  //solicita un número de bytes
                                   al esclau en la direcció address
Wire.available(); // Detecta si hi ha dades pendents de ser llegides
Wire.write();  // Envia un byte
Wire.read();   // Rep un byte

Wire.onReceive(handler); // Registra una funció de callback al rebre
                            una dada
Wire.onRequest(handler); // Registra una función de callback al
                            solicitar una dada
```

CONÈIXER LA DIRECCIÓ DEL DISPOSITIU

De vegades, en adquirir un dispositiu amb funcionament per I2C, el fabricant no proporciona la direcció del dispositiu. Per aquest motiu existeix l'*sketch* anomenat *"Scanner I2C"*, que fa un recorregut per totes les possibles direccions del bus i mostra el resultat en cas de trobar un dispositiu connectat.

L'*sketch* es copia directament de la pàgina web d'Arduino i s'enganxa a l'IDE (*integrated development environment*). Un cop connectat el dispositiu del qual volem saber l'adreça, carreguem el programa a la placa i pels ports sèrie (Figura 64), podem veure la direcció on es troba.

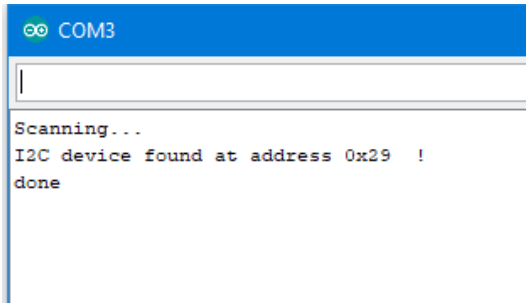


Figura 64. Finestra port sèrie al haver executat el I2Cscanner.

Tots els sensors làsers comparats tenen la mateixa adreça, fet molt usual en tractar-se de components del mateix fabricant. La direcció predeterminada pel VL53L0X és la 0x29, tot i que pot canviar-se amb el software. Per fer-ho s'han d'apagar tots els sensors i, a continuació, engegar-los un per un posant a *high* el pin XSHUT. Durant la seva inicialització s'ha de cridar `lox.begin(new_adress)`. S'ha de tenir en compte que aquest canvi no és permanent i que per tant, en reiniciar la placa, les adreces tornaran a la predeterminada.

Connexió del sensor a la placa Arduino

La connexió d'un sensor és senzilla, ja que la comunicació es realitza a través de l'I2C (*inter-integrated circuit*). Per tant, simplement s'alimenta el mòdul des de l'Arduino mitjançant el GND i 5V i connectem el pin SDA i SCL d'Arduino amb els pins corresponents del sensor. XSHUT servirà per apagar i engegar el sensor per programa. A la Figura 65 és mostra l'esquema de connexió.

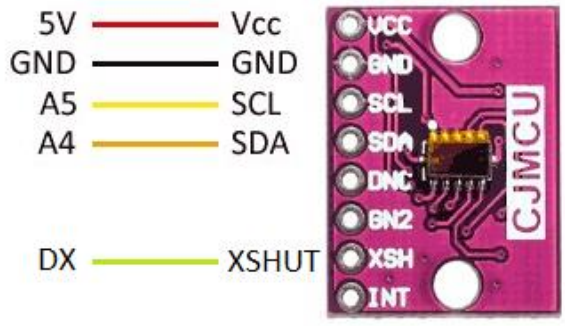


Figura 65. Esquema connexió sensor VL53L0X.

Per la construcció de Carpher s'ha connectat més d'un sensor làser, per tant s'han connectat a la línia de SDA d'Arduino tots els pins SDA dels sensors i el mateix per la SCL. També s'ha connectat el pin XSHUT de cada un dels sensors a una entrada/sortida digital diferent de l'Arduino. L'esquema resultant per sis sensors es pot veure en la Figura 66.

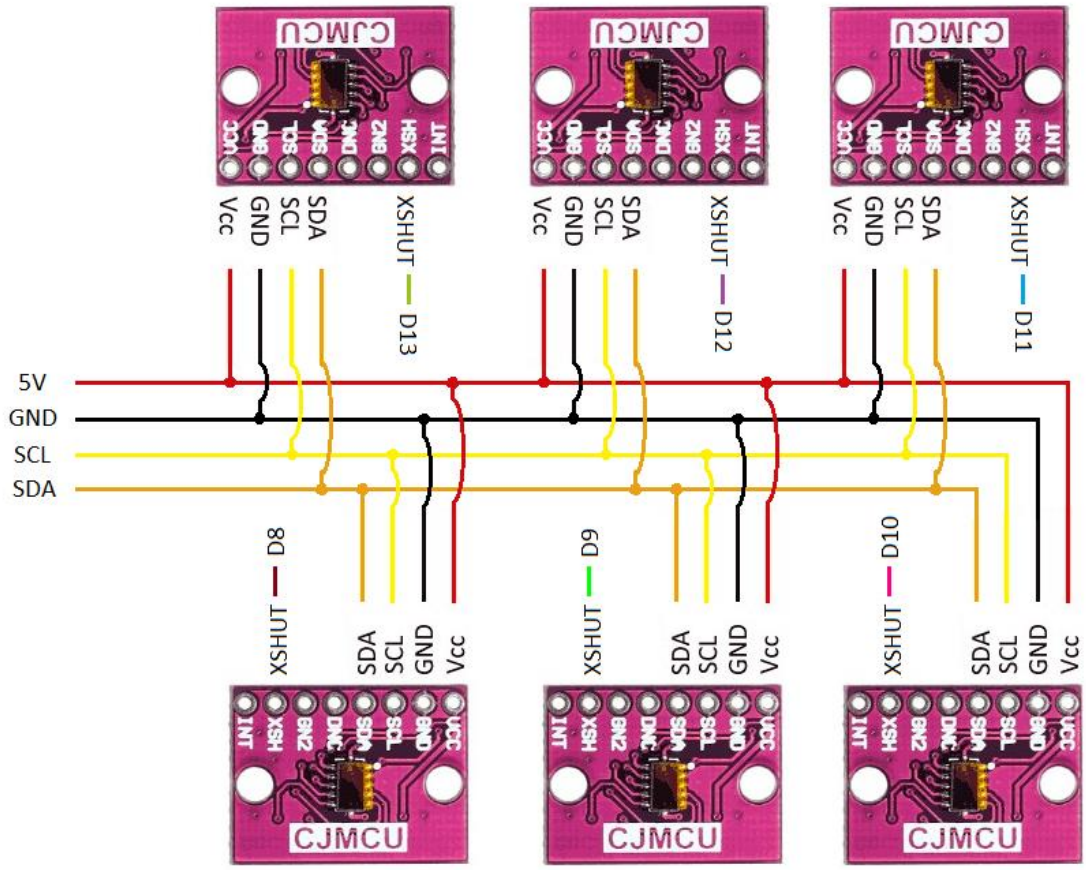


Figura 66. Esquema connexió sis sensors VL53L0X.

Programació sensor làser

El codi que és mostra a continuació és aplicable a dos sensors, si se'n volguessin posar més, simplement s'haurien d'afegir les mateixes línies però canviant el nom de l'objecte.

Primer s'ha d'incloure la llibreria `"Adafruit_VL53L0X"` pel control del sensor, aquesta es pot trobar de forma gratuïta a GitHub. Seguidament es defineixen les adreces

que se'ls ha donat als sensors i els pins XSHUT de cada un. Després es crea un objecte per cada un dels sensors i una variable què guardarà el valor de la mesura.

```
#include "Adafruit_VL53L0X.h"

// address we will assign if dual sensor is present
#define LOX1_ADDRESS 0x30
#define LOX2_ADDRESS 0x31

// set the pins to shutdown
#define SHT_LOX1 11
#define SHT_LOX2 12

// objects for the vl53l0x
Adafruit_VL53L0X lox1 = Adafruit_VL53L0X();
Adafruit_VL53L0X lox2 = Adafruit_VL53L0X();

// this holds the measurement
VL53L0X_RangingMeasurementData_t measure1;
VL53L0X_RangingMeasurementData_t measure2;
```

A la part del `setup()` es declaren els pins XSHUT com a sortides i es posen a LOW durant 10ms, seguidament es posen a HIGH durant 10ms per tal de fer un reinici a cada un dels sensors. A continuació es posa a HIGH el primer sensor i la resta a LOW, d'aquesta manera ja es pot inicialitzar el primer sensor. En el següent codi és mostra el procediment:

```
  // initing LOX1
  If (!lox1.begin(LOX1_ADDRESS)){ //S'inicialitza el sensor amb la nova adreca
        Serial.println(F("Failed to boot first VL53L0X"));
        while (1);
  }
  delay(10);
```

Un cop inicialitzat el primer es posa el XSHUT, mentre que el segon sensor es posa a HIGH. Un cop fet, ja es pot repetir el procés.

En el `loop()` es crida una funció de la llibreria per tal d'obtenir dades, aquestes es guarden a l'objecte *measure1*, què s'ha definit a la part de les declaracions així com les característiques derivades d'aquesta.

```
// Inicia el mostreig
lox1.rangingTest(&measure1, false); //per obtenir més informació posar true


// Imprimeix pel port sèrie els resultats
if (measure1.RangeStatus != 4){ // Si la mesura no està fora de rang
      Serial.print(measure1.RangeMilliMeter);  // S'escriu el valor
} else {
      Serial.print("Out of range");
}
```

2.3.5 Sensor de posició

En aquest punt s'expliquen els dos tipus de sensors utilitzats per tal de saber la posició de Carpher en l'espai. El primer és l'encoder, que permet saber la velocitat de gir dels motors i, mitjançant càlculs, podem obtenir els graus de gir i conseqüentment la distància que s'ha recorregut. Per altra banda s'ha implementat un giroscopi que permet conèixer en quina direcció està orientat Carpher, aquest segon sensor s'utilitza com a complement del primer per controlar millor els girs i l'orientació en l'espai.

2.3.5.1 Encoder

En aquest projecte s'han utilitzat codificadors amb optointerruptors. Aquests tenen en un extrem un díode emissor de llum infraroja i en l'altre un fototransistor que rep el senyal. Quan un objecte passa per la ranura interromp el raig de llum i és detectat pel fototransistor. Al mercat es poden trobar mòduls com el FZ0888 (Figura 67), que està dissenyat per funcionar amb Arduino. El mòdul a part de l'optointerroptor, porta un comparador diferencial anomenat LM393. Aquest dispositiu consisteix en dos comparadors de voltatge independents que estan dissenyats per operar des d'una única font d'alimentació.

Figura 67. Encoder FZ0888.

Aquest tipus de sensor són molt utilitzats per detectar la velocitat de gir i la posició de l'eix d'un motor. Per fer-ho, s'acobla a l'eix del motor una rodeta amb ranures (Figura 68) i es compten els polsos què es reben en un determinat temps per saber els graus de gir del motor. Aquest compte es fa mitjançant una interrupció.

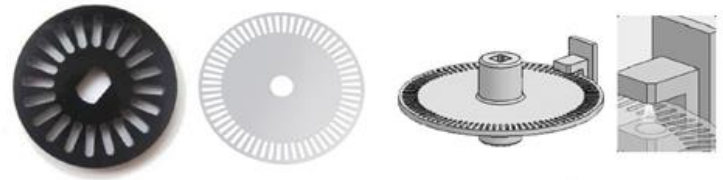


Figura 68. Roda dentada i funcionament d'aquesta.

Càlculs

La rodeta dentada que s'ha utilitzat per a la construcció de Carpher consta de 20 forats, per tant se sap que un gir complet del motor equival a 20 polsos. Per obtenir mm/s s'ha de contar els polsos que s'envien en un segon. Després, saben la mida de la circumferència de la roda, es poden saber els mil·límetres que s'han recorregut.

Per obtenir la circumferència ho farem a partir del diàmetre de la roda:

$$Circumferència = 2\pi R = 2 \cdot 3{,}14 \cdot \frac{67{,}5\ mm}{2} = 212{,}06\ mm$$

En conclusió, cada 20 polsos el mòbil recorre 212,06 mm què, passats a unitat de polsos, equival a 10,6 mm per pols. Per calcular els mm/s s'utilitza la següent fórmula:

$$Velocitat = \frac{N\ polsos}{1\ segon} \cdot \frac{0{,}0106\ mm}{1\ pols}$$

Interrupcions en Arduino

Les interrupcions són un mecanisme que permet associar una funció a un esdeveniment. La funció s'executa quan es produeix l'esdeveniment. Aquesta funció *callback* s'anomena ISR (*interruption service routine*).

Quan succeeix l'esdeveniment el processador surt immediatament del flux normal del programa i executa la funció ISR associada, ignorant per complet qualsevol altra tasca. En finalitzar aquesta funció, el processador torna al flux principal, en el mateix punt on s'havia interromput.

Arduino és capaç de detectar dos tipus d'esdeveniments, per una banda hi ha les interrupcions de *timers*, que en aquest projecte no es tractaran, i per altra banda les interrupcions per hardware, que responen a esdeveniments en certs pins físics.

Dins de les interrupcions per hardware, que són les que s'utilitzen, Arduino és capaç de detectar els següents esdeveniments:

- RISING, quan l'estat del pin passa de LOW a HIGH.
- FALLING, quan passa de HIGH a LOW.
- CHANGING, quan passa qualsevol dels dos anteriors.
- LOW, s'executa contínuament quan el pin es troba en LOW.

Els pins susceptibles a interrupcions varien en funció del model d'Arduino, el MEGA, que és l'utilitzat, disposa de 6 interrupcions en els pins 2, 3, 21, 20, 19 i 18.

És molt important que el temps d'execució de la funció d'interrupció sigui el més baix possible, ja que la resta de programa queda aturat i en alguns casos podria desestabilitzar el sistema. Per exemple, si s'executa la funció quan un motor està apropant un braç a un objecte, pot ser que aquest no s'aturi a temps i tiri l'objecte o el faci malbé. També s'ha de tenir en compte que la funció millis() no s'actualitza, i això pot acabar provocant un desfasament en la mesura del temps.

Les variables dins d'una ISR han de ser declarades com a volàtils. Així s'indica al compilador que la variable ha de ser consultada sempre abans de ser utilitzada, ja que pot haver estat modificada de forma aliena al flux del programa principal.

Connexió del mòdul

L'esquema de muntatge del codificador és realment senzill, l'alimentació del mòdul es pot fer de 3,3V a 12V. La recomanació és connectar-lo directament a la sortida de 3,3V de l'Arduino UNO, ja que aquesta sortida no té tants rebots com podria tenir la de 5V. Tot i així, si encara hi ha soroll, es pot connectar un condensador ceràmic de 100nF entre el pin D0 i el GND. Per últim s'utilitza una entrada digital per connectar el pin D0 del codificador, s'ha de tenir en compte què aquesta entrada ha de ser un pin d'interrupció, ja que ajudarà a comptabilitzar els polsos de forma precisa.

A la Figura 69 es pot veure la connexió de cada encoder i a la Figura 70 el muntatge que s'ha fet per tal de comprovar el seu funcionament.

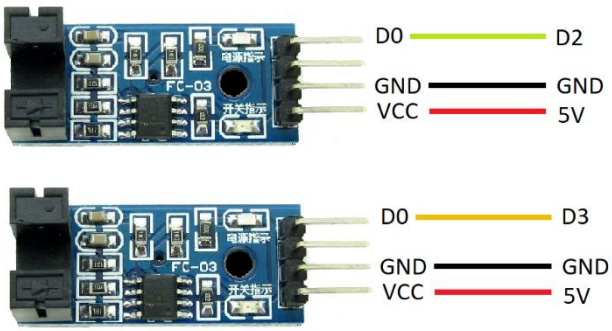


Figura 69. Esquema de connexió encoders.

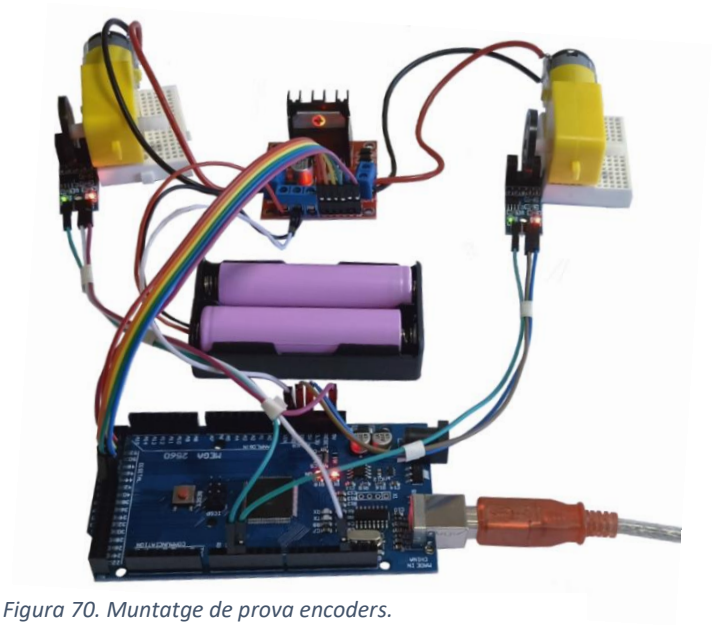


Figura 70. Muntatge de prova encoders.

Programació

En aquesta part s'expliquen les instruccions bàsiques pel funcionament dels codificadors.

Primer de tot s'han de declarar les variables necessàries:

```
float velocity = 0;        //Resultat del càlcul de la velocitat

                           // en mm/s
volatile byte pulses = 0;   //Comptador que s'incrementa amb
                            // cada pols
unsigned long timeold = 0;    //Temps en que s'ha fet l'últim
                              // reset del comptador

unsigned int pulsesperturn = 20;  //Numero de forarts que te el
                                  // disc del codificador
const long wheel_diameter = 67.5;   //Diametre de la roda en mm
float distance_per_pulse = 0.0;  //Resultat càlcul de la distancia
                                 // que recorre per cada pols
```

En el `setup()` s'inicialitza el pin del codificador com a entrada amb la funció `pinMode()`. Tot seguit s'inicialitza la interrupció.

```
attachInterrupt(digitalPinToInterrupt(2), lectura_ecoder, RISING);
```

La primera variable és el pin en què es vol associar la interrupció, la segona variable és el nom que porta la ISR i per últim, l'esdeveniment que fa que la funció s'executi. En aquest cas el què es vol és que s'incrementi un comptador cada vegada que es rebi un pols positiu, és a dir, quan el feix de llum travessa un dels forats de la roda del codificador. Per tant l'esdeveniment que interessa detectar és el *RISING*.

Per tancar el `setup()` s'ha de calcular la variable `distance_per_pulses`, explicada anteriorment en els càlculs. El resultat seran mm/pols.

```
distance_per_pulse = 2 * 3.1416 * (wheel_diameter / 2) / pulsesperturn;
```

A continuació escrivim la ISR, que serà declarada com a `void`, ja que no ha de retornar cap valor. Recordar que les instruccions han de ser el més breu possible, pel que només fem que s'incrementi el comptador de polsos.

```
void lectura_ecoder() {
  pulses++;
}
```

Per últim s'escriu el `loop()`, que comprova que hagin passat 1000 ms des de l'últim reinici del comptador i que calcula la velocitat del motor amb la fórmula que s'ha explicat anteriorment en la part dels càlculs. Després, guarda a la variable `timeold`, el temps actual, i posa a 0 el comptador de polsos. S'imprimeixen els resultats dels càlculs per pantalla.

```
void loop() {

   if ( (millis() - timeold) >= 1000 ) {

     char text1[50];
     sprintf(text1, "Polsos          %i", (int)pulses);
     Serial.println(text1);
```

```
    velocity = (pulses * distance_per_pulse * 1000) / ( (millis() -
timeold));
    timeold = millis();
    pulses = 0;

    sprintf(text1, "Velocitat mm/s  %d.%02d", (int)velocity,
(int)(velocity * 100) % 100);
    Serial.println(text1);
  }

}
```

2.3.5.2 Sistema de mesura inercial MPU-9250

En aquest projecte s'utilitza el sensor MPU-9250, Figura 71 i 72, fabricat per l'empresa Invensesen. Aquest incorpora, en un mateix integrat, una IMU MPU-6050 que al seu torn integra un acceleròmetre de tres eixos i un giroscopi de tres eixos en un mateix xip, així com un magnetòmetre AK8963 també de tres eixos. Per tant, és un IMU de nou graus de llibertat (9DOF) que permet, mitjançant càlculs, conèixer tota la informació necessària per determinar l'orientació i acceleracions a la que està sotmès.

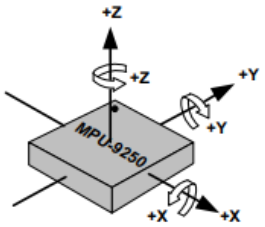


Figura 71. Orientació dels eixos en el sensor MPU-9250.

Figura 72. Sensor MPU-6050.

També trobem un DMP (*Digital Movements Processor)* que és pot programar amb firmware per tal que realitzi càlculs complexos de forma ràpida amb els valors del sensor. Això redueix la càrrega pel microcontrolador, ja que no és ell qui els ha de realitzar.

El sensor utilitza el bus I2C, explicat anteriorment pel sensor làser, per interactuar amb Arduino.

Altres característiques del sensor:

- Giroscopi amb una precisió programable de ±250, ±500, ±1000 i ±2000 °/sec.
- Acceleròmetre amb una precisió programable de ±2g, ±4g, ±8 I ±16g.

  (gravetat terrestre => 9,80665 m/s2 = 1g)

- Magnetòmetre amb una precisió de ±4800μT. (Tesla => 1T = 1 kg·s−1·C−1)

Obtenció de valors i càlculs necessaris

Els valors que ens proporciona l'IMU són enters de 16 bits, a continuació es veu una taula que mostra les diferents dades què podem obtenir directament d'aquest i sobre les que es treballarà posteriorment:

Addr (Hex)	Addr (Dec.)	Nom Registre	Serial I/F	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Descripció
3B	59	ACCEL_XOUT_H	R	ACCEL_XOUT[15:8]								Acceleració eix X
3C	60	ACCEL_XOUT_L	R	ACCEL_XOUT[7:0]								
3D	61	ACCEL_YOUT_H	R	ACCEL_YOUT[15:8]								Acceleració eix Y
3E	62	ACCEL_YOUT_L	R	ACCEL_YOUT[7:0]								
3F	63	ACCEL_ZOUT_H	R	ACCEL_ZOUT[15:8]								Acceleració eix Z
40	64	ACCEL_ZOUT_L	R	ACCEL_ZOUT[7:0]								
41	65	TEMP_OUT_H	R	TEMP_OUT[15:8]								Temperatura
42	66	TEMP_OUT_L	R	TEMP_OUT[7:0]								
43	67	GYRO_XOUT_H	R	GYRO_XOUT[15:8]								Giroscopi eix X
44	68	GYRO_XOUT_L	R	GYRO_XOUT[7:0]								
45	69	GYRO_YOUT_H	R	GYRO_YOUT[15:8]								Giroscopi eix Y
46	70	GYRO_YOUT_L	R	GYRO_YOUT[7:0]								
47	71	GYRO_ZOUT_H	R	GYRO_ZOUT[15:8]								Giroscopi eix Z
48	72	GYRO_ZOUT_L	R	GYRO_ZOUT[7:0]								

Taula 3. Mesures què és poden obtenir del MPU-6050. Font: Datasheet MPU-6050 Adafruit.

Si es llegeixen directament els valors del sensor s'obtenen uns valors bruts que posteriorment s'hauran de tractar per obtenir uns valors utilitzables.

A Internet es poden trobar moltes llibreries que realitzen els càlculs necessaris per aconseguir filtrar aquests valors en brut. Després de provar-ne de varis, s'ha decidit

utilitzar la creada per l'usuari *hideakitai* de GitHub. Aquesta és una modificació més complexa de l'original de Kris Winer.

Connexió del sensor

El mòdul GY-521 té un regulador de voltatge que permet connectar Vcc directament als 5V de la placa Arduino, recordar que la tensió màxima que pot suportar el sensor GPU-6050 és de 3.4V. Els pins SDA i SCL van connectats als pins 21 i 20 de la part de comunicació de la placa MEGA. El mòdul ja incorpora resistències *pull-up* de 10k Ohms així que no és necessari incorporar-les externament. A la Figura 73 és pot veure l'esquema del connexió del mòdul.

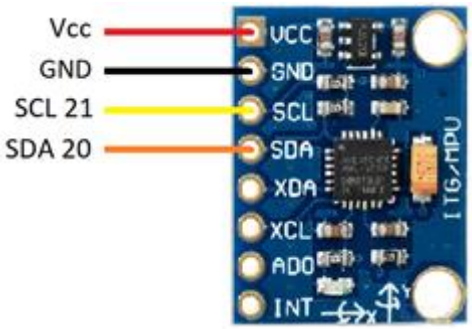


Figura 73. Esquema connexió MPU-9250

Programació del mòdul

El primer pas és conèixer la direcció I2C que té assignat el mòdul, per saber-ho s'ha carregat a la placa el programa del I2Cscanner. A l'executar el programa, aquest indica que és troba a l'adreça 0x68.

En el codi d'Arduino primer s'inclou la llibreria *MPU9250.h* i es crea un objecte anomenat `mpu`. En el `setup()` s'inicialitza la comunicació I2C mitjançant la funció `Wire.begin()` i seguidament s'inicialitza el giroscopi utilitzant `mpu.setup()`.

En el `loop()` cada x mil·lisegons s'actualitzen les variables mitjançant `mpu.update()` i després, ja es poden obtenir el balanceig (`mpu.getRoll()`), la guinyada (`mpu.getYaw()`) i el capcineig (`mpu.getPitch()`).

2.3.6 Sensor de temperatura i humitat

El DHT22 (Figura 74) és un sensor que permet obtenir de forma simultània mesures de temperatura i humitat. Té unes característiques acceptables per projectes que necessitin d'una precisió mitjana, com és en el cas de Carpher. Els sensors de temperatura, i sobretot els d'humitat, compten amb una inèrcia i un temps de resposta elevats. És a dir, són lents en mostrar un canvi de les seves mesures. Algunes de les característiques més importants són:

- Mesurament de temperatura entre -40 a 125 ºC, amb una precisió de 0,5ºC
- Mesurament d'humitat entre 0 i 100%, amb una precisió de 2-5%
- Freqüència de mostreig de dues mostres per segon (2Hz).

Figura 74. Sensor de temperatura i humitat DHT22.

Connexió del sensor

Dels quatre pins que té el sensor, només se n'utilitzen tres. Per alimentar-lo es fa des de l'Arduino a través dels pins GND i 5V. Per altra banda, la sortida es connecta a una entrada digital. És necessària una resistència de 10kOhms entre Vcc i el pin de sortida. A la figura 75 és mostra l'esquema de connexió del sensor.

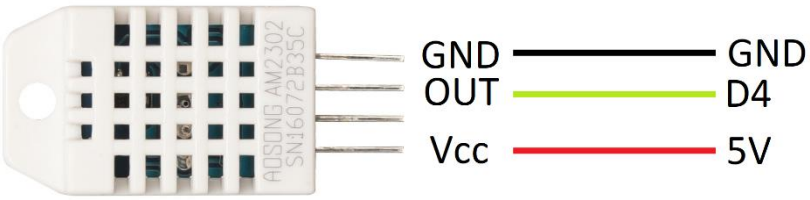


Figura 75. Esquema de connexió del sensor DHT22.

Programació del sensor

Per facilitar l'obtenció de la mesura i els posteriors càlculs s'utilitza la llibreria d'Adafruit *DHT-sensor-library.* A la part de declaracions s'indica el tipus de sensor que s'està utilitzant i el pin on s'ha connectat la sortida. Després, es crea un objecte passant els paràmetres anteriors a una funció de la llibreria.

```
#include  "DHT.h"

#define  DHTTYPE  DHT22  //Pot ser DHT11, DHT21 o DHT22
const  int  DHTPin  =  5 ;  //Pin on s'ha connectat la sortida

DHT  DHT ( DHTPin ,  DHTTYPE ) ;
```

En el `setup()` s'inicialitza el component amb la funció `dht.begin()` on `dht` és el nom que se l'hi ha donat anteriorment. En el `loop()` senzillament s'obtenen els valors de les mesures de temperatura i humitat a través de dues funcions.

```
flotar  h  =  dht.readHumidity ( );
flotar  t  =  dht.readTemperature ( );

if ( isnan ( h )  ||  isnan ( t ) )  {
  Serial.println ("¡Error al llegir desde el sensor DHT!");
}
```

2.3.7 Comandament PS2 *wireless*

En l'actualitat, una de les millors formes de les quals es disposa pel control de robots és mitjançant l'ús d'un comandament tipus PlayStation 2. Gran quantitat de fabricants han creat models clònics, exemple a la Figura 76, i els han adaptat per ser utilitzats amb Arduino i un preu molt baix pel control ofereixen.

Figura 76. Comandament PS2 modificat per funcionar amb Arduino.

Entre les seves característiques com a controlador, el comandament de la PS2 funciona sense fils i disposa d'un rang bastant ampli. Per altra banda, disposa d'una gran varietat d'entrades que incloent 4 botons de direcció, 4 botons d'acció, 4 botons digitals ubicats a la part frontal i 2 palanques analògiques. Tots els botons són sensibles a la pressió. A la Figura 77 es pot veure la ubicació dels diferents botons en el comandament.

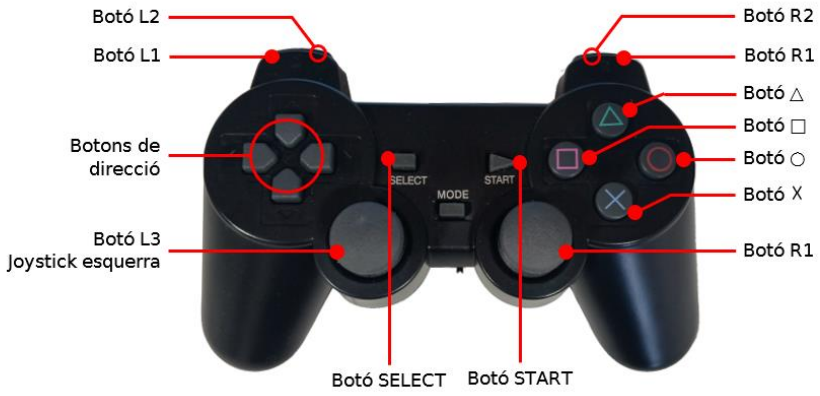


Figura 77. Botons disponibles en el comandament PS2.

Per fer la comunicació amb la placa Arduino s'utilitza el protocol de comunicació SPI (*serial peripheral interface*). A l'hora de programar-lo s'utilitza una llibreria creada per Bill Porter, la qual permet de forma molt senzilla llegir els estats dels botons del comandament.

Protocol de comunicació SPI

El bus SPI va ser desenvolupat per Motorola el 1980. És un protocol de dades en sèrie síncrona que utilitzen els microcontroladors per comunicar-se ràpidament amb un o més dispositius perifèrics a distàncies curtes.

El bus SPI té una arquitectura mestre-esclau, on el mestre pot iniciar la comunicació amb un o diversos dispositius esclaus a la vegada, i enviar o rebre dades d'ells. Hi ha tres línies comunes a tots els dispositius:

- **MISO** (*Master In Slave Out*): la línia per on l'esclau envia dades al mestre.
- **MOSI** (*Master Out Slave In*): la línia per on el mestre envia dades a l'esclau.

- **SCK** (*Serial Clock*): transmet els polsos de rellotge que mantenen a tots els dispositius sincronitzats agafant com a referència el mestre.

I una línia específica per a cada dispositiu:

- **SS** (*Slave Select*): el pin que identifica cada dispositiu i que el mestre pot utilitzar per habilitar o deshabilitar la comunicació amb ell.

En posar la línia SS a un nivell baix, el dispositiu identificat amb aquesta es pot comunicar amb el mestre, i en canviar-la a un nivell alt es talla la comunicació. A la imatge 78 és mostra un esquema de connexió SPI entre un mestre i diversos esclaus.

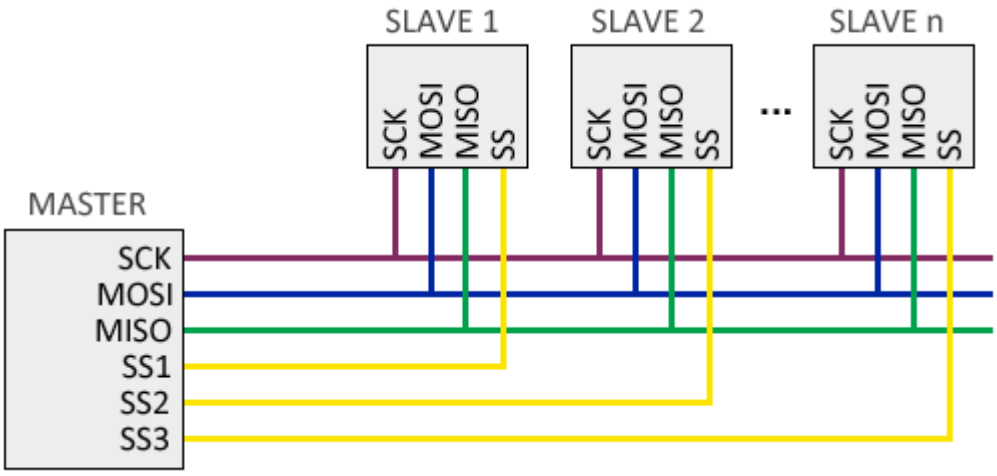


Figura 78. Esquema comunicació SPI.

A la taula 4 és mostra els pins de les línies SPI de la placa MEGA2560:

	MOSI	MISO	SCK	SS (esclau)	SS (mestre)	Voltatge
Mega1280 o Mega2560	51 o ICSP-4	50 o ICSP-1	52 o ICSP-3	53	-	5V

Taula 4. Pins per la comunicació SPI en la placa Arduino Mega.

Si es volgués connectar més d'un dispositiu esclau en Arduino, es pot utilitzar qualsevol altra entrada digital que no sigui la 10. A l'hora de la programació s'ha d'indicar el pin escollit a la llibreria SPI.

Connexió del comandament

Tot el muntatge de cablejat es fa exclusivament en el receptor, així que el comandament únicament s'haurà d'alimentar amb dues piles AAA en sèrie. Per fer la connexió es necessita un convertidor lògic de nivell de 5V a 3.3V, ja que l'electrònica del comandament PS2 funciona a 3.3V. Aquest, ha de permetre adaptar 4 senyals de forma bidireccional. A la Figura 79 es pot veure l'esquema del muntatge.

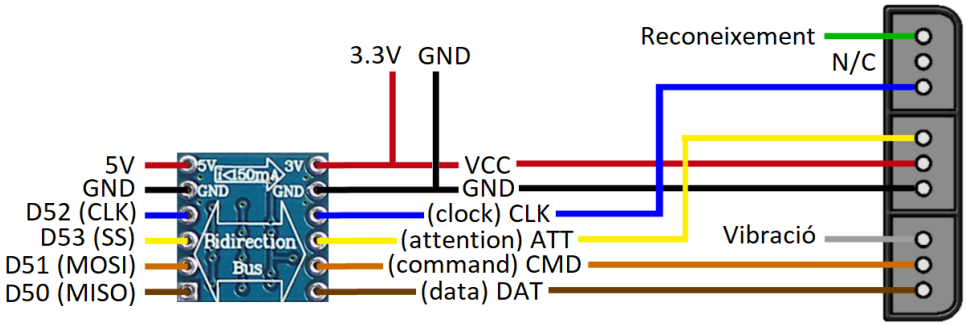


Figura 79. Esquema connexió receptor comandament PS2.

A la Figura 80 es pot veure el muntatge del comandament utilitzant una *protoboard* i un Arduino UNO com a microcontrolador.

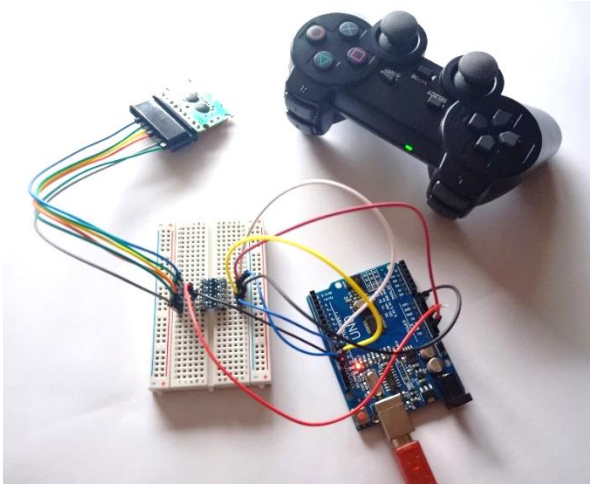


Figura 80. Muntatge de prova comandament PS2.

Programació del comandament

La programació del comandament és molt senzilla i es fa mitjançant la llibreria *PS2X_lib* creada per Bill Porter l'any 2013. A continuació s'expliquen les funcions bàsiques per configurar el comandament i per llegir els estats dels botons.

```
error = ps2x.config_gamepad(13,11,10,12, true, true);
```

on, `config_gamepad(pin clock, pin command, pin attention, pin data, enable pressures, enable rumble).`

Aquesta funció configura els pins per la comunicació SPI i habilita la lectura de la pressió dels botons i la vibració del comandament. La funció retorna un número de 0 a 3 que indica si la inicialització del comandament s'ha completat correctament. A continuació es llisten els significats:

- **`error = 0;`** El comandament s'ha configurat correctament.
- **`error = 1;`** No s'ha trobat el comandament, pot ser que s'hagi cablejat malament o que fallin les piles del controlador.
- **`error = 2;`** S'ha trobat un comandament però no deixa escriure-hi ni llegir.
- **`error = 3;`** El comandament no suporta la vibració.

```
ps2x.read_gamepad(false, vibrate);
```

El primer terme és un booleà que habilita o deshabilita el motor petit del comandament, el segon terme és un número d'entre 0 i 255 que indica la força amb la qual funciona el motor gran, fent així que el comandament vibri. Aquesta funció també habilita el comandament per tal que es pugui obtenir nous valors, ha de ser cridada almenys una vegada cada segon.

Per llegir els valors de les palanques analògiques, així com dels estats dels botons i la pressió exercida en aquests, s'ha creat la Taula 5 on s'especifica el nom què tenen dins

de la llibreria, els valors que poden prendre i el seu significat i marcat amb una X la funció que s'ha d'utilitzar per obtenir-los.

Nom boto llibreria	Característiques	`Button(nom_boto)`	`Analog(nom_boto)`
PSB_START	Polsat = 1, sense polsar = 0.	X	
PSB_SELECT			
PSB_L3			
PSB_R3			
PSB_PAD_UP			
PSB_PAD_RIGHT			
PSB_PAD_DOWN			
PSB_PAD_LEFT			
PSB_L2			
PSB_R2			
PSB_L1			
PSB_R1			
PSB_GREEN			
PSB_RED			
PSB_BLUE			
PSB_PINK			
PSAB_PAD_RIGHT	Pressió amb la que es polsa el boto (0-255).		X
PSAB_PAD_UP			
PSAB_PAD_DOWN			
PSAB_PAD_LEFT			
PSAB_L2			
PSAB_R2			
PSAB_L1			
PSAB_R1			
PSAB_GREEN			
PSAB_RED			
PSAB_BLUE			
PSAB_PINK			
PSS_RX	Lectura palanques analògiques (0-255).		X
PSS_RY			
PSS_LX			
PSS_LY			

Taula 5. Característiques dels botons i palanques del comandament de PS2.

Altres funcions que s'utilitzen durant el programa són les següents:

Funció que retorna *true* si hi ha hagut un canvi d'estat en el botó especificat. Si no s'entra cap paràmetre de botó específic, s'aplica a tots els botons. És a dir, que retorna *true* si hi ha hagut un canvi en qualsevol botó del comandament.

```
ps2x.NewButtonState(optional name_button);
```

Retorna *true* si l'estat del botó indicat passa de *false* a *true*.

```
ps2x.ButtonPressed(name_button);
```

Funció inversa a l'anterior, retorna *true* si l'estat del botó indicat passa de *true* a *false*.

```
ps2x.ButtonReleased(name_button);
```

2.3.8 Comunicació sèrie inalàmbrica

En aquest projecte és indispensable la comunicació del microcontrolador amb l'ordinador, ja que en aquest es troba el programa que s'utilitza per al control i visualització dels estats de Carpher.

La forma més fàcil que té Arduino per passar dades és mitjançant la comunicació sèrie, utilitzada en aquest projecte per a carregar els programes a la placa mitjançant l'IDE d'Arduino. Normalment aquesta transmissió de dades es fa mitjançant un cable què va connectat del microcontrolador a una entrada USB del PC. Aquest fet provoca que el robot hagi d'estar permanentment unit per un cable amb l'ordinador.

El fabricant estatunidenc Polulu, ha posat solució a aquest problema creant un mòdul programable anomenat Wixel (Figura 81).

Figura 81. WIXEL de Polulu

Wixel es basa en el microcontrolador CC2511F32 de Texas Instruments, que té un transceptor de ràdio integrat de 2.4Ghz, 32 KB de memòria flash, 4 KB de RAM i una interfície USB de gran velocitat. Hi ha disponibles un total de 15 línies d'E/S de propòsit general, inclouen 6 entrades analògiques. A la Figura 82 és mostra l'esquema d'entrades i sortides del mòdul proporcionada pel fabricant.

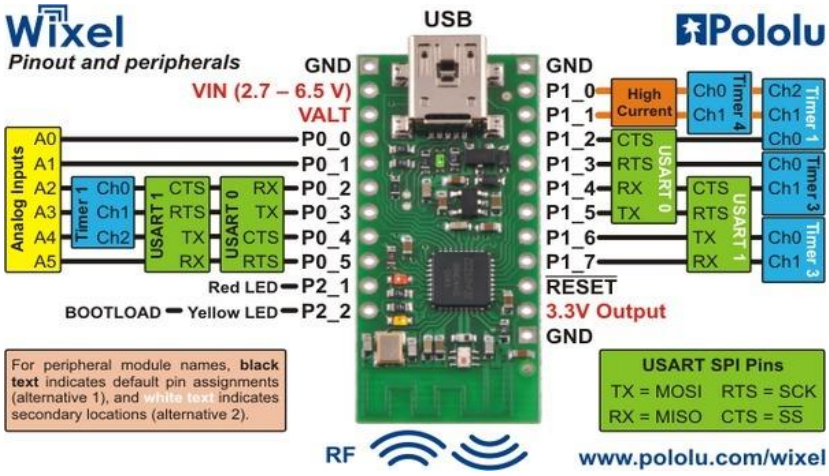


Figura 82. Esquema Wixel de Polulu

Les característiques explicades anteriorment li permeten treballar de tres maneres:

1. Com a controlador principal d'un sistema.
2. Com un adaptador USB a sèrie.
3. Com una comunicació sense fils entre un microcontrolador i un ordinador a en utilitzar dos dispositius Wixel.

En aquest projecte s'utilitzen dos Wixels per aconseguir una comunicació sense fils entre l'Arduino Mega i l'ordinador. Utilitzant una velocitat de bits RF de 350kbps, l'aplicació en sèrie és capaç de transmetre o rebre fins a 10KB de dades per segon i pot arribar a un rang aproximat de 15 metres (en condicions típiques d'interior). També té fins a dos canals que poden treballar simultàniament (USART 0 i USART 1).

El mòdul opera a un voltatge d'entre 2.7 – 6.5 V, i s'ha de tenir en compte que les línies d'E/S no toleren més de 3.3V, pel que s'ha utilitzat el convertidor lògic ja explicat anteriorment. El consum màxim de corrent és de 30 mA.

Protocol de comunicació sèrie

Els ports sèrie són la forma principal de comunicar una placa Arduino amb un ordinador. Un port es el nom genèric que se li dóna a les interfícies, físiques o virtuals, que permeten la comunicació entre dos ordinadors o dispositius.

Un port sèrie envia la informació mitjançant una seqüència de bits. Per fer-ho es necessiten dos connectors; RX (recepció) i TX (transmissió).

Sovint als ports sèrie se'ls denomina UART (*universally asynchronous receiver/transmitter*), una unitat que incorporen alguns processadors i que és l'encarregada de realitzar la conversió de les dades a una seqüència de bits i transmetre'ls o rebre'ls a una velocitat determinada.

Quan es parla del terme TTL (*transistor-transistor logic*), significa que la comunicació es realitza mitjançant variacions en el senyal entre 0V i Vcc (on Vcc sol ser 3.3V o5V).

Pràcticament totes les plaques d'Arduino disposen almenys d'una unitat UART. L'Arduino Mega2560 disposa de fins a 4 unitats UART TTL 0V/5V, i la localització de Rx i Tx per cada canal es poden veure en la taula 100.

BOARD	SERIAL PINS	SERIAL1 PINS	SERIAL2 PINS	SERIAL3 PINS
Mega2560	0(RX), 1(TX)	19(RX), 18(TX)	17(RX), 16(TX)	15(RX), 14(TX)

La placa Mega, està equipada amb un USB que utilitza el canal 1 (pins 0 i 1) per tal de comunicar-se amb l'ordinador.

Com programar els mòduls com a port sèrie sense fils

Pololu proporciona un programa per configurar el mòdul, aquest es troba a la pàgina web del Wixel en la pestanya de recursos i està disponible per sistemes operatius com Windows, Linux o Mas OS X. Es troba en forma de *zip* amb el nom *Wixel Windows Drivers and Software.* A la Figura 83 es pot veure una captura de pantalla on es ressalta l'enllaç per descarregar el programa.

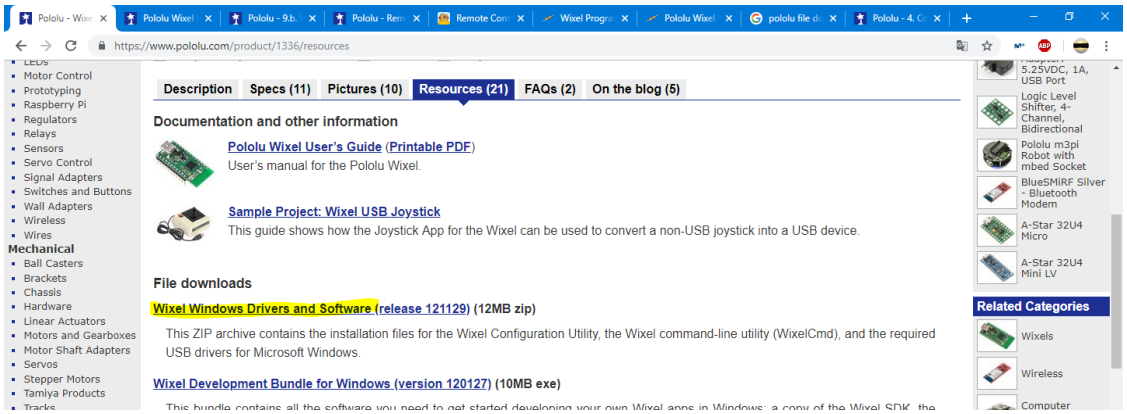


Figura 83. Captura de la pàgina web de Polulu on és troba tota la informació i programes del Wixel.

Un cop descomprimit el zip, s'ha d'executar el fitxer *setup.exe*. Un cop finalitzada la instal·lació, accedim a l'aplicació a través de *C:\Program Files (x86)\Pololu\Wixel\bin\WixelConfig.* Aquesta ens permet carregar aplicacions al Wixel. Connectem un dels Wixels a l'ordinador mitjançant USB. A la Figura 84 és mostra una captura del programa quan s'obre per primera vegada i quan s'ha connectat un mòdul.

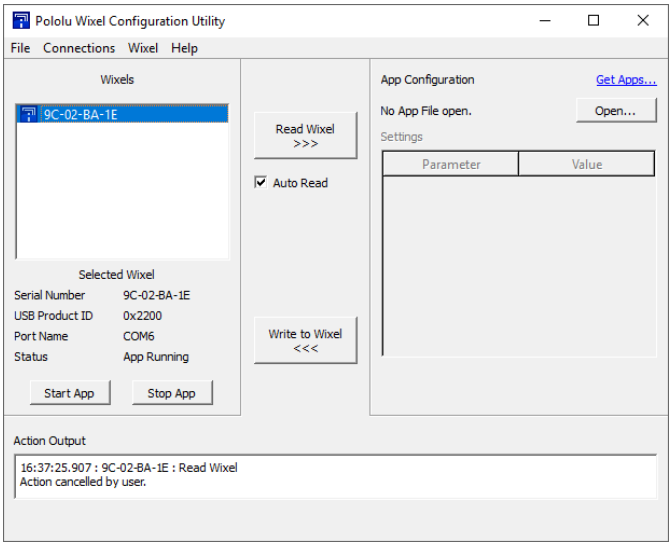


Figura 84. Programa Polulu Wixel Configuration Utility.

A la part d'*Action Output* es pot veure indicat que actualment aquest mòdul no té cap aplicació instal·lada (venen així per defecte de fàbrica).

Polulu ja ha creat una aplicació per tal que el mòdul funcioni com a port sèrie sense fils, i aquesta s'ha de descarregar de la seva pàgina web. El seu nom és *Wireless Serial App (v.1.3).*

Per escriure-la en el mòdul, primer s'ha de carregar en el programa *Polulu Wixel Configuration Utility*. Per fer-ho s'ha d'anar a la pestanya *File* i triar l'opció *Open App...* Es selecciona l'arxiu que s'ha acabat de descarregar i s'obre. La Figura 85 mostra una captura de pantalla mostra el resultat de carregar l'aplicació Wireless Serial App a l'aplicació de Polulu.

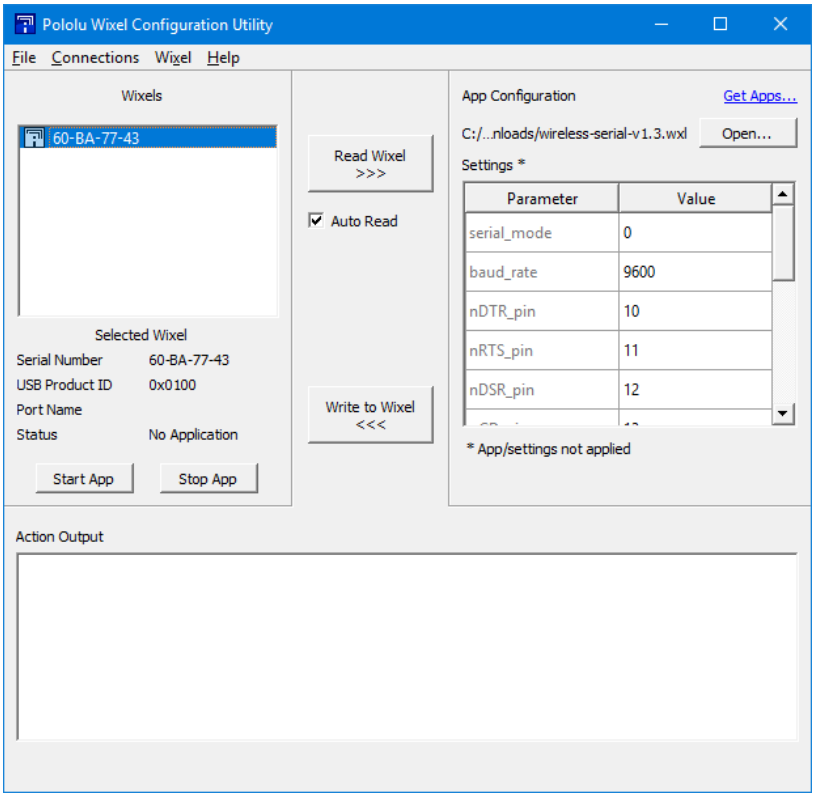


Figura 85. Programa Polulu Wixel un cop s'ha connectat el mòdul i s'ha carregat el programa.

En el quadre d'*App Configuration* es poden veure i modificar els diferents paràmetres de l'aplicació. El valor de *baud rate* es canvia a 115200, ja que és la velocitat d'escriptura

(kbps) per defecte del port sèrie en carregar un programa a la placa Mega2560. El valor *radio channel* no es canvia, però s'ha d'assegurar que és el mateix en els dos Wixels.

Després d'escollir l'aplicació i havent seleccionat el mòdul, es pot escriure l'aplicació al Wixel fent clic al botó *Write to Wixel*. Això esborra qualsevol aplicació que estigués anteriorment al Wixel i hi escriu la nova. Tot seguit es reinicia el mòdul i s'inicia amb la nova aplicació. La Figura 86 mostra un captura del resultat un cop escrit el programa.

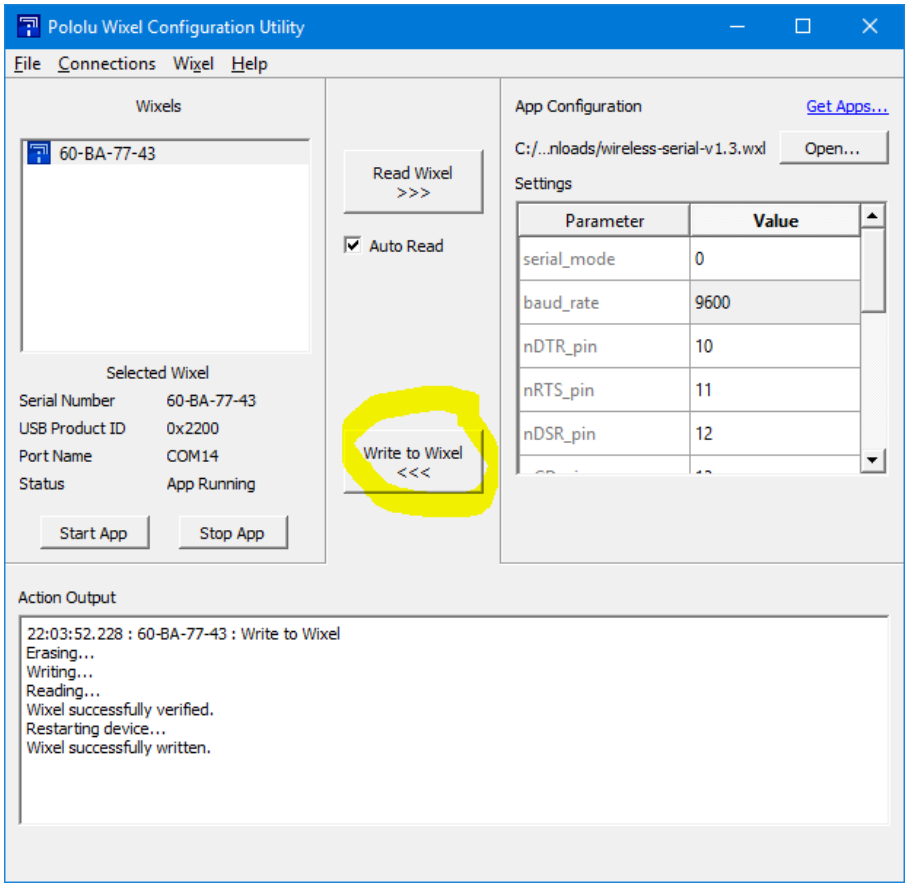


Figura 86. Programa Polulu Wixel un cop s'ha gravat l'aplicació al mòdul.

Si ens fixem amb el color de la icona del costat de l'adreça del Wixel, podem veure que ha canviat, això ens indica que s'ha carregat correctament i sense problemes. També es pot veure a la finestra d'*Action Output.*

Connexió dels mòduls

Per tal de poder carregar sense fils el programa des de l'ordinador a la placa, es necessita reiniciar l'Arduino durant l'enviament d'aquest, si no indicarà un error de sincronització.

Polulu ja ha solucionat aquest problema en el *shield* que ha creat per poder instal·lar de forma còmoda el Wixel sobre l'Arduino, a la Figura 87 es pot veure el *shield* instal·lat i funcionant.

Figura 87. Wixel shield per Arduino de la marca Polulu.

La marca proporciona en l'esquema elèctric, Figura 88, la circuiteria implementada entre el pin P0_0 i el pin de *reset* del microcontrolador per tal de fer el reinici de la placa de forma automàtica en detectar que es vol carregar un nou programa.

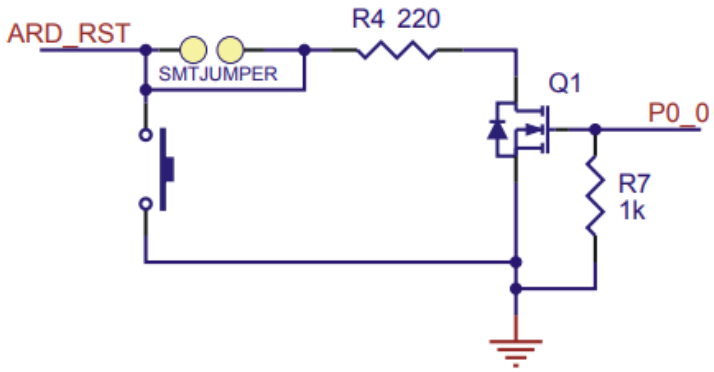


Figura 88. Circuit proporcionat per Polulu per reiniciar la placa automàticament.

En cas de no voler implementar el circuit anterior, es pot reiniciar el microcontrolador de forma manual mitjançant el polsador de la placa un cop ha compilat el programa i comença a pujar-lo.

L'esquema final utilitzat durant el projecte és el què es mostra en la Figura 89. El primer mòdul es connecta a l'ordinador mitjançant mini-USB. En el segon es connecta el Tx (D1) de l'Arduino al Rx (P1_6) del Wixel mitjançant un convertidor de voltatge de 5v a 3.3V, ja que els pins del Wixel funcionen a 3.3V. El pin Rx (D0) de l'Arduino va connectat directament al pin Tx (P1_7) del Wixel. Com ja s'ha vist anteriorment, per tal de reiniciar la placa automàticament en pujar un *sketch*, s'utilitza un Mosfet BSS138 (50V 0.2A).

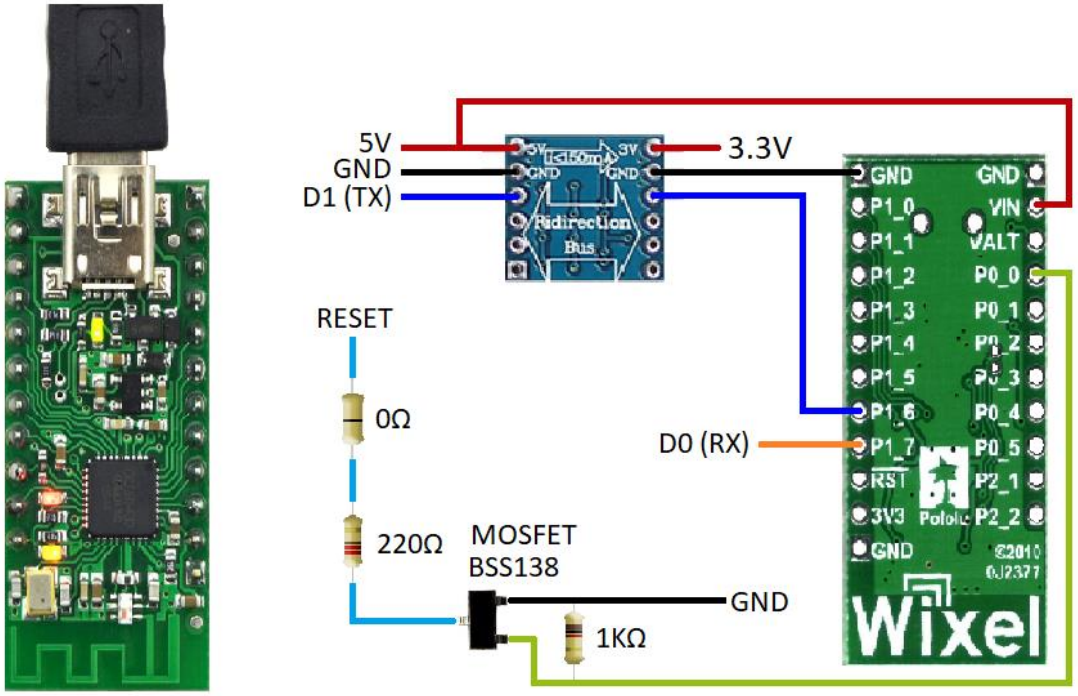


Figura 89. Esquema de muntatge dels Wixels

A la Figura 90 es pot veure la connexió entre el Wixel i el microcontrolador utilitzada en la construcció de Carpher.

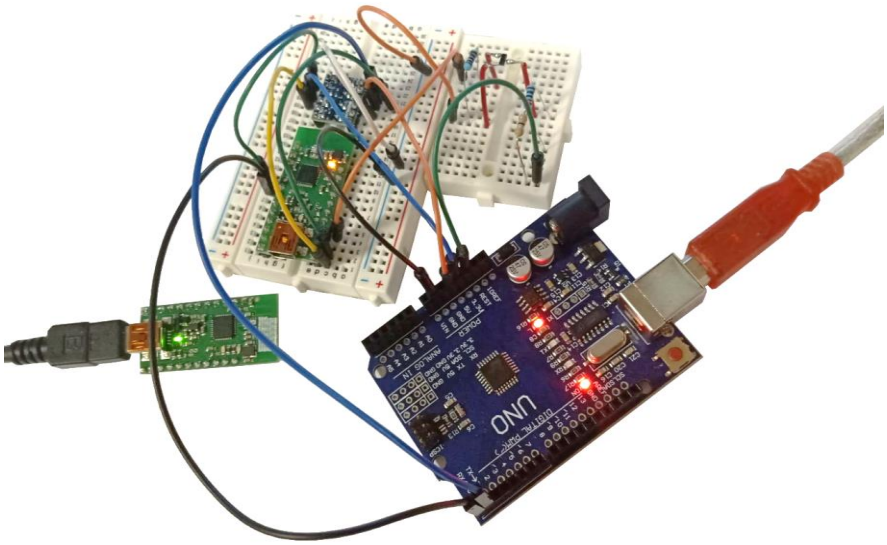


Figura 90. Muntatge de prova Wixels.

Programació del mòdul

Simplement s'ha de declarar a la part del *setup()* el pin Rx de l'Arduino com a entrada i obrir el port sèrie amb una velocitat de transmissió de 115200 kbps, velocitat amb la qual és càrrega el programa a la placa d'Arduino MEGA2560.

```
 void setup() {
  Serial.begin(115200);
  pinMode(rf_pin, INPUT);
}
```

Per enviar i rebre dades pel port sèrie es fa amb les mateixes instruccions que quan tenim el microcontrolador directament connectat amb l'ordinador. A les següents línies de codi es mostren les instruccions més utilitzades. Per enviar dades al port sèrie s'utilitza `Serial.print(...)`, el missatge de dins dels parèntesis és el què s'envia.

```
Serial.print("Hola"); //Escriu la cadena "Hola" al port sèrie com
                        a caràcters ASCII.
```

```
Serial.println("Adeu"); //Escriu la cadena "Adeu" al port sèrie com
                        a caràcters ASCII seguida d'un salt de línia.
```

Per rebre dades s'utilitza `Serial.available()`, que guarda el nombre de bytes que estan en el bus de dades esperant per ser llegits. Després, es guarden en una variable mitjançant `Serial.read()`. En el següent codi és mostra un exemple en què, en rebre un caràcter entre 0 i 9, s'escriu en el port sèrie que aquest s'ha rebut.

```
if (Serial.available()) {
   char data = Serial.read();
   if (data >= '0' && data <= '9') {
     Serial.println("S'ha enviat un número");
   }
 }
```

2.3.9 Tira de LEDs

Quan l'usuari interactua directament amb Carpher, aquest mostra el seu estat mitjançant LEDs. S'han implementat fins a 8 LEDs, un per cada sensor de distància VL53L0X, dos a la part posterior i l'últim per indicar l'estat de la bateria. Aquests indiquen si s'ha detectat un obstacle i quin sensor ho ha fet, si el robot s'aturà o gira, si està esperant ordres, etc.

S'ha utilitzat una tira de LEDs direccionals del tipus WS2812B (Figura 91), aquests poden ser controlats un a un per Arduino i són RGB, el què significa que els LEDs vermell, verd i blau s'integren en un mateix xip de controlador (Figura 92) en un petit paquet de muntatge superficial controlat a través d'un sol cable.

Figura 91. Tira de LEDs RGB.

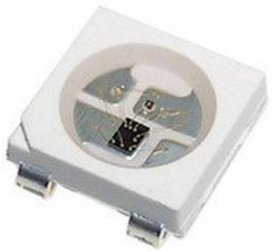


Figura 92. LED RGB 5050

Cada LED disposa d'un integrat que guarda 3 bytes (24 bits), que corresponen als tres colors del RGB. Això significa que quan es vol encendre un LED s'han d'enviar aquests 3 bytes d'informació, on cada byte tindrà un valor d'entre 0 i 255 que serà la intensitat del color. Per tant es poden arribar a reproduir fins a 16,7 milions de colors. A la Taula 6 es mostren els colors més utilitzats en els LEDs RGB.

Color		R	G	B
	Vermell	255	0	0
	Verd	0	255	0
	Blau	0	0	255
	Groc	255	255	0
	Rosa	255	0	255
	Cian	0	255	255
	Blanc	255	255	255

Taula 6. Colors primaris i colors bàsics combinats d'aquests expressats en codi RGB.

Quan un LED rep el flux de bytes, guarda els últims bytes rebuts i transmet els que contenia al següent LED. Finalment, amb un senyal de *"resetcode*", cada LED mostra l'últim valor guardat.

Connexió

La connexió (Figura 93) és bastant senzilla, només es necessita un sol pin PWM de l'Arduino i connectar-lo al pin Din de la tira de LEDs. Entre aquests, s'ha d'intercalar una resistència de 470 Ohm per evitar possibles danys als primers LEDs.

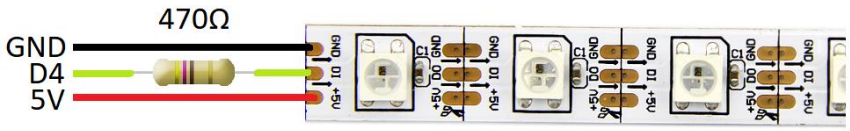


Figura 93. Esquema connexió tira LEDs RGB.

S'ha de tenir en compte que cada LED pot consumir fins a 60 mA (20 mA per cada un dels tres LEDs que formen un RGB), tenint en compte que se n'han implementat 8, el consum que es pot produir en posar-los tots en blanc és de 480 mA. Arduino només pot entregar pel pin 5V fins a 300 mA, per tant s'ha recorregut a una font externa capaç de lliurar aquest corrent.

Per tal d'evitar equipar el robot mòbil amb més d'una bateria, s'ha utilitzat la sortida de 5 V que té el mòdul L298N, aquest pot entregar fins a 1A però es recomana no passar dels 600 mA, ja que pot afectar el rendiment.

A la Figura 94 es pot veure el circuit muntat utilitzant de font d'alimentació dues piles en sèrie de 3,6 V, que alimenten tant el mòdul L298N com el microcontrolador.

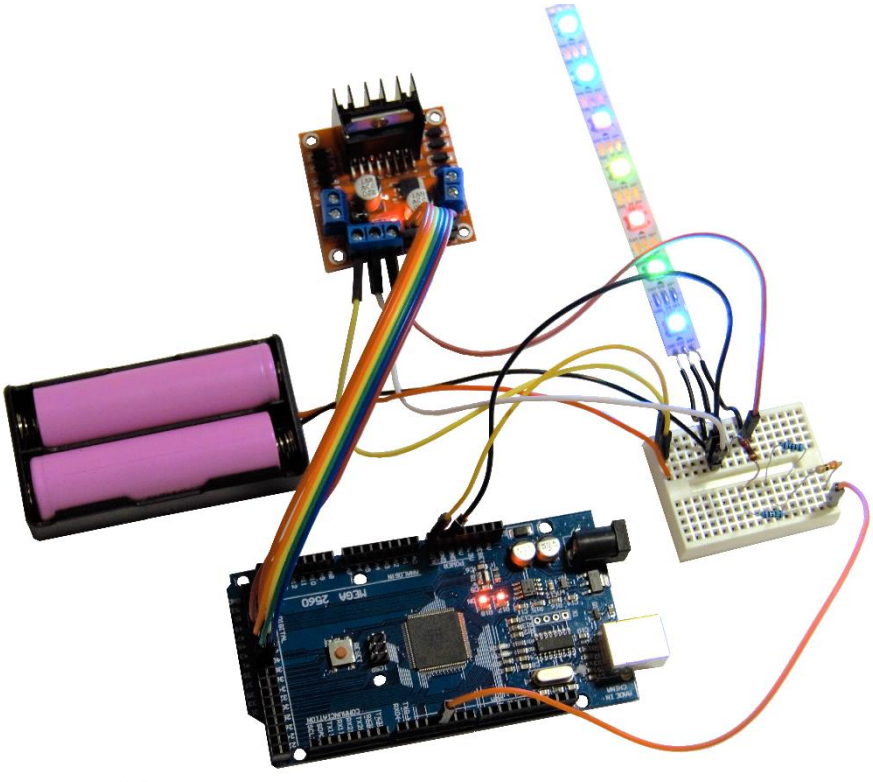


Figura 94. Muntatge de prova tira de LEDs RGB.

Programació

Per controlar la tira de LEDs existeixen diverses llibreries que faciliten molt la feina, la més coneguda i la que s'utilitza en aquest projecte és la llibreria NeoPixel d'Adafruit, gratuïta a GitHub. Un cop instal·lada la incloem al programa i seguidament, amb la funció que és mostra a continuació, indiquem el número de LEDs a controlar, el pin on està connectada la tira, l'ordre dels bytes del LED i finalment la freqüència d'actualització.

```
#include <Adafruit_NeoPixel.h>
#define PIN            4
#define NUMPIXELS      7
Adafruit_NeoPixel pixels = Adafruit_NeoPixel(NUMPIXELS, PIN, NEO_GRB
                                             + NEO_KHZ800);
```

A la part del `setup()` s'utilitza la funció `begin()` per tal d'inicialitzar el pin i posar-lo en un estat inicial LOW.

```
pixels.begin();
```

Per fer el control dels diferents LEDs s'utilitza la funció `setPixelColor(...)`, on s'indica el LED que s'aplicarà i a través de la funció `Color(...)`, el valor dels tres bytes que conformen cada un dels colors RGB. Un cop s'han indicat els diferents colors pels LEDs, s'utilitza la funció `Show()` per mostrar l'últim valor guardat de cada LED.

```
pixels.setPixelColor(num_led, pixels.Color(255, 255, 255));
pixels.show();
```

2.3.10 Bateria

Les plaques Arduino estan dissenyades per ser alimentades de diverses maneres. La més usual és mitjançant 5 volts al port USB, extrets de l'ordinador o d'una bateria portàtil (és la utilitzada per a carregar els mòbils). D'altra banda es pot optar pel connector jack, connectant un adaptador de corrent (AC/DC) directament des d'un endoll. Si es vol optar per un sistema més autònom, com és en el cas de Carpher, s'implementen bateries que proporcionin un voltatge d'entre 6 V i 12 V i es connecten en el pin Vin i GND del microcontrolador. A la Figura 95 es pot veure el resum d'alimentació.

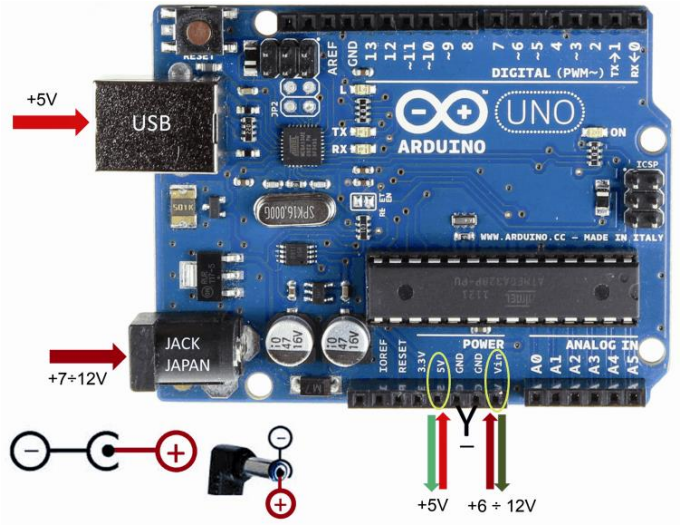


Figura 95. Resum de les diferents entrades per alimentar l'Arduino.

Pel funcionament de Carpher es necessita una bateria capaç d'alimentar tots els components del sistema. Tot i que la majoria seran alimentats a través d'Arduino, el mòdul L298N necessita rebre un voltatge superior al que aquest pot subministrar. Per altra banda, la tira de LEDs pot arribar a consumir 600mA i Arduino pot subministrar un màxim de 200mA.

En consideració a les necessitats anteriors, s'han optat per la instal·lació de bateries de liti 18650 de la marca Samsung. Aquestes proporcionen 3,6V així que, utilitzant dues bateries en sèrie, es proporciona un voltatge total de 7,2V sent aquest perfecte per alimentar tant l'Arduino com el mòdul L298N.

A la Taula 7 es pot veure el voltatge d'alimentació de cada component del sistema i el voltatge final al qual s'alimentarà.

	Voltatge d'alimentació (V)	Voltatge al qual s'alimenta (V)
Arduino Mega	7 - 12	7.2
Controlador motors L298N	6 - 12	7.2
Sensor ultrasonsHC-SR04	5	5
Sensor làser VL53L0X	3 - 5	5
Encoder LM393	3.3 - 12	5
Giroscopi MPU-6050	5	5
Sensor temperatura DHT22	3 - 6	5
Altaveu	5	5
Comandament PS2	3.3	3.3
Wixel Polulu	2.7 - 6.5	3.3
Led RGB	5	5

Taula 7. Resum voltatge d'alimentació de cada component utilitzat en el projecte.

A la Figura 96 és pot veure un resum de com s'alimenta el sistema. Pels components que s'utilitzen més d'una vegada, com serien els sensors làser, només se'n posa un perquè tots s'alimenten d'igual forma.

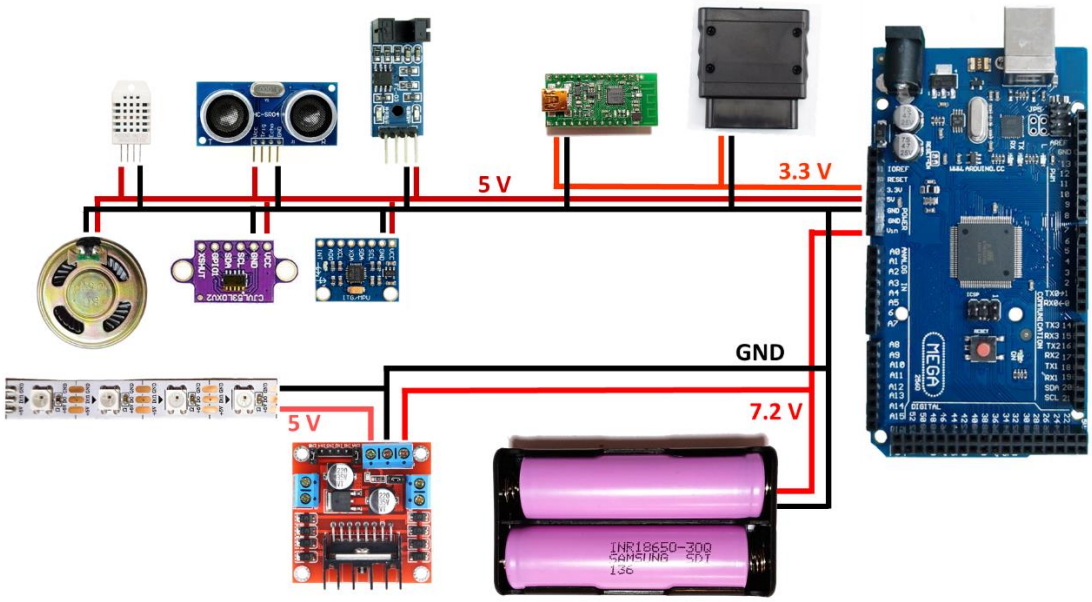


Figura 96. Esquema d'alimentació del sistema

Les piles, en ser recarregables, disposen d'una capacitat de càrrega més gran que les què no ho són. I a la vegada, es poden tornar a fer servir evitant així la contaminació què produeixen les piles d'un sol ús. Tenen una capacitat de 3000mAh i poden alliberar fins a 10A. A la Taula 8 es pot veure el voltatge al qual han de ser alimentades i el seu consum màxim, així com la suma final del consum total del sistema.

	Consum màxim (mA)	**Unitats utilitzades**	**Consum màxim total (mA)**
Arduino Mega	93	1	93
Controlador motors L298N	36	1	36
Sensor ultrasonsHC-SR04	15	1	15
Sensor làser VL53L0X	3,9	6	23,4
Encoder LM393	0,4	2	0,8
Giroscopi MPU-9250	3,7	1	3,7
Sensor temperatura DHT22	1,5	1	1,5
Altaveu	1	1	1
Comandament PS2	15	1	15
Wixel Polulu	30	1	30
Led RGB	60	8	480
	TOTAL:		699,2 mA

Taula 8. Consum dels diferents components del sistema.

Una capacitat de 3000mAh indica que si el nostre sistema consumis 3A de forma constant, la bateria seria capaç d'alimentar el robot durant una hora. A la taula anterior es pot veure que el consum màxim és aproximadament de 700 mA, per tant, es tarda unes 4 hores i 17 minuts a descarregar per complet les bateries. Cal dir que, el consum màxim molt difícilment es produirà, ja que el sistema no funciona mai a màxima potència.

2.4 Disseny del xassís

Un xassís és indispensable en un robot mòbil, ja que és l'estructura on es recolzen cadascun dels components escollits anteriorment. S'ha optat per un disseny exterior minimalista de metacrilat. I quant a dimensions s'han procurat les menors possibles, ja que Carpher ha de ser capaç d'accedir a espais petits i estrets.

Per la creació de les peces s'ha utilitzat el mètode de tall làser (Figura 97), tècnica que consisteix en el tallat o gravat de material mitjançant un làser d'alta precisió. Es treballa amb planxes de fins a 200 mm de grossor i de materials com plàstic, fusta, cartó, etc. Es caracteritza per ser un tall ràpid i sense residus, ja que assoleix una elevada temperatura que vaporitza total o parcialment el material sobrant. Com la zona afectada per la calor és petita (al voltant de 0,5 mm), les peces tallades presenten una deformació mínima i unes cantonades esmolades i netes.

El material escollit ha estat el metacrilat (Figura 98), també conegut com a PMMA. Aquest és un termoplàstic amorf, dur i rígid davant de les ratllades però a la vegada fàcil de tallar i modelar. Té una bona resistència a tota classe de condicions exteriors normals, en especial als raigs ultraviolats. S'utilitza bàsicament per mampares divisòries, elements de decoració, senyalització, expositors, caixes, etc.

Figura 97. Mètode del tall làser per crear peces en forma d'engranatge a partir d'una planxa de fusta.

Figura 98. Planxes de metacrilat

Pel disseny del xassís s'ha utilitzat el software de SolidWorks (Figura 99 i 100) de la companyia Dassault Systèmes. Es un software de CAD (disseny assistit per computadora) 3D que permet modelar peces i acoblaments en 3D i plànols en 2D.

Figura 99. Logotip del software SolidWorks.

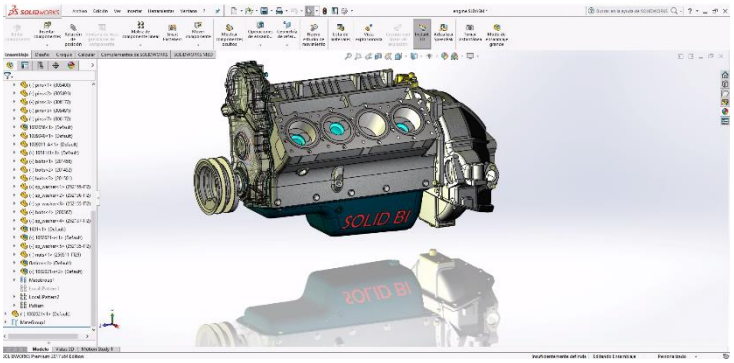


Figura 100. Interfície d'usuari de SolidWorks on és mostra el disseny d'un motor en 3D.

Per enviar les peces a la talladora làser es necessita un model en 2D d'aquestes i situar-les de forma eficient dins d'un requadre què simula les dimensions de la planxa què es tallarà. Per dur a terme aquest procés s'ha utilitzat el programa DraftSight (Figura 101 i 102), un software CAD 2D desenvolupat per Dassault Systèmes, igual que SolidWorks. Aquest permet crear, visualitzar i editar arxius DWG (*drawing*) i DXF.

Figura 101. Logotip del software DraftSight.

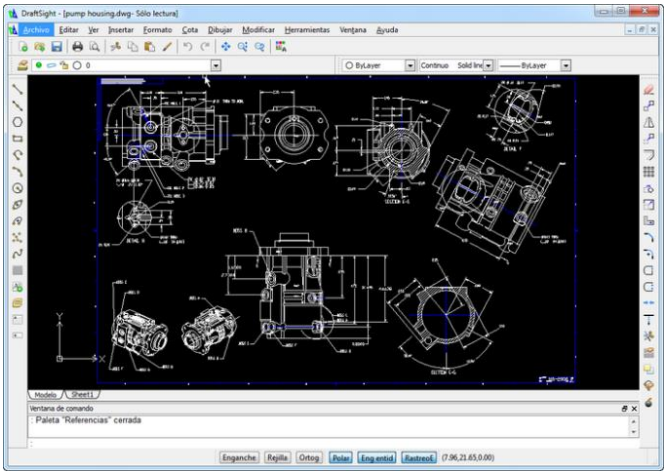


Figura 102. Interfície d'usuari de DraftSight.

2.4.1 Distribució dels components i disseny de les parts del xàssis

En aquest apartat s'explica com s'ha fet el disseny del robot tenint en compte aspectes com la millor ubicació per cada component o el centre de masses del robot. Primer s'explica el disseny fet en 3D utilitzant SolidWorks i, posteriorment, s'explica la transformació d'aquest en 2D utilitzant DraftSight per a poder enviar les peces a la talladora làser.

2.4.1.1 Disseny 3D utilitzant SolidWorks

S'ha de tenir en compte que el disseny de les diferents parts del xassís s'han anat modificant a la vegada que s'introduïa un nou component, tot i això, en aquest apartat només es reflecteix el resultat final. Per exemple, el disseny de la part principal del xassís, la base on es recolzen la resta de peces i components, ha estat la primera peça creada (Figura 103), però si és mira la transformació de la peça en el disseny final es poden veure canvis significatius sobretot en l'interior de la mateixa, on s'han obert diversos forats per encaixar-hi altres peces o components i espai per passar cables.

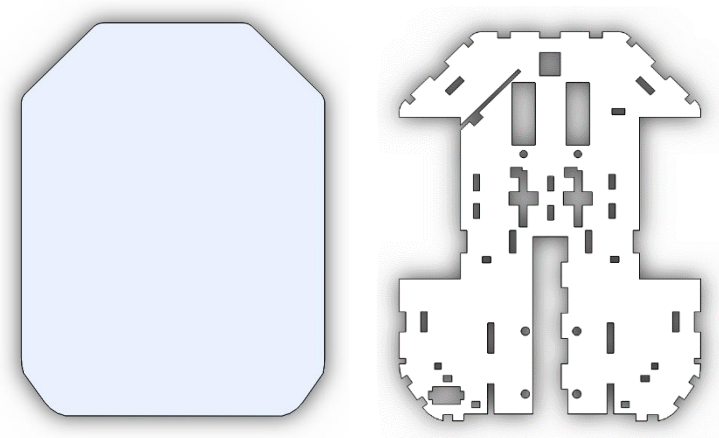


Figura 103. Disseny de la plataforma principal. A la esquerra la forma inicial i a la dreta la final, després d'haver dissenyat la resta del robot.

Com ja s'ha dit, aquesta és la part més important del xassís, ja que fa de suport a tota la resta. El disseny s'ha fet partint de les plataformes que es comercialitzen (Figura 104) en *kits* de cotxes preparats per Arduino. La base final que s'ha dissenyat es pot veure a la Figura 105, és molt semblant a les comercialitzades però s'ha preparat per tal de poder crear un tancament al voltant d'aquesta per protegir els components.

Figura 104. Kit cotxe robot per Arduino comercialitzat per la marca LAFVIN.

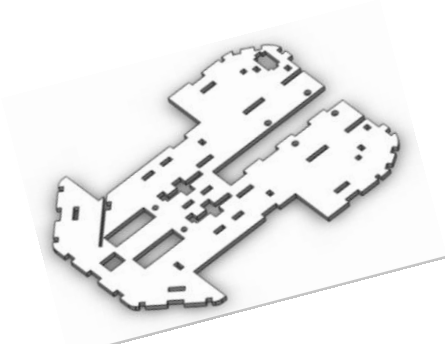


Figura 105. Disseny de la base final.

En ser un 2WD s'ha col·locat cada un dels motors en un dels laterals del xassís, també s'ha col·locat una roda boja a la part posterior, a una quarta part de la llargada del robot, tot per mantenir l'estabilitat d'aquest.

Tal com es mostra a la Figura 106, per col·locar els codificadors s'ha obert un forat a la base per encaixar l'optointerruptor, i un altre per als pins del component. Perquè quedi més subjecte, s'ha utilitzat un caragol que l'entravessa. A partir d'aquest punt, les peces que s'han afegit al disseny, es mostren a les imatges pintades en color blau.

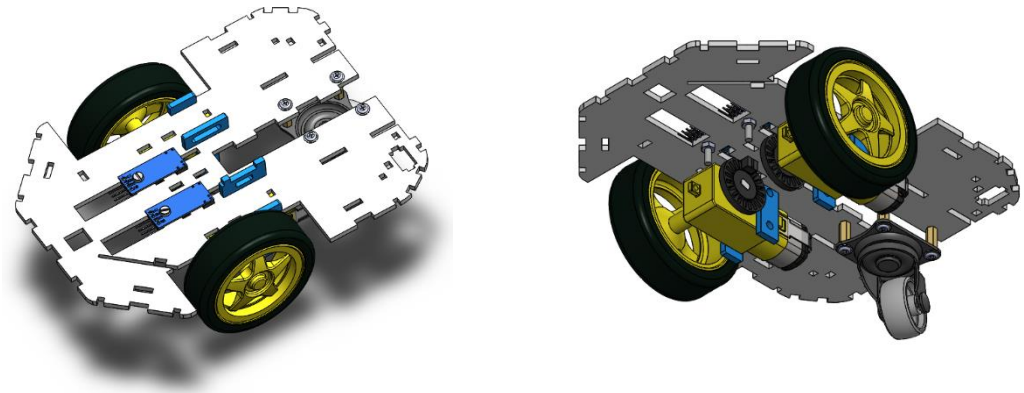


Figura 106. En blau és ressalten les peces que faran de suport als motors.

Pels sensors làsers s'ha dissenyat un tancament de protecció (Figura 107) amb la part frontal de metacrilat transparent per evitar interferències amb el camí del feix de llum. La part posterior és oberta per tal que els pins es puguin manipular des de fora i s'han afegit dos forats per poder assegurar el sensor a la posició mitjançant caragols.

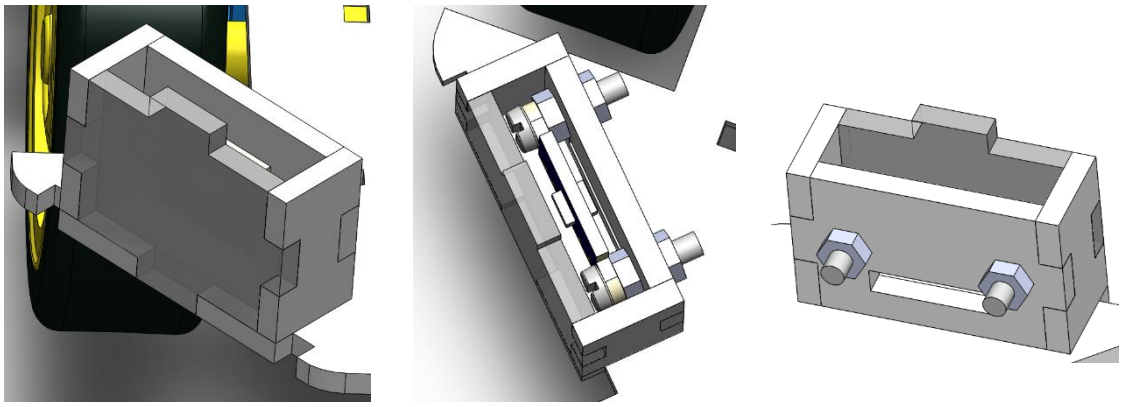


Figura 107. Coberta de protecció dels sensors làser.

Com és mostra a la Figura 108, s'ha equipat a Carpher amb sis sensors làser, un al capdavant de la base, dos a 45º a ambdós costats del primer i dos més als laterals de la part posterior del robot. L'últim sensor està situat a la tapa superior per tal de detectar l'altura del sostre, sempre que aquest estigui dins del rang del sensor.

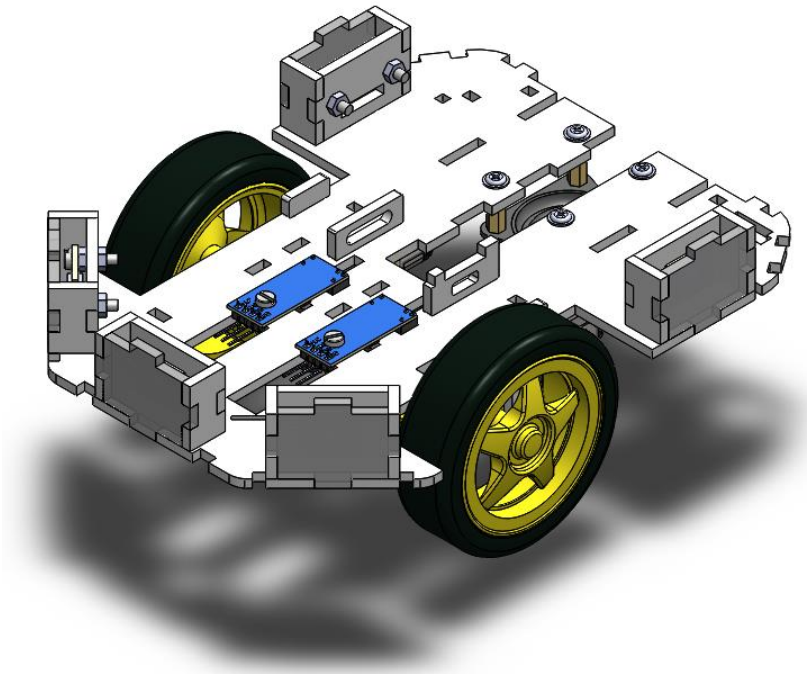


Figura 108. Posicionament dels tancaments sobre la base del robot.

Per tal d'encabir tots els components dins del robot s'ha dissenyat una petita plataforma (senyalitzada en blau a la Figura 109) preparada per encaixar el Wixel de Polulu, el giroscopi i el L298N. El giroscopi s'ha col·locat sobre l'eix central del robot (Figura 110), de forma que l'eix Y de mesura del sensor quedi situat sobre l'eix Y del robot.

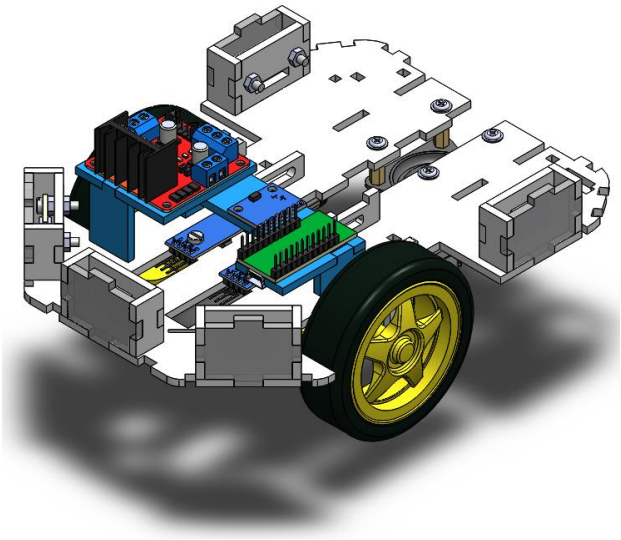


Figura 109. En blau la plataforma on se situen el wixel, l'MPU-6050 i el L298N.

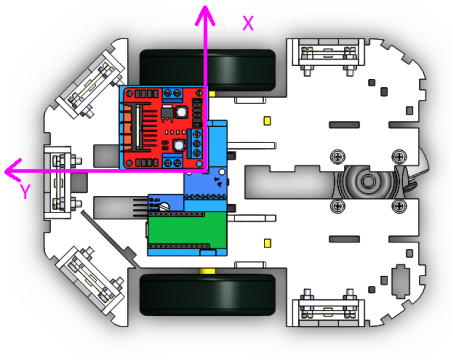


Figura 110. Eixos de referència MPU-6050.

El receptor del comandament de la PS2, per tal que ocupi menys espai, se l'ha tret de la carcassa deixant la PCB a l'aire. A la base s'ha fet un forat on el tancament dels pins serveix de suport i s'ha dissenyat una peça per tal d'assegurar que aquest queda subjecte al robot (Figura 111).

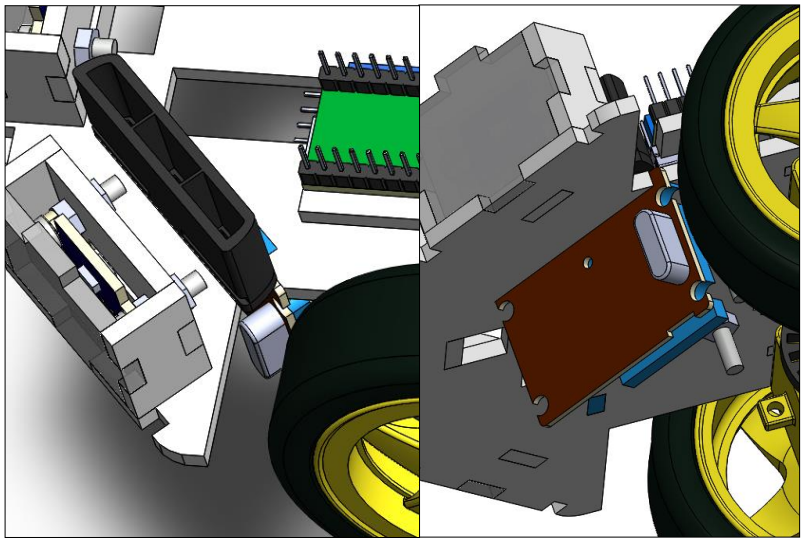


Figura 111. Suport i col·locació del receptor del comandament PS2.

El porta piles s'ha col·locat a la part posterior del robot, d'aquesta manera el seu pes és suportat per la roda boja. Si pel contrari, s'hagués posat a la part frontal, el robot podria tenir problemes d'estabilitat en frenar, fent que aquest tendís a bolcar-se endavant.

Tenint en compte que el robot està tancat pels laterals, s'ha dissenyat una plataforma capaç de lliscar (Figura 113), de manera que quan s'acabin les bateries i aquestes s'hagin de carregar, sigui fàcil accedir a elles. Per obrir aquesta plataforma s'ha instal·lat un sistema *push to open* (Figura 112), actualment molt comercialitzat en armaris i que, en pressionar la fusta cap dins, el sistema actua com a motlle i fa que la plataforma es desencaixi i surti per si mateixa.

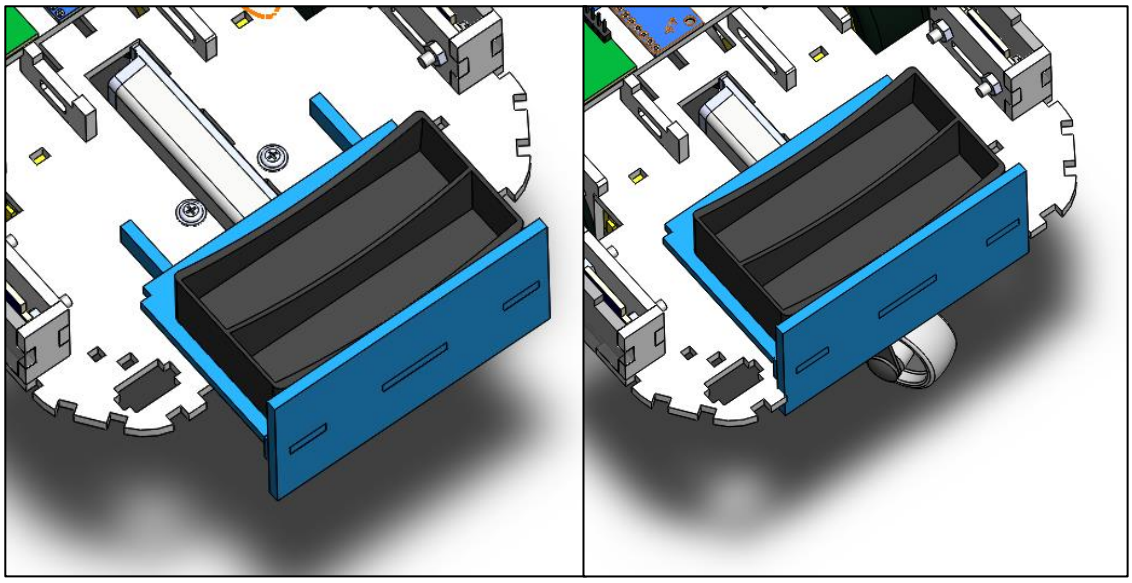


Figura 112. Lliscament de la plataforma de la bateria.

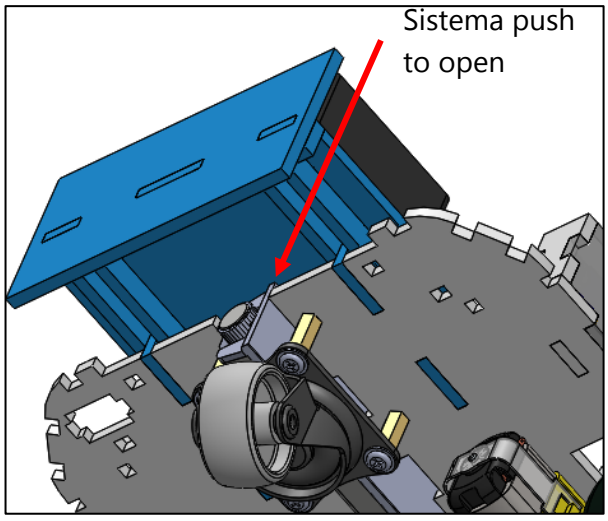


Figura 113. Sistema push to open utilitzat per fer lliscar la plataforma de la beteria.

Tal com és mostra la Figura 114, el botó per encendre i apagar el robot no és visible a simple vista però a la vegada s'ha procurat que sigui fàcil d'accedir, per tant s'ha col·locat a la part inferior de la base.

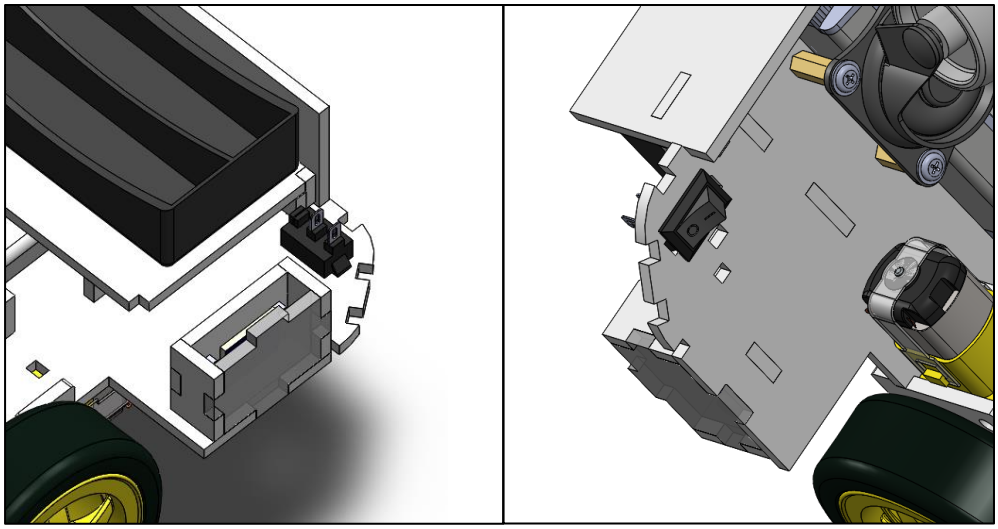


Figura 114. Posicionament botó per apagar i encendre el microcontrolador.

La placa Arduino està situada sobre la bateria i està subjecte per una plataforma (Figura 115) i assegurada per dos cargols.

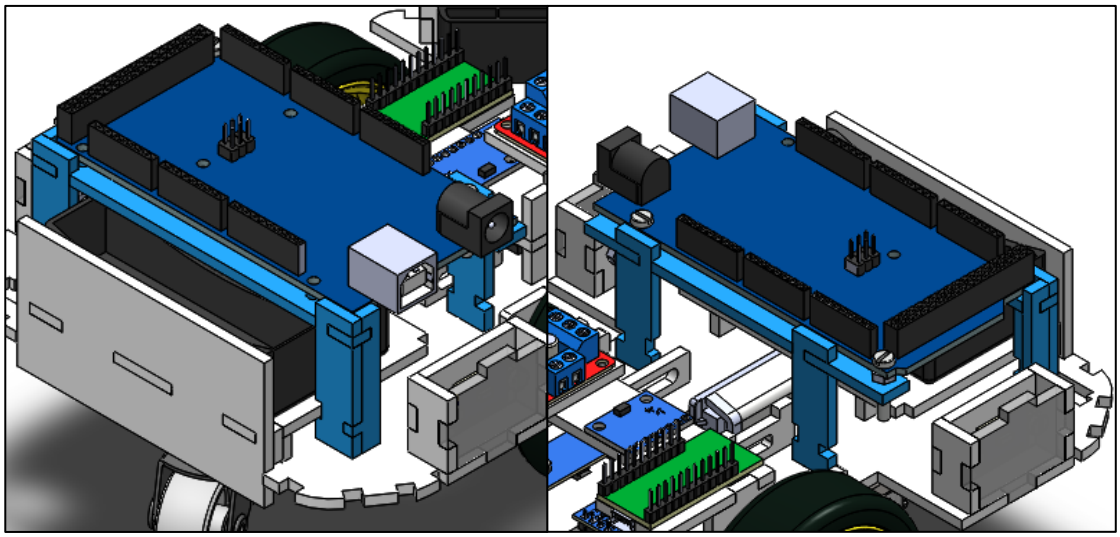


Figura 115. En blau la plataforma dissenyada per col·locar la placa d'Arduino MEGA2560.

Tot seguit s'han col·locat les peces laterals per tancar el robot (Figura 116), tenint en compte que s'ha de deixar un forat per tal de poder connectar l'entrada USB de la placa d'Arduino i dos forats més a la part frontal per encaixar el sensor d'ultrasons.

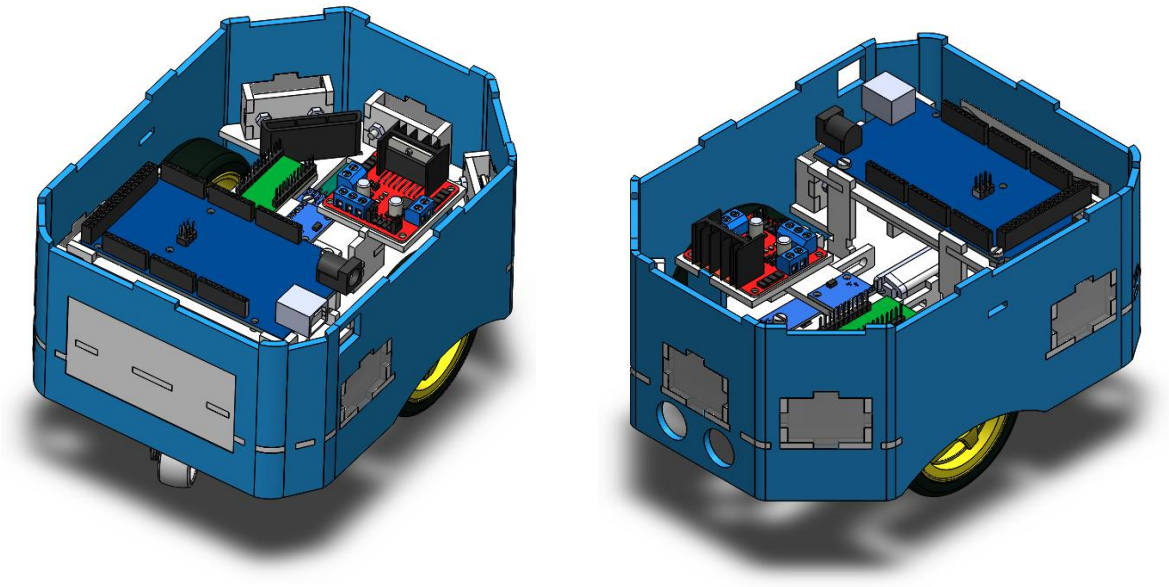


Figura 116. En blau els tancaments laterals per protegir la circuiteria interior.

Tal com es mostra a la Figura 117 i 118 el sensor s'ultrasons s'ha col·locat a la part frontal del Carpher. Els dos forats s'han fet de forma que el sensor quedes totalment encaixat evitant així la necessitat de utilitzar caragols.

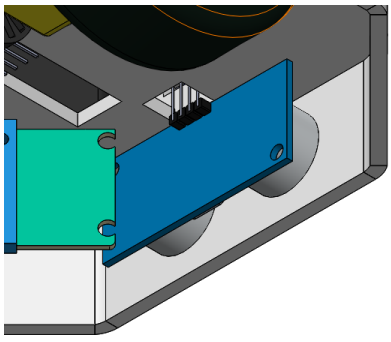


Figura 118. Vista desde la part inferior de l'encaixament del sensor d'ultrasons.

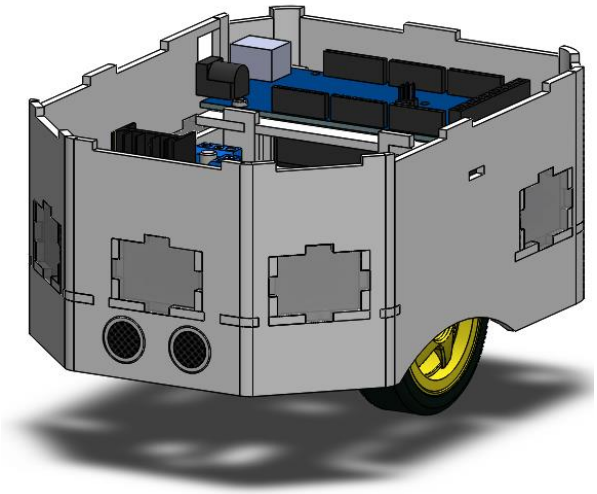


Figura 118. Vista frontal del robot on ñes pot veure el sensor d'ultrasons encaixat.

La següent part a dissenyar ha estat una petita base per als LEDs (Figura 119), se n'han utilitzat un total de set, els cinc primers corresponents a cada un dels sensors i amb la funció d'indicar els estats d'aquest, per tant s'han col·locat just a sobre. Els dos restants s'han col·locat a la part posterior i serveixen per indicar la frenada i complementar els altres cinc. Els LEDs s'han col·locat a 45º respecte a la base, ja que d'aquesta manera la llum arriba d'igual forma a la part superior i a la lateral. Aquests s'han enganxat a una peça petita que es recolza entre la paret i la base. La part superior i el lateral que dóna a l'exterior, s'han retallat sobre metacrilat blanc opac (Figura 120).

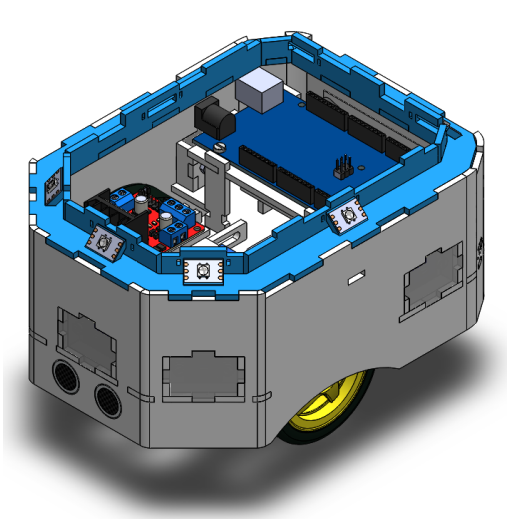


Figura 120. En blau és mostrà la plataforma de suport dels leds.

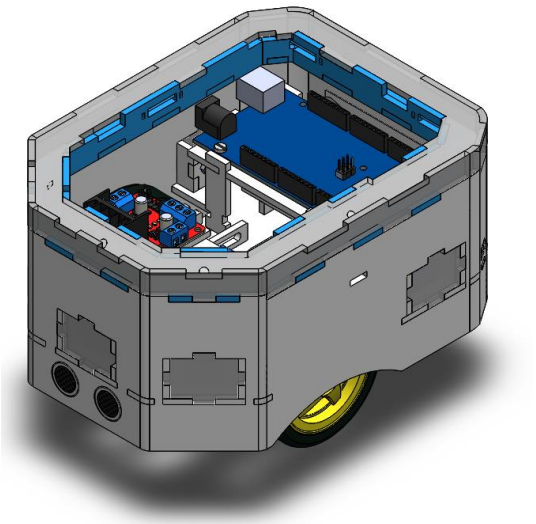


Figura 119. Sobre de la plataforma blava és pot veure el tencament que protegira els leds.

Per tancar el robot s'han col·locat dues tapes diferents, la primera fixa i situada a la part frontal (Figura 121). En aquesta s'incorpora un tancament per un detector làser i un forat per poder passar els pins del sensor de temperatura i humitat DHT22. La segona tapa, col·locada a la part posterior, té un petit forat per poder aixecar-la amb el dit, el que permet accedir a la placa d'Arduino de forma ràpida i senzilla (Figura 122).

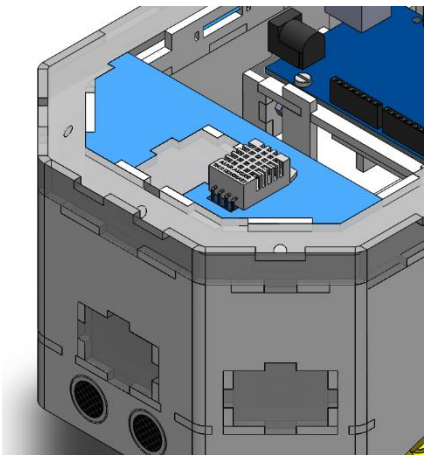


Figura 122. En blau es mostra la tapa petita on se situa el sensor DHT22 i el tancament per sensor làser superior.

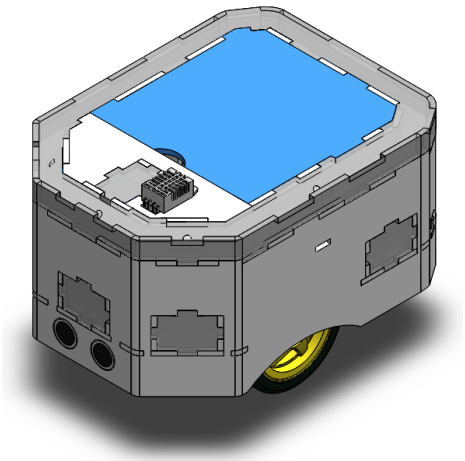


Figura 121. En blau és mostra la tapa superior gran.

2.4.1.2 Disseny 2D utilitzant DraftSight

SolidWorks proporciona una eina per exportar els sòlids en DXF o DWG, formats amb els quals treballa DraftSight. En el menú desplegable es pot escollir la cara del sòlid què es vol exportar i finalment, abans de guardar, és mostra una vista preliminar del resultat. A la Figura 123 es pot veure una captura de pantalla en exportar la base del robot en DXF/DWG.

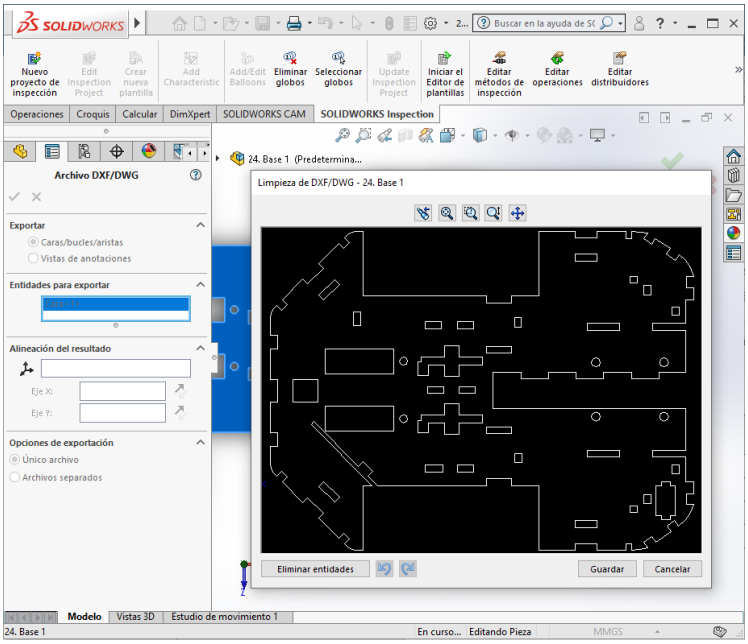


Figura 123. Captura pantalla SolidWorks en exportar un sòlid.

Un cop s'han exportat les peces del robot què s'han de tallar, s'obren en el DraftSight i mitjançant l'opció copiar i enganxar, es col·loquen en una mateixa fulla que és una plantilla dissenyada expressament per la talladora làser que s'ha utilitzat. Abans d'enviar les peces a imprimir, se n'han de modificar algunes i col·locar-les de forma eficient dins d'un requadre que simula les mides del metacrilat on es tallarà.

Per tal que les parets laterals del robot siguin una sola peça uniforme, s'han d'ajuntar mitjançant un patró que permet corbar-les. Tal com es veu en les Figures 124 i 125, a internet hi ha infinitats de patrons i dissenys que permeten curvatures de moltes formes i flexibilitats diferents.

Figura 124. Braçalets dissenyats en fusta corbada i tallada amb làser per l'empresa CounterglowDesigns.

Figura 125. Caixa de fusta amb curvatura tallada en làser per guardar anells i dissenyada per l'empresa KWS.

A la Figura 126 és mostra el patró que s'ha aplicat per a la construcció del xassís de Carpher, aquest consta de columnes de línies intercalades entre si de forma equivalent que aporten una gran flexibilitat. Només és vàlid per angles menors de 90º, com és en el cas d'aquest projecte.

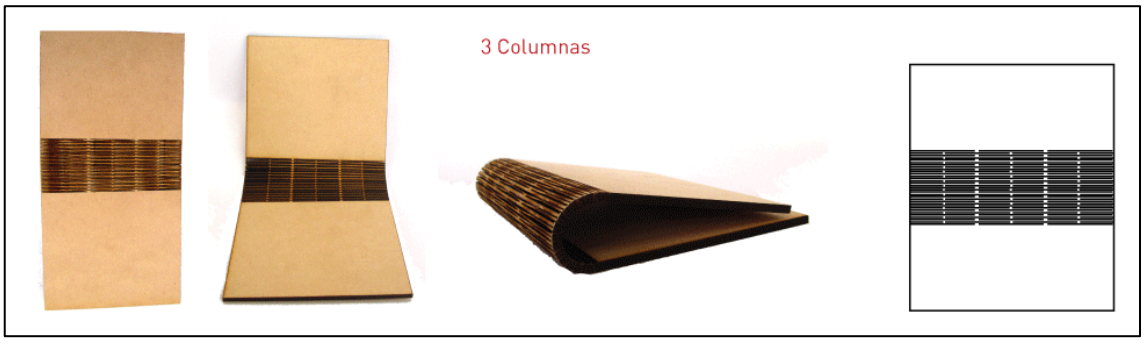


Figura 126. Patró de curvatura lineal per angles menors de 90º.

A la Figura 127 es pot veure un full d'Excel creat per calcular les mides ideals del patró. Les ranures, de llargada, no han de ser mai inferiors a 0,5 mm ni superiors a 1 mm. Per altra banda, les columnes han de ser superiors a 1mm tot i que l'ideal és 1 mm. El nombre de ranures dependrà de l'arc exterior de la curvatura, i se li sumarà 2mm per costat per suavitzar la implementació del patró. Respecte a l'alçada, les ranures s'intercalen entre elles i l'espai entre ranures ha de ser d'1/5 part de la llargada d'aquesta. El nombre de ranures ideal és de dues i mitja, el mig és definit per la necessitat d'intercalar-les entre elles.

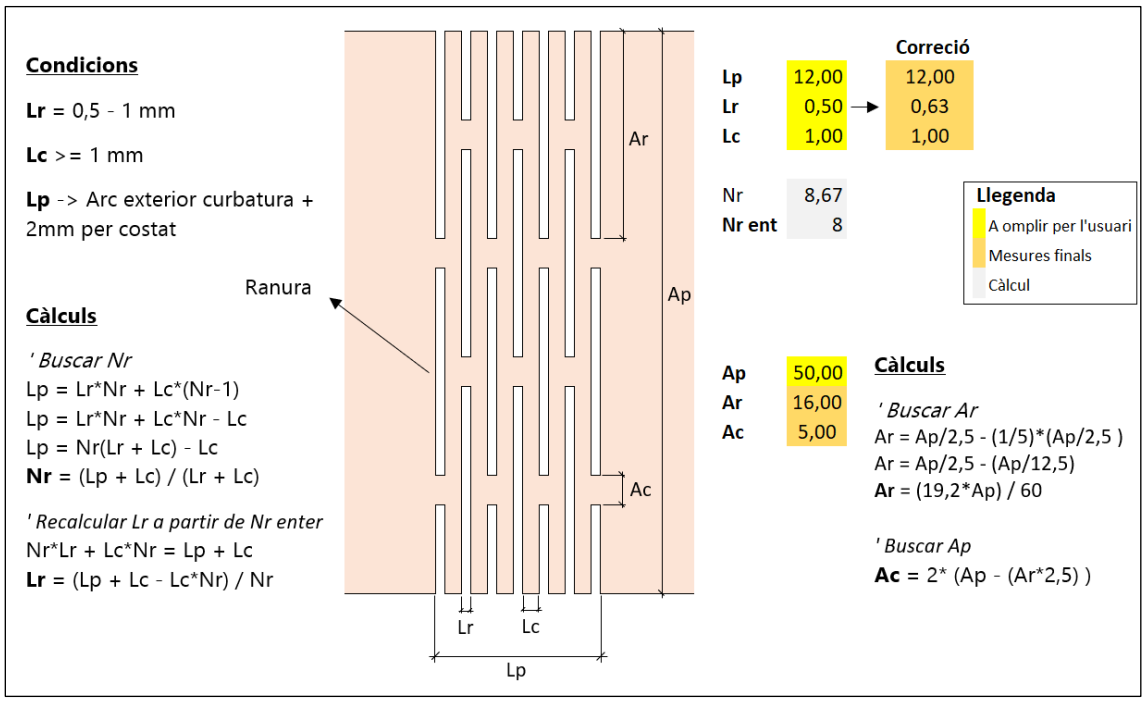


Figura 127. Captura de pantalla del full d'Excel creat per calcular les mides del patró.

Un cop aplicades les curvatures a les parets es passa a col·locar totes les peces dins del requadre que simula la placa de metacrilat. Mitjançant capes, s'indica a la talladora làser l'ordre en què ha de tallar les línies, que ja vénen creades en la plantilla. Si s'hagués de tallar una peça quadrada, on a dins hi hagués una rodona per més tard poder passar un clau, s'ha de tenir en compte que primer s'ha de tallar la rodona i després el quadrat, ja que si primer és talles el quadrat, i aquest caigués d'un costat deixant-lo en diagonal respecte la placa de tall, llavors la rodona quedaria deformada.

2.5 Construcció del robot

Per a la construcció de Carpher s'ha decidit tallar les peces sobre fusta. Aquestes serviran per validar el disseny i corregir les possibles errades. La fusta utilitzada es pot veure a la Figura 129, i és contraplacada, el què significa que està feta amb capes fines de fusta d'un mm cada una i les quals es col·loquen unes sobre les altres en direcció de la veta alternada, es pot observar a la Figura 128. El gruix és de 3 mm (igual que el metacrilat mencionat anteriorment).

Figura 129. Tauler de contraplacat comercialitzat per AKI i usat en el projecte.

Figura 129. Contraplacat de calabó on és poden apreciar les diferents fulles de fusta.

La talladora amb la qual s'ha treballat en aquest projecte és l'Epilog Zing 24 (Figura 130) amb una potència de 40 W. L'àrea de gravat és de 610x305 mm i pot tallar materials de fins a 197 mm de gruix.

Figura 130. Talladora làser Epilog Zing 24.

Per enviar les peces a la talladora làser és necessari imprimir-les primer en format pdf. Un cop exportat, s'obre l'arxiu i s'envia al programa de verificació. En aquest es comprova que totes les línies són de tall, no de gravat, i què el gruix és de 0,05 mm. Un cop verificat, ja es pot enviar l'arxiu a la talladora. A la Figura 131 es pot veure la màquina treballant.

Figura 131. Talladora làser treballant en les peces del projecte.

Un cop es va imprimir la primera remesa de peces es van detectar diversos problemes. El primer relacionat amb la flexibilitat de la peça lateral del xassís, ja que en doblegar-la els laterals tendien a trencar-se amb facilitat. A les Figures 132 i 133 es poden veure algunes de les peces trancades. Per solucionar el problema es van redissenyar els patrons de curvatura i es van reimprimir les peces que presentaven el defecte.

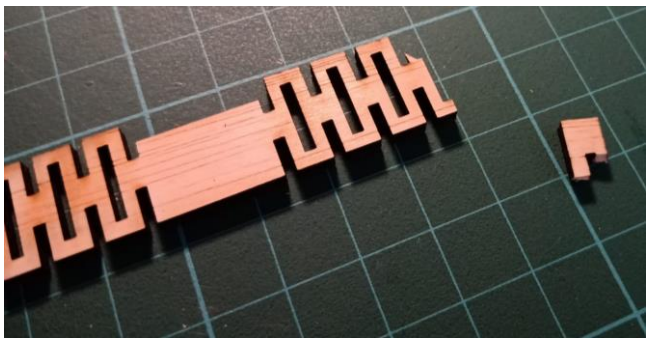


Figura 133. Peca corba trencada per culpa d'un mal disseny del patró de cobertura.

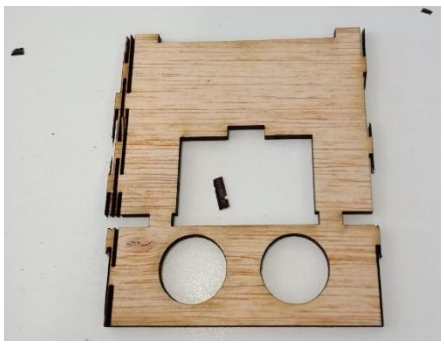


Figura 133. Peca corba trencada per culpa d'un mal disseny del patró de cobertura.

Un cop es va muntar un dels motors a la base i s'hi va col·locar la roda, es va veure que en tancar el lateral, aquesta peça xocava amb la roda i per tant no encaixava. Un cop detectada l'errada, es va mesurar l'espai que faltaria (3 mm) i en el nou disseny es va

augmentar la base amb els mm necessaris per cada una de les bandes. També es van haver de modificar els laterals perquè tanquessin bé. Un cop redissenyat, es van imprimir les noves peces.

En instal·lar i provar el sistema *push-to-open*, es va observar que actuava amb molta força, això provocava que per obrir la plataforma s'hagués de pressionar molt i a la llarga podria provocar danys a les peces. Per aquest motiu es va decidir eliminar-lo i en el nou disseny, per tal d'obrir la plataforma, s'ha d'empentar per sota. A la Figura 134 es pot observar el procediment.

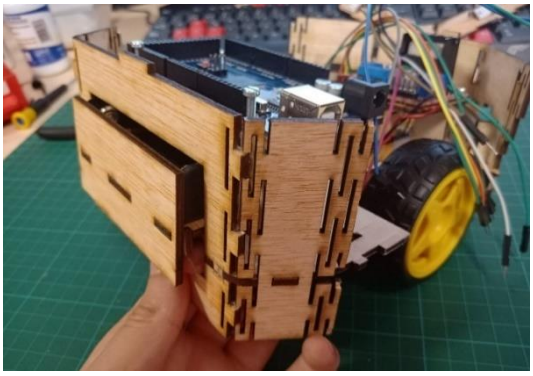


Figura 134. Demostració de com obrir la plataforma de la bateria.

A la Figura 135 es poden veure totes les peces tallades un cop aplicats els canvis en el disseny.

En aquesta segona impressió els laterals es corbaven correctament, però en tots sobrava entre 1 i 2 cm a causa d'un error en el càlcul de l'arc exterior. Això ha provocat que les peces no encaixessin amb la facilitat que estava prevista.

Tot i el problema anterior s'ha decidit tirar endavant amb la construcció del robot, ja que la fase de disseny s'ha allargat més del previst robant temps a la part de programació. També s'ha de tenir en compte que aquest problema és estètic i per tant no efecte al funcionalment del robot. Així doncs, Carpher ha estat construït amb peces de fusta i si, finalment es disposa de temps, es redissenyarien les petites parts defectuoses i es tallarien totes les peces directament en metacrilat.

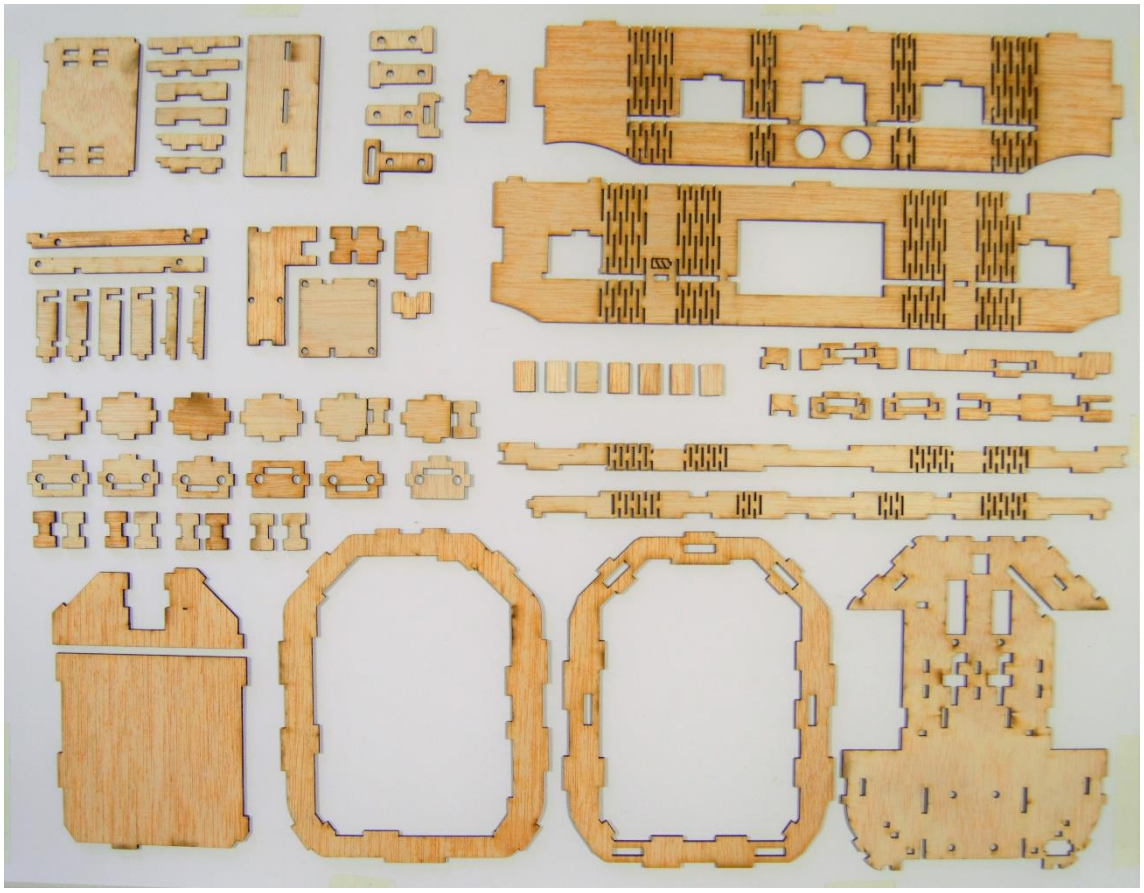


Figura 135. Conjunt de totes les peces necessàries per construir el robot.

Per enganxar les peces s'ha utilitzat cola blanca, i en les zones més fràgils adhesiu d'enduriment ràpid, més coneguts pel nom de Superglue™. A les Figures 136 i 137 es pot veure el moment del procés d'enganxat que consisteix a aplicar força per tal d'assegurar que les peces queden ben fixades.

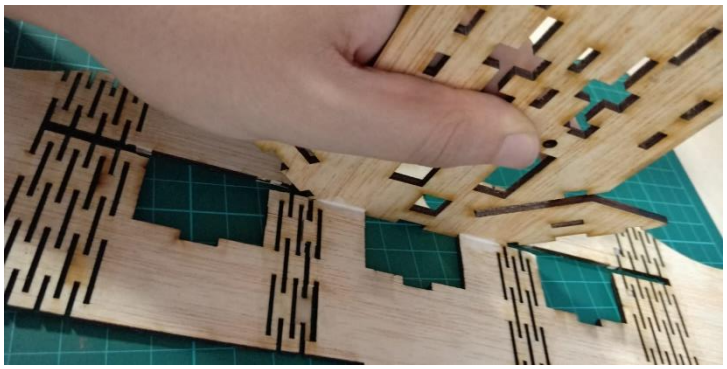


Figura 137. Imatge feta durant l'encolament de la base i la part frontal.

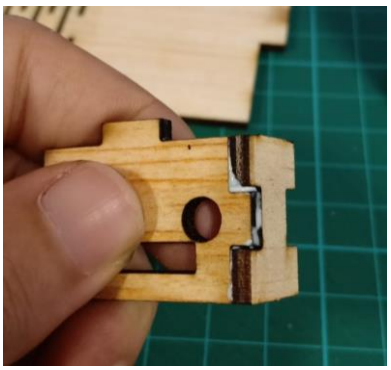


Figura 137. Unió d'un tancament per sensor làser.

Per solucionar el defecte de les peces laterals, s'ha tallat el cm que sobrava per corba i s'ha llimat la zona amb paper de vidre. Finalment, s'han enganxat les peces amb l'adhesiu d'enduriment ràpid. A la imatge 201 es pot veure el resultat.

Figura 138. Imatge feta durant l'enganxament dels laterals.

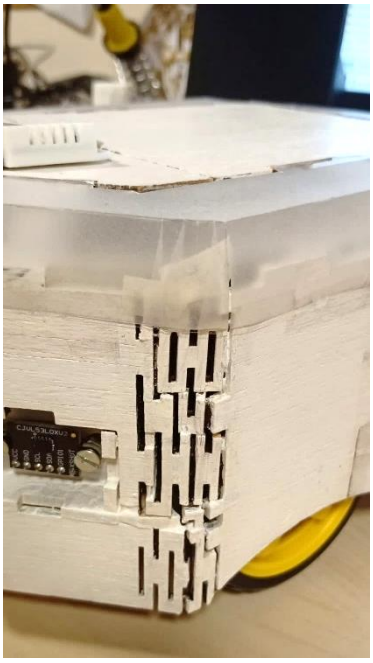


Figura 139. Resultat final d'enganxar dos laterals.

A mesura que s'encaixaven les peces, s'han anat col·locant els components i pintant les peces de fusta ja fixades de color blanc. Les Figures 140 i 141 són imatges fetes durant la construcció del Carpher.

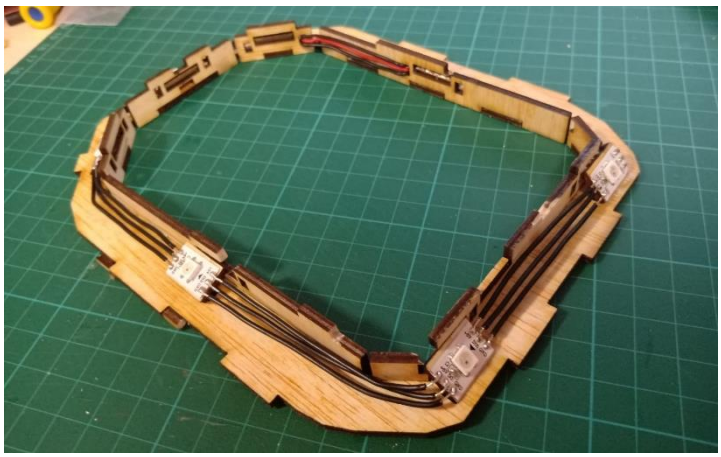


Figura 140. Plataforma de suport dels LEDs un cop finalitzada.

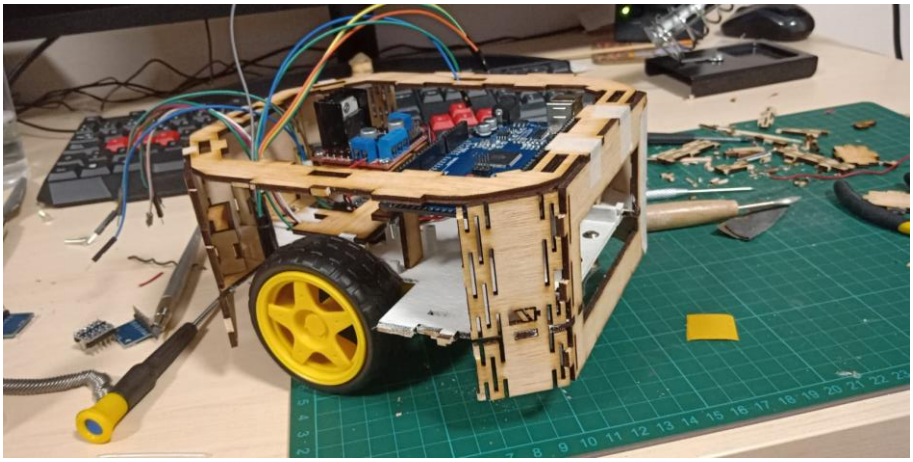


Figura 141. Imatge feta durant la connexió dels encoders.

Al moment de fer les connexions s'han adaptat els cables dupont perquè tinguessin una llargada adequada. Si es deixessin tots llargs provocaria que l'interior del robot fos un caos i, en cas d'haver de modificar algun component, seria molt difícil accedir-hi i diferenciar els cables.

A continuació és mostra el procés que s'ha seguit per adaptar els cables del controlador de motors. En aquest cas s'han agafat un conjunt de sis cables dupont (Figura 143) i se'ls hi han tallat els extrems. Després se'ls ha soldat connectors femelles (Figura 142).

Figura 143. Cables Dupont mascle -mascle de 20cm. Comercialitzats per l'empresa Naylamp Mechatronics

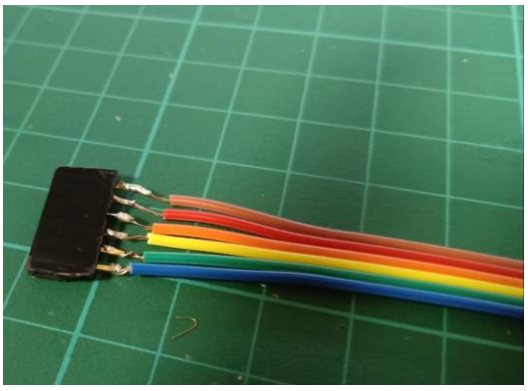


Figura 142. Soldadura de varis cables amb pins femella.

Per protegir i aïllar el punt de soldat s'ha utilitzat un tub termoretràctil (Figura 145, 144 i 146). Aquest s'ha tallat a mida i s'hi han introduït els cables a l'interior de forma que el tub quedes sobre l'àrea a protegir. Per acabar s'ha utilitzat una font de calor, en aquest cas un secador de cabell, per a què el plàstic és comprimeixi i quedi adherit.

Figura 146. Tubs termoretràctils comercialitzats per LIBERRWAY.

Figura 146. Abans i després d'aplicar calor sobre el tub termoretràctil.

Figura 146. Cable modificat a mida per la connexió del mòdul controlador de motors.

Per tal de connectar respectivament i en una sola línia tots els pins SDA, els SCL, l'alimentació i el terra dels sensors làser, s'ha utilitzat una placa de prototipat per soldar. Aquesta s'ha tallat a mida i s'hi han connectat els cables de cada sensor. A la Figura 147 es pot veure el resultat final.

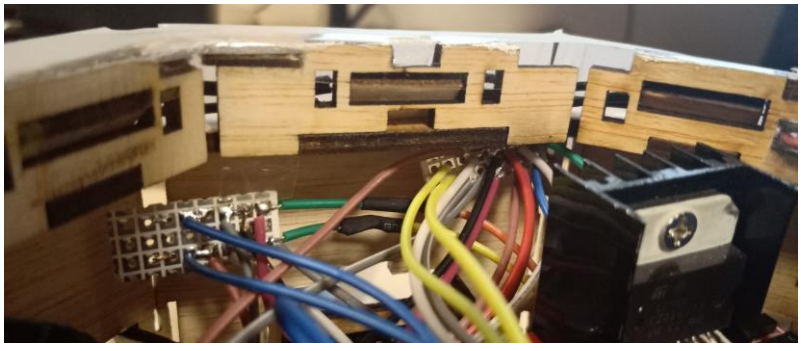


Figura 147. Resultat de soldar en una sola línia l' Vcc i GND dels sensors làser.

Pel circuit addicional del Wixel, el que reinicia l'Arduino, també s'ha utilitzat una petita part de placa de prototipat per aconseguir que ocupes el menor espai i a la vegada el muntatge fos més robust. A la Figura 148 es pot veure el resultat un cop soldat i a la Figura 149 un cop connectat en el robot.

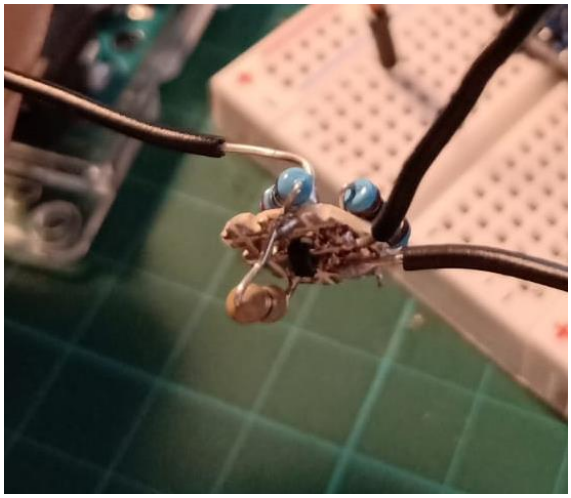


Figura 148. Circuit resultant de soldar es components per reiniciar la placa.

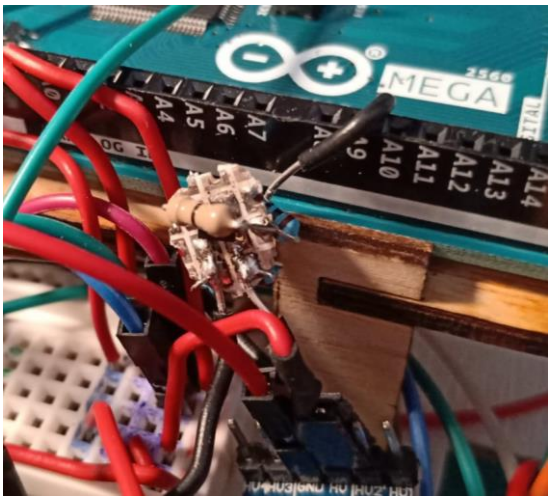


Figura 149. Circuit implementada en el robot.

A les peces que guarden els LEDs, que haurien de ser originalment de metacrilat opac, s'ha utilitzat paper vegetal. A les Figures 150, 151, 152 i 153 es poden veure més imatge de la construcció del robot.

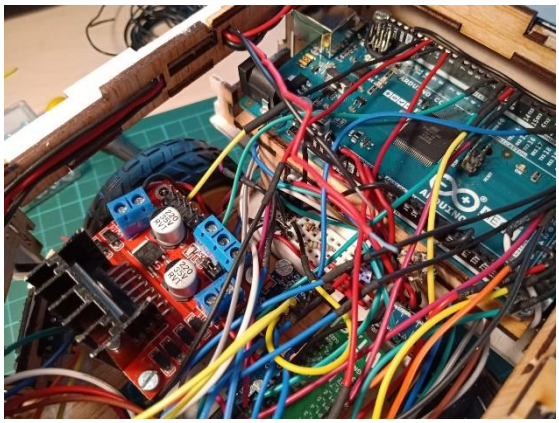


Figura 150. Interior del robot.

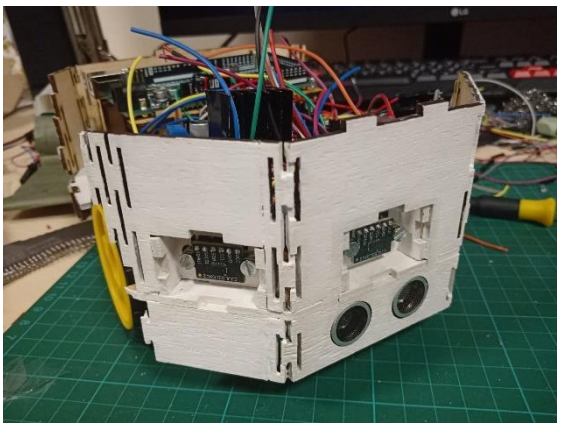


Figura 151. Part frontal druant la connexió dels sensors làser.

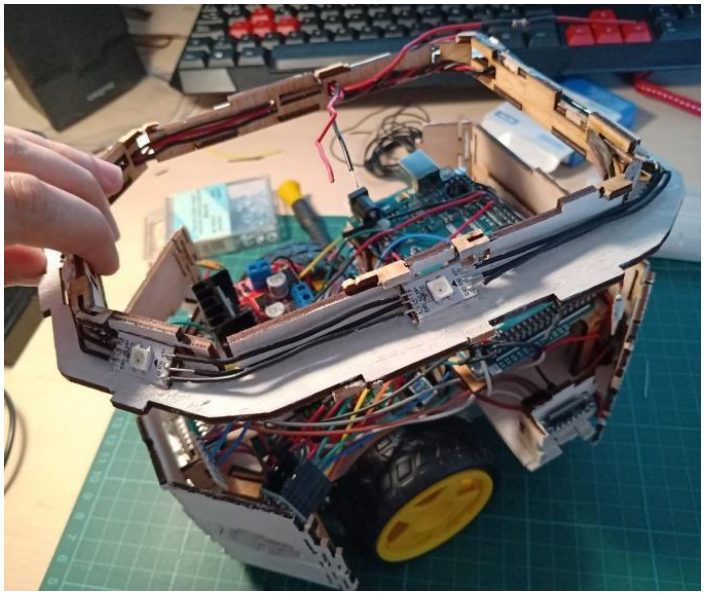


Figura 152. Posicionament de l'estructura dels LEDs.

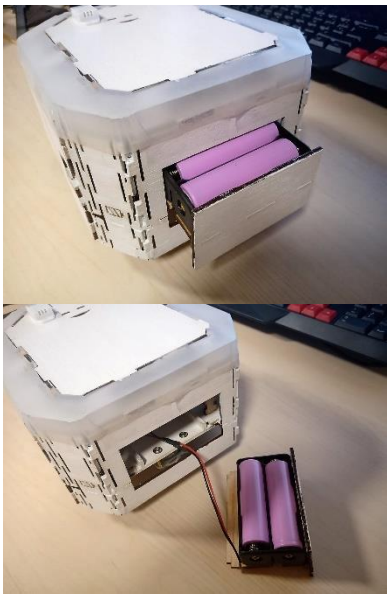


Figura 153. Plataforma bateria.

El robot finalitzat es pot veure a les figures 140, 141, 142 i 143.

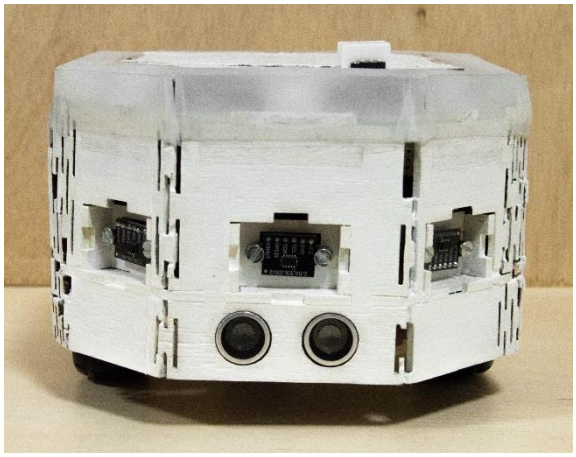


Figura 154. Vista frontal del robot.

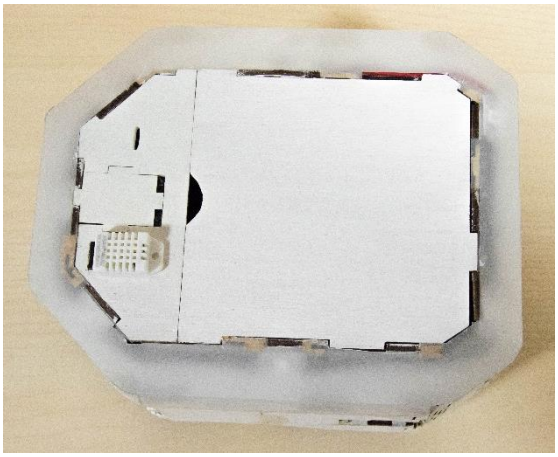


Figura 155. Vista superior del robot.

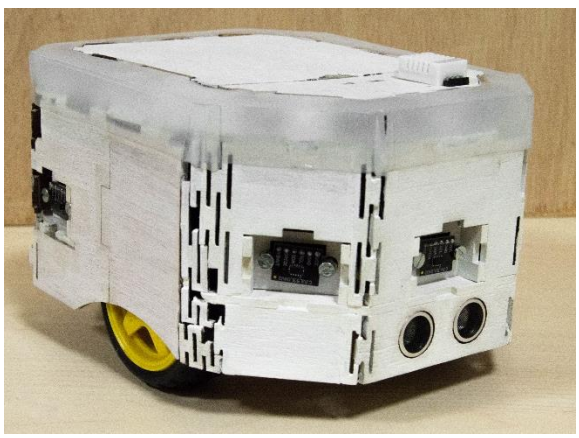


Figura 157. Vista lateral del robot.

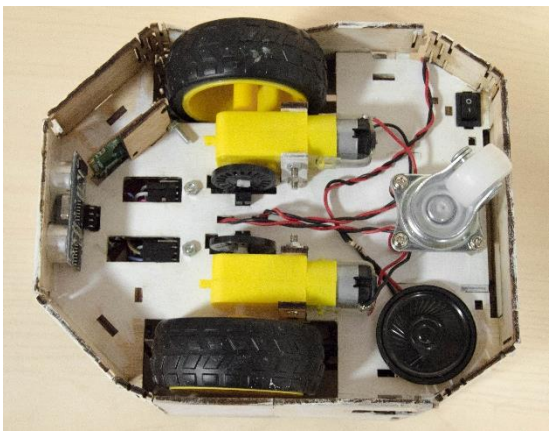


Figura 156. Vista inferior del robot.

2.6 Cost de fabricació

Els components i materials necessaris per a la creació de Carpher s'han comprat tant en botigues físiques com en línia. A l'hora de realitzar el pressupost del hardware es diferencien tres grups:

- **Elements mecànics**. S'inclouen les rodes, el cost de la plataforma i els diferents materials que s'han utilitzat per muntar-la.
- **Elements elèctrics**. Es comptabilitza el preu de les piles i tot el material relacionat amb aquestes, l'interruptor, els cables, plaques de prototipat, etc.
- **Elements electrònics**. Inclou tots els mòduls que s'han utilitzat en el projecte així com les resistències i condensadors entre d'altres.

A la Taula 9 és mostra el pressupost total derivat d'aquests tres grups. En l'Annex I és mostra el pressupost detallat.

Hardware	
Concepte	Import
Elements mecànics	49,55 €
Elements elèctrics	79,11 €
Elements electrònics	153,48 €
Subtotal	282,15 €

Taula 9. Resum del cost del Hardware del projecte.

El software utilitzat és tot gratuït sempre que es compleixin certes condicions, a la Taula 10 és mostra el resum d'aquest i les circumstàncies necessàries per poder fer-ne un ús amb cost zero.

SOFTWARE		
Producte	Observacions	Import
IDE Arduino	Software lliure	0,00 €
UE4 Editor	Gratuït fins a superar 3000 € de benefici amb el producte creat	0,00 €
SolidWorks 2018	Gratuït amb llicència d'estudiant.	0,00 €
DraftSight	Gratuït versions anteriors a la del 2019	0,00 €
	TOTAL	**0,00 €**

Taula 10. Resum del cost del software del prjecte.

Sense contar el cost de mà d'obra el pressupost del projecte suma un total de **DOS-CENTS VUITANTA-DOS AMB QUINZE CÈNTIMS D'EURO.**

CAPÍTOL 3: PROGRAMACIÓ DEL ROBOT

En aquest capítol s'explicà el funcionament bàsic del robot diferenciant els dos modes de control. El primer és el control manual que l'usuari podrà fer sobre el robot mitjançant el comandament tipus PS2. El segon serà el control autònom, és a dir, que serà el mateix robot el què prendrà les decisions de per on moure's. També s'explicarà el protocol utilitzat per la comunicació en sèrie entre l'emissor, el robot, i el receptor, l'ordinador. Per últim s'explicarà la comunicació, mitjançant els LEDs i l'altaveu, del robot cap a l'usuari.

Quan s'engega el mòbil aquest inicialitza tots els mòduls i llibreries del programa. Un cop ha finalitzat, ho indica a l'usuari a través dels LEDs, aquests canvien entre diferents tons de blaus a la vegada què es reprodueix una curta melodia per l'altaveu. Després es posa en espera, fins que rep una resposta a través del comandament tipus PS2. Aquesta resposta determina el funcionament del robot, si es pressiona L2 es passa a control manual, és a dir, que l'usuari pot moure el robot a través del comandament. D'altra banda, si es pressiona R2 es passa a control autònom, en el qual el robot cartografia l'espai sense l'ajuda humana. En aquest apartat s'entra en detall en el control manual deixant pel següent apartat el control autònom.

3.1 Control manual del robot

Un cop activat el control manual, el funcionament de Carpher es fa mitjançant els dos *joysticks* del comandament tipus PS2. El dret serveix per indicar la direcció i l'esquerra la velocitat i el sentit. Abans de continuar és important explicar el funcionament d'aquestes palanques analògiques.

Un *joysticks* està format per dos potenciòmetres a 90° que transformen el moviment en x i en y de les palanques en dos senyals elèctriques proporcionals a la seva posició. A la Figura 158 es pot veure la seva estructura interna. A més, també incorpora un botó que s'activa en pressionar una de les palanques.

Figura 158. Estructura interna d'un joysticks.

En ser dispositius analògics proporcionen mesures entre 0 i 255, per tant sabem què, quan el *joystick* es troba en repòs, els valors que marcarà tant de x com de y seran 127.

El control de la velocitat i el sentit és senzill, ja que només s'utilitza la palanca y del *joystick* esquerra. A la Figura 159 es pot veure la relació què hi ha entre el valor de la palanca i la velocitat que se li aplica al robot. Quan la palanca està en repòs, valor de 127, la velocitat és 0, mentre que si es tira la palanca endavant, el valor incrementa fins a 255, i proporcionalment també ho fa la velocitat, llavors el robot va endavant. Per contra, si la palanca es tira cap a sota, el valor es redueix fins a 0, però la velocitat s'incrementa de forma inversament proporcional, ja que el robot canvia el seu sentit cap endarrer.

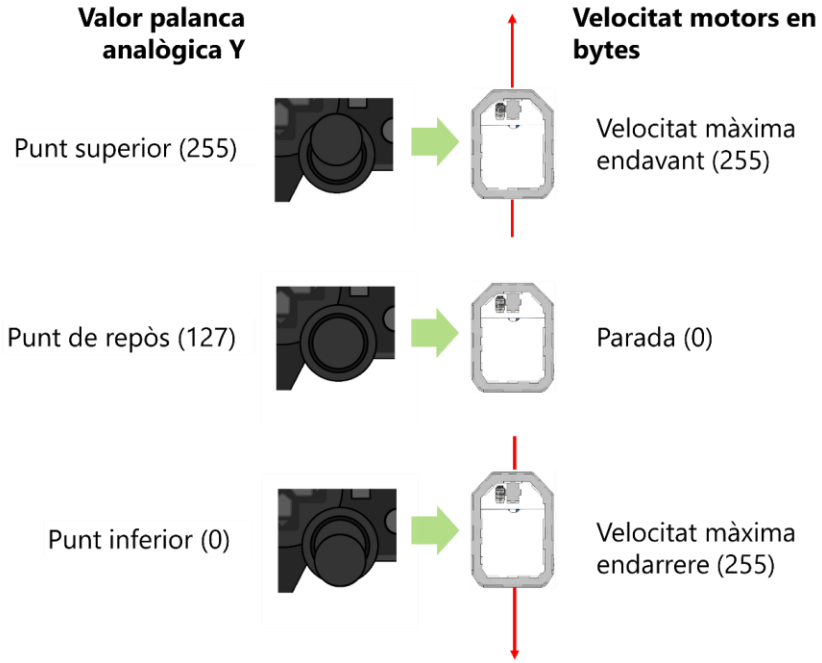


Figura 159. Esquema de la relació entre el valor del joystick i la velocitat i sentit que s'aplicarà als motors.

S'ha de tenir en compte que els motors funcionen a partir d'uns bytes determinats, per sota d'aquests la bateria de sistema no aporta suficient potència, tant pot ser que cap motor funcioni com que només ho faci un. Per trobar aquest punt s'ha posat els dos motors en marxa i s'ha anat augmentant la velocitat byte per byte fins que ambdós funcionessin correctament, el valor mínim resultant ha estat 130. També cal tenir en compte què el joystick pot quedar mal posicionat en deixar-lo en repòs, per això s'han

deixat cinc valors per sobre i cinc per sota on, igual que el 127, indiquen una aturada del robot. A la Figura 160 es pot veure la relació que s'aplica entre els valors de la palanca analògica Y i el valor que és dona de velocitat als motors per valors de la palanca entre 117 i 138.

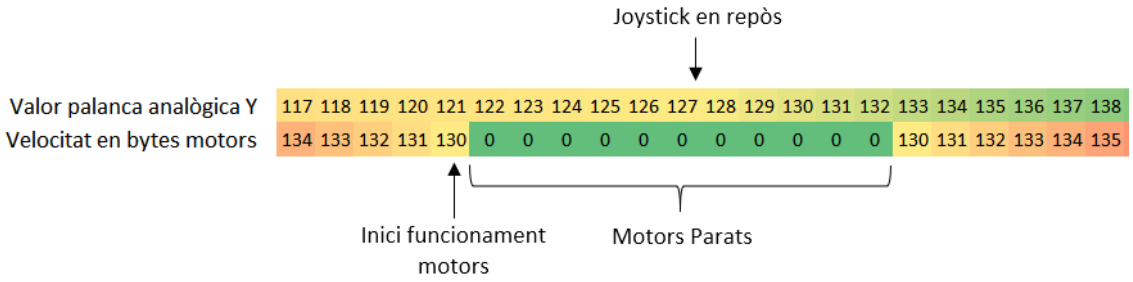


Figura 160. Rang de seguretat aplicat a la relació entre el valor de la palanca analògica Y d'un joysticks i la velocitat en bytes aplicada als motors.

Com s'ha dit anteriorment, el *joystick* dret servirà per a la direcció que se li vol donar al mòbil. A la Figura 161 es veu el resum del funcionament tenint en compte que el *joystick* esquerra està en la posició màxima (255), és a dir velocitat màxima i sentit endavant.

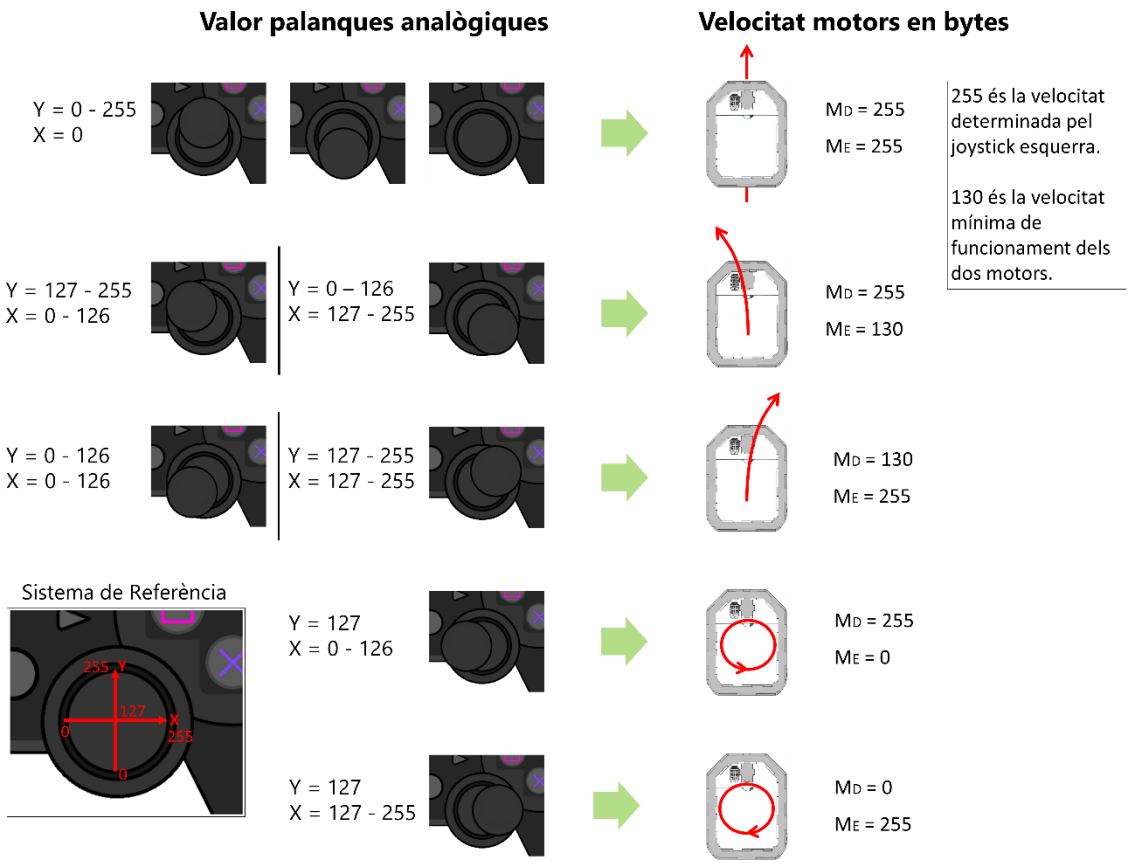


Figura 161. Relació entre les posicions del joystick dret i la direcció que se li donarà al robot.

PID

Els motors utilitzats en el projecte no són de molt bona qualitat i això porta com a conseqüència errors en el moviment del robot. Quan es configuren els dos motors a la mateixa velocitat per tal que vagin recte aquests no ho fan, ja que el dret té tendència a funcionar amb més força el que acaba causant que el robot vagi cap a l'esquerra.

Per arreglar el problema s'implementarà un controlador PID, aquest utilitzarà com a consigna els polsos del motor dret i calcularà l'error comparat amb els polsos del motor esquerre. El resultat se sumarà a la velocitat que se li dóna al motor esquerre per tal d'igualar el nombre de polsos dels dos actuadors i d'aquesta forma aconseguir que el robot es mogui recte.

Un controlador PID és un algoritme utilitzat pel control de sistemes realimentats. És relativament senzill d'implementar i és capaç de proporcionar un bon comportament en una gran varietat de situacions sense la necessitat de conèixer en detalla la planta que es vol controlar. En el diagrama de blocs de la Figura 162 es pot veure el sistema què s'implementarà.

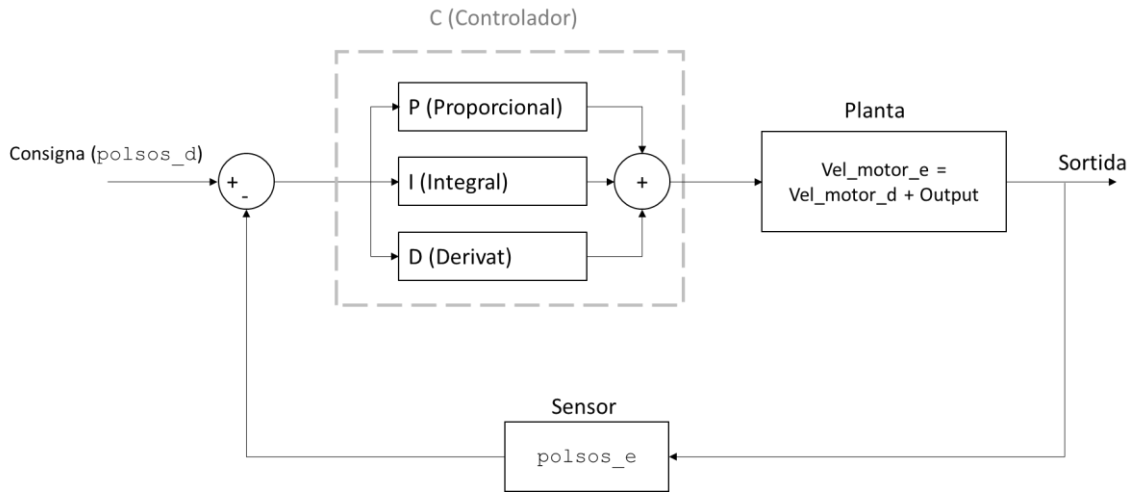


Figura 162. Diagrama de blocs del sistema de control de la velocitat dels motors.

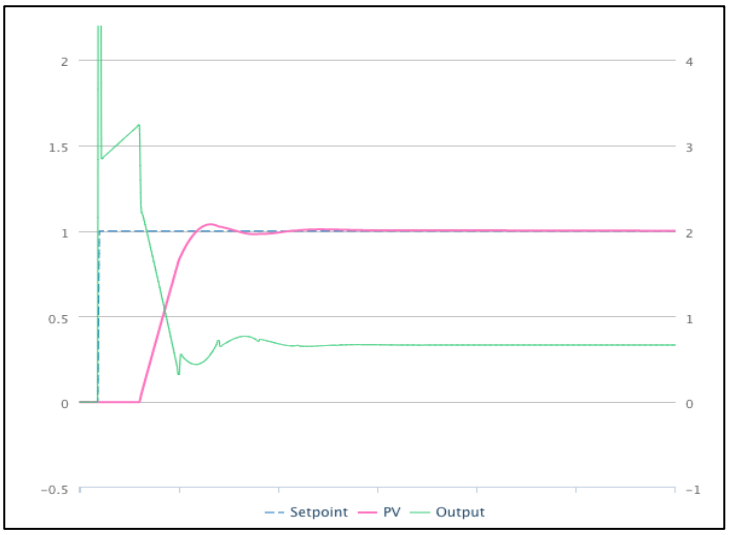
L'algoritme PID està format per la suma de tres components: proporcional, integral i derivat. Matemàticament s'expressa de la següent forma:

$$Output(t) = K_p \cdot e(t) + K_i \int_0^t e(t) \cdot dt + K_d \cdot \frac{de}{dt}$$

Cada component PID és independent dels altres i l'error que calculen és el què per ells s'hauria d'implementar per tal d'aconseguir una millor resposta del sistema. Els tres components se sumen per tal de donar la sortida del controlador. Cada un compleix amb una certa funció la qual s'explicarà a continuació:

- **Proporcional,** s'associa al present, és la diferència entre la consigna i valor real.
- **Integral,** reacciona a la memòria del passat el què permet al controlador eliminar per complet l'error a llarg termini. De forma senzilla es calcula amb la suma de tots els errors anteriors.
- **Derivat,** respon a la velocitat de canvi de l'error, s'associa al futur. De forma se senzilla, és la diferència entre l'error actual i l'anterior dividit pel temps de mostreig.

Cada planta és diferent i per tant té una reacció diferent davant d'un PID, Els factors Kp, Ki i Kd permeten controlar els tres components per tal d'adaptar l'algoritme al sistema. Una bona calibració de les constants proporcionarà una resposta ràpida i estable mentre que una mala calibració pot portar des de respostes molt lentes a respostes que oscil·lin indefinidament. A la Figura 163 es pot veure un exemple de la resposta ideal del sistema mentre que a la Figura 164 es pot veure, pel mateix controlador, una resposta oscil·lant.

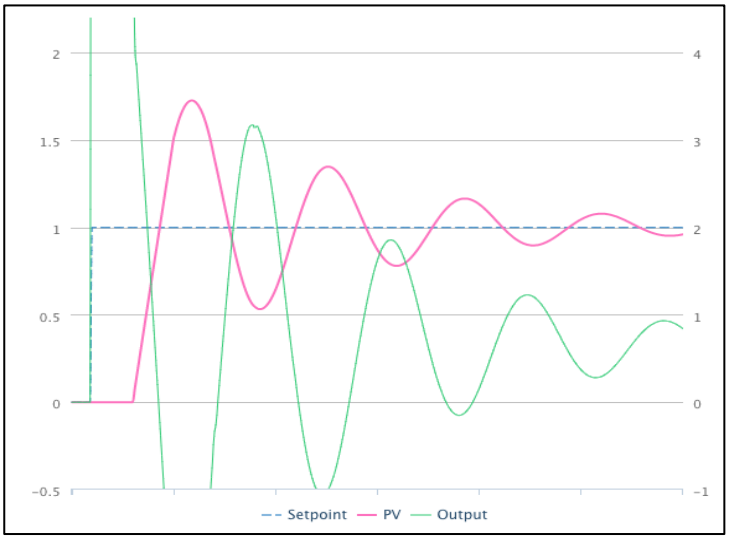


Kp = 2.9

Ki = 0.022

Kd = 20.4

Figura 163. Resposta ideal d'un controlador PID.



Kp = 5.1

Ki = 0.045

Kd = 18.7

Figura 164. Resposta oscil·lant d'un controlador PID.

Programació del PID

El PID que s'implementarà en el robot és molt senzill, per Internet es poden trobar formes més complexes de calcular els diferents components i com a resultat obtenir una resposta molt més precisa, però en el cas d'aquest projecte no es busca aquest grau de precisió.

El càlcul del PID es realitzarà dins d'una funció anomenada `Compute()`, les variables però seran declarades abans del `setup()` i per tant es podran llegir i modificar en qualsevol punt del programa.

El següent codi correspon a la part que aniria dins del `loop()`, en aquesta es defineix el `Setpoint` i l'`Input` i després es crida la funció `Compute()`. Un cop finalitzat el càlcul es defineix la velocitat a la qual aniran els motors, el dret anirà a la qual l'usuari defineixi i l'esquerra serà el valor del dret més l'error del PID. Després es flitaran les velocitats especificades per assegurar que es troben dins del rang, prèviament definit, de funcionament del motor. Per últim s'enviarà la velocitat al L298N per tal que faci el control.

```
//CÀLCUL PID
Setpoint = polsos_d;
Input = polsos_e;
Compute();

//DEFINIR VELOCITAT
sped_d = 175;
sped_e = sped_d + Output;

//FILTRE VELOCITAT MOTORS
if (sped_e > 255)sped_e = 255;
if (sped_d > 255)sped_d = 255;
if (sped_e < 100)sped_e = 100;
if (sped_e < 100)sped_e = 100;

//CONTROL DELS MOTORS
moveForward(pinMotorD, sped_d);
moveForward(pinMotorE, sped_e);
```

Com s'ha dit abans, la funció `Compute()` és l'encarregada de realitzat els càlculs del PID. Primer calcularà el temps que ha passat des de l'últim càlcul, després calcularà les variables d'error i seguidament la funció de sortida del PID. Per últim guardarà el temps actual pel pròxim càlcul.

```
void Compute(){
  // Temps que ha passat des de l'últim càlcul
  unsigned long now = millis();
  double timeChange = (double)(now - lastTime);

  // Càlcul de les variables d'error
  double error = Setpoint - Input;
  errSum += (error * timeChange);
  double dErr = (error - lastErr) / timeChange;

  // Càlcul de la funció de sortida del PID
  Output = Kp * error + Ki * errSum + Kd * dErr;

  // Guardar variables pel pròxim càlcul
  lastErr = error;  lastTime = now;
}
```

Per definir els valors dels factors K s'ha creat la funció `SetTunings(Kp, Ki, Kd)`, aquesta es pot cridar en qualsevol moment del programa.

Diagrama de flux del control manual

Tot l'explicat en aquest apartat s'engloba en una sola funció en el programa d'Arduino.

A la Figura 165 es pot veure un diagrama de flux que la resumeix.

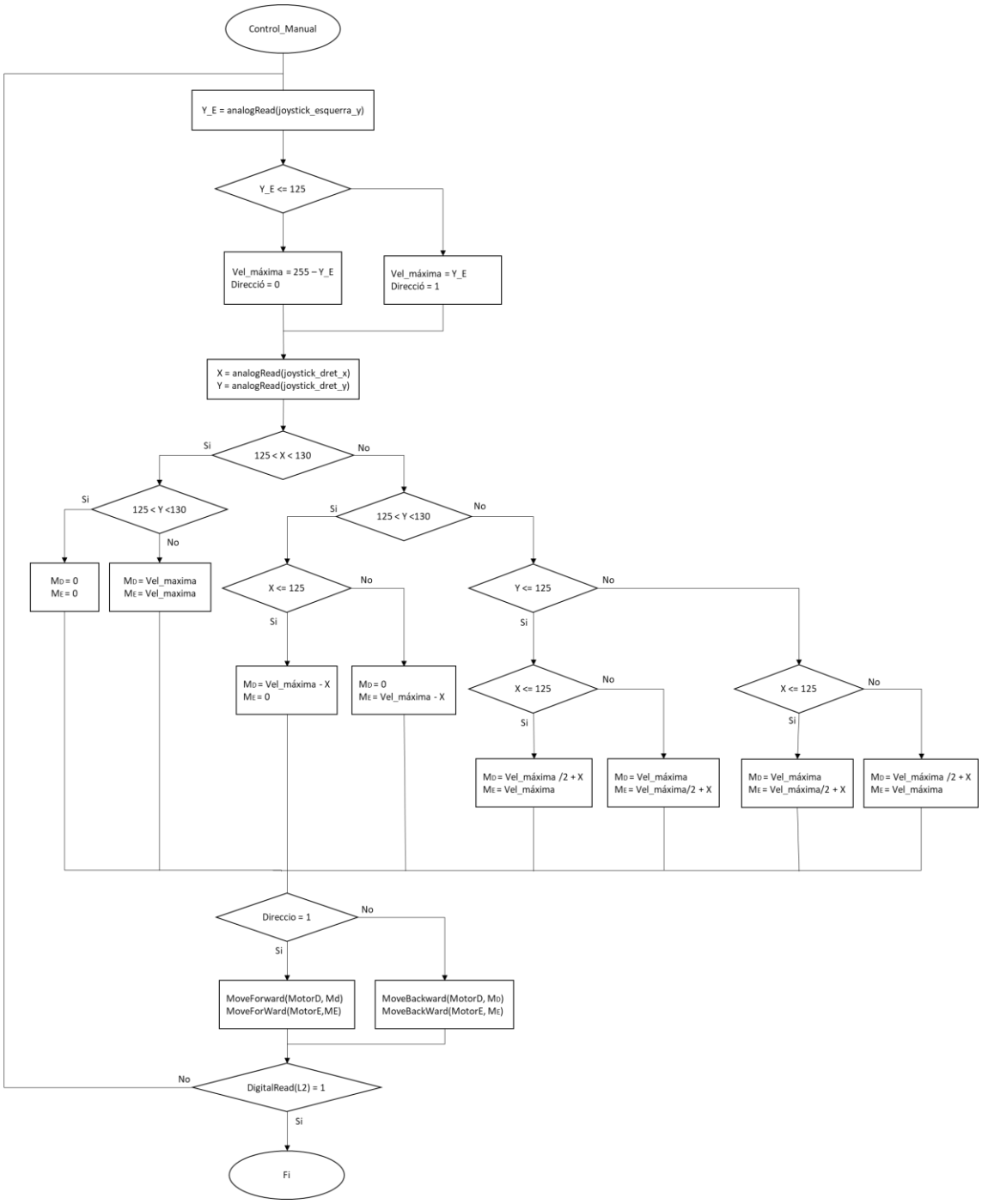


Figura 165. Diagrama de flux del control manual del robot.

3.2 Programa pel control del robot de forma autònoma

En aquest subapartat s'explica com és el funcionament de Carpher per tal que pugui moure's de forma autònoma per un espai i evitant passar diverses vegades per un mateix punt.

El funcionament per tal de seguir les parets es basarà en un PID que mantindrà al robot en paral·lel i a 10 cm de la paret més propera. La consigna seran els 10 cm, i el sensor serà el làser. L'output s'aplicarà a la velocitat dels motors, és a dir que si l'error és de 2 cm, s'aplicarà més força a la roda contrària a la paret per tal d'apropar-se, per altra banda si l'error és de -2 cm s'aplicarà força al motor de la paret per tal que el robot s'allunyi.

Tal com està dissenyat el robot, pel PID, s'han d'utilitzar els sensors frontals que estan a 45 graus, ja que el requisit més important és que estiguin davant de les rodes per tal de poder anticipar-se al moviment. Els sensors laterals s'utilitzaran com a complement, es mirarà que sempre mereixin entre 5 i 20 cm i si la mesura surt d'aquest rang, s'aplicarà una correcció extra a la velocitat dels motors.

A la Figura 166 es pot veure un esquema del Carpher seguint la paret i els sensors que utilitzarà si se segueix una paret que queda a mà esquerra.

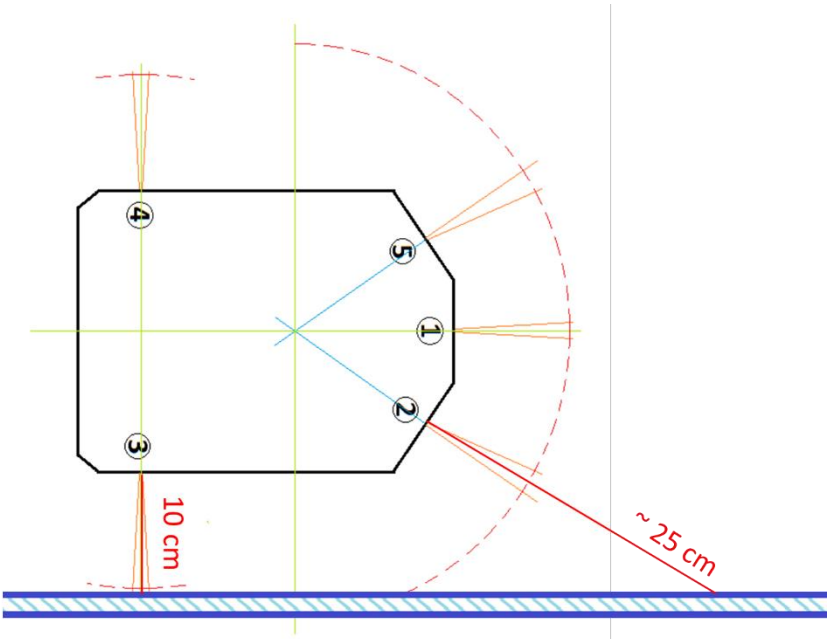


Figura 166. Esquema del robot seguint la paret a 10 cm.

Possibles finals en seguir una paret

Quan el recorregut del Carpher quedi interromput, sigui per una paret o un objecte, aquest ha de ser capaç d'evitar-lo i seguir. Per fer-ho s'han ideat possibles problemes i s'ha buscat com solucionar-los.

El primer cas seria que la paret que s'està seguint gires 90°. Per detectar-ho s'utilitzaria el sensor 1, quan aquest marca menys de 25 cm es farà que el robot giri 90° suaument, de forma que acabes el gir i tingues la nova paret a uns 10 cm aproximadament, en aquest punt es tornaria a posar en marxa el PID. A la Figura 167 es pot veure la situació descrita anteriorment i la solució que hi aplicaria el robot.

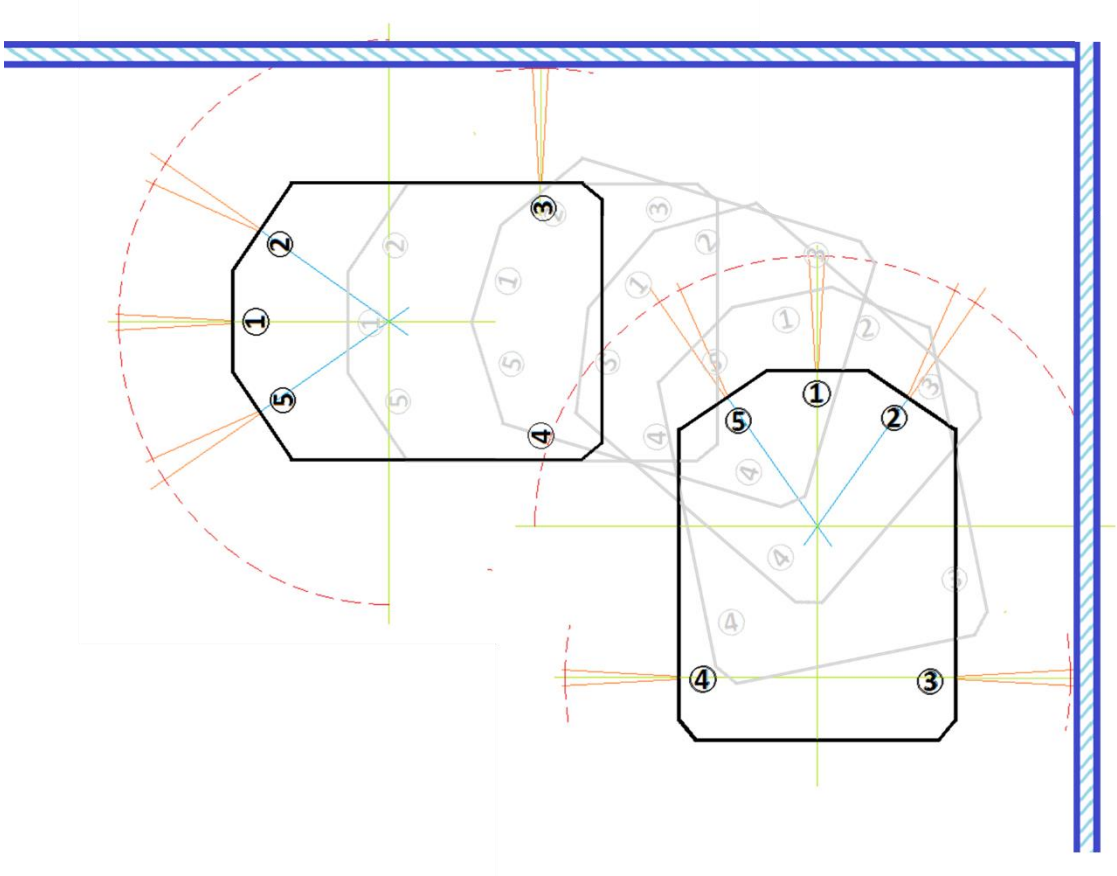


Figura 167. Acció del robot enfront una paret tancada a 90º.

Un altre cas seria que la paret que està seguint s'acabés. Per detectar que s'ha acabat primer el sensor 2 haurà de fer un canvi brusc de la mesura, si això passa s'aturarà immediatament el PID. Després es farà que el robot segueixi recte fins que el sensor 3 deixi de mesurar dins del rang de 5 cm a 15 cm. Un cop en aquest punt es farà un gir sobre el mateix robot de 90° i després es farà anar aquest uns pocs centímetres endavant. L'objectiu ara serà buscar una nova paret per tant, es comprovarà si el sensor 3 mesura alguna cosa o bé està fora de rang. Si és mesura alguna cosa i el sensor 2 mesura una distància menor de 30 cm s'iniciarà el PID per seguir la paret, si la distància és major de 30 cm, es farà que el robot doni una volta sobre si mateix fins a 90º, si abans d'assolir-los el sensor 2 detecta menys de 40 cm s'iniciarà PID. A la Figura 168 és mostra l'exemple quan la paret acabés en un angle superior a 270º. És mostra en diferents colors les diferents accions que realitzaria, aquestes correspondrien cada una a un estat de la màquina d'estats que s'implementaria per cada cas.

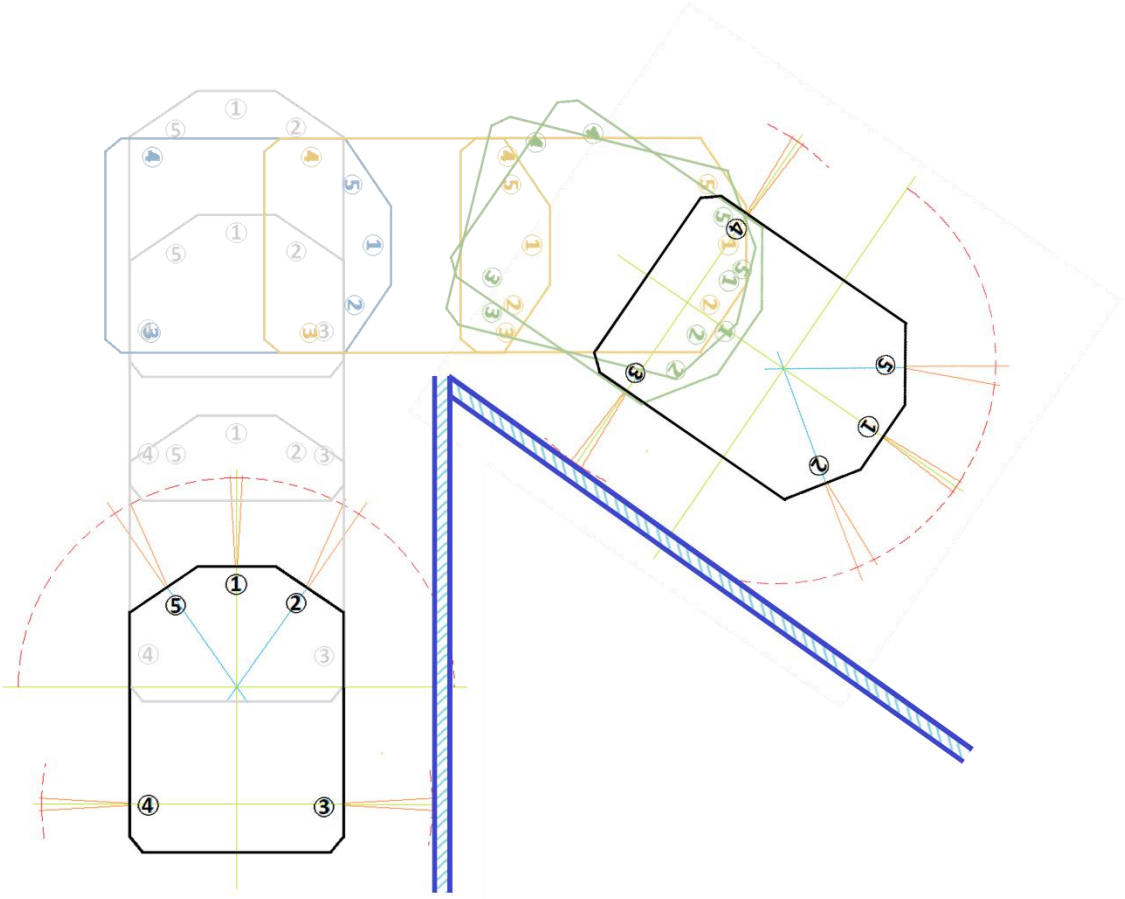


Figura 168. Reacció del robot enfront una paret de més de 270º.

En la figura 169 es pot veure el cas en què la paret que està seguint s'acabés, quan això passi rodejarà aquesta i seguirà endavant fins, si el sensor 3 detectà menys de 10 cm i el sensor 2 menys de 30 cm, s'iniciarà altra vegada el PID.

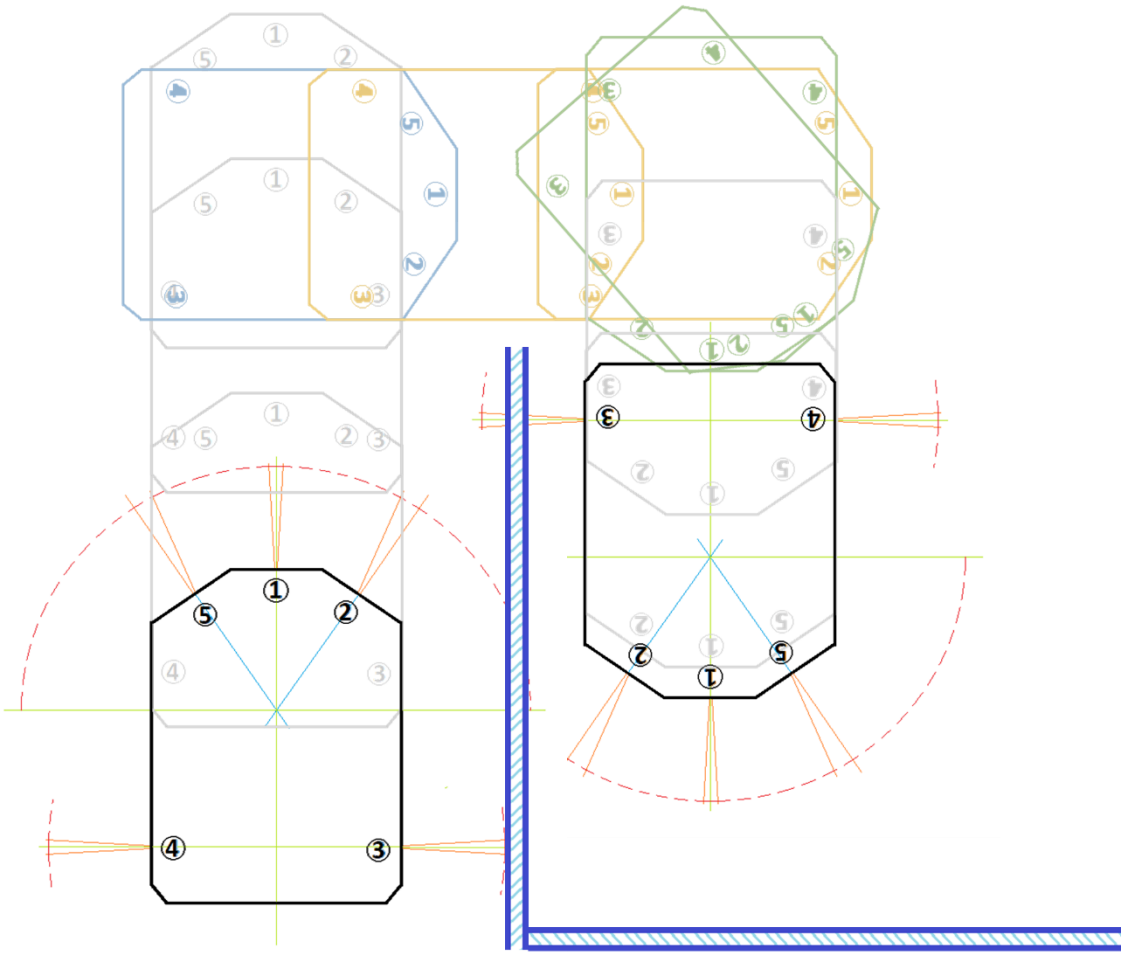


Figura 169. Reacció del robot davant una paret què acaba.

Els casos explicats anteriorment són els que es poden produir amb més freqüència a l'hora de seguir una paret, però ni han molts més que ni intentat imaginar-ho es trobarien. La millor forma d'afrontar tots els casos possibles és provar el robot en diferents situacions i a partir d'aquestes proves trobar i solucionar nous problemes.

3.3 Adquisició i transmissió de dades

El robot ha d'enviar un total d'onze mesures diferents; cinc preses pels sensors làser, dues pels codificadors, tres pel giroscopi i una del temps que ha passat des de l'inici de l'execució del programa. Els valors d'aquestes no són els que finalment s'utilitzaran, però s'envien aquestes per tal de no carregar el microcontrolador amb operacions matemàtiques llargues i complexes.

Per obtenir les mesures dels sensors làser es fa com s'ha explicat a l'apartat 2.3.4.2, mitjançant l'ús d'una llibreria que facilita enormement l'obtenció. Tot el codi necessari és agrupat en una sola funció la qual s'anomena *read_laser_sensors.* Les unitats que s'utilitzen són mm.

Dels codificadors obtenim un pols cada vegada que un objecte travessa l'optointerruptor, mitjançant una interrupció es va incrementant un comptador cada vegada que passa. Al cap d'un segon, és guarda el valor que indica els polsos per segon de la roda i es neteja el comptador. Aquest valor és el que s'envia pel port sèrie.

Amb el giroscopi també s'utilitza una llibreria que permet obtenir de forma directa els valors de guinyada, balanceig i capcineig. Aquests es representen en graus.

3.3.1 Protocol de comunicació sèrie

Per tal que el robot sigui capaç de mantenir una comunicació sòlida i estructurada amb el dispositiu receptor és necessari el disseny d'un protocol de comunicació. Aquest es basa en enviar paquets de paraules començats per dues paraules de sincronització, 0xFA i 0xFB. Fins que el dispositiu receptor no rebi aquestes dues paraules juntes no es començarà la lectura del paquet sencer. La següent paraula que es rebrà serà la que s'anomenarà *Lenght*, tal com diu el seu nom aquesta contindrà la suma de paraules que conté el paquet.

Per últim, abans de la transmissió de les dades, es calcularà i s'enviarà el *checksum*. Aquest es realitzà per tal de comprovar la integritat del paquet de dades, ja que pot ser

que un o diversos bytes de la seqüència canvi o inclús es perdin per complet duran l'enviament. Aquest càlcul es realitza analitzant paraula per paraula el paquet i guardant el resultat en dues paraules que seran enviades al final del paquet, tornant a realitzar el càlcul en el receptor per comprovar la igualtat, en el codi de continuació es pot veure la funció que s'usarà per al càlcul del *checksum*.

```
int CalcChecksum() {
  unsigned int c = 0;
  int n = lenght - 2;
  int posicion = 0;

  while ( n > 1) {
    c = c + ((pack[posicion] << 8) | (pack[(posicion + 1)]));
    n = n - 2;
    posicion = posicion + 2
  }

  if ( n < 0) c = c ^ pack[(posicion + 1)];

  return (c)
}
```

La freqüència de transmissió del paquet de dades serà de 500 ms. Com s'ha descrit abans, els codificadors aporten una mesura de polsos per segon, això significa que s'actualitzen els valors cada dues cadenes enviades, però enviar dues vegades la mateixa mesura és inútil i no aporta informació així què s'utilitzarà una regla de tres per tal de recalcular la mesura dels polsos:

$$Polsos\ a\ enviar = Polsoso\ actuals \cdot \frac{1000\ (ms)}{temps\ actual\ (ms)}$$

Per enviar les dades des d'Arduino s'utilitzarà la funció `sprintf()` per crear una cadena de caràcters amb tots els valors i finalment s'enviarà pel port sèrie utilitzant `Serial.println()`, és recomanable en Arduino, per tal de millorar el temps d'execució, utilitzar la menor quantitat possible de Serial.print() i en canvi, si es pot, enviar-ho tot en una sola cadena. A continuació es pot veure el codi que s'utilitzaria si només s'hagués d'enviar les mesures dels polsos dels motors.

```
char cadena[10];
sprintf (cadena, " %i, %i", polsos_d, polsos_e);
Serial.println(cadena);
```

S'ha de tenir en compte que la funció no permet escriure *floats* de forma directa, per tant s'haurà de fer una petita conversió per les mesures del giroscopi. A la Taula 11 es pot veure l'ordre en què s'enviaran les mesures així com el format que se'ls donarà dins de la funció `sprintf()` i com s'escriuran.

	Nom	TIPUS DE VARIABLE	EXPRESSIÓ EN `sprintf()`	REFERÈNCIA `sprintf()`
1	Paraula inici sincronització	string	`0xFA`	-
2	Paraula final sincronització	string	`0xFB`	-
3	Llargada	integer	`%i`	`length`
4	Checksum	integer	`%i`	`c`
5	Mesura sensor 1	integer	`%i`	`measure1.RangeMilliMeter`
6	Mesura sensor 2	integer	`%i`	`measure2.RangeMilliMeter`
7	Mesura sensor 3	integer	`%i`	`measure3.RangeMilliMeter`
8	Mesura sensor 4	integer	`%i`	`measure4.RangeMilliMeter`
9	Mesura sensor 5	integer	`%i`	`measure5.RangeMilliMeter`
10	Polsos motor dret	integer	`%i`	`polsos_d`
11	Polsos motor esquerra	integer	`%i`	`polsos_e`
12	Capcineig	float	`%d.%02d`	`(int)mpu.getRoll(),abs((int)(mpu.getRoll() * 100)% 100)`
13	Balanceig	float	`%d.%02d`	`(int)mpu.getPitch(),abs((int)(mpu.getPitch()*100)% 100)`
14	guinyada	float	`%d.%02d`	`(int)mpu.getYaw(),abs((int)(mpu.getYaw() * 100) % 100)`
15	Temps actual	unsigned long	`%lu`	`millis()`

Taula 11. Ordre de la cadena que s'enviarà per comunicació sèrie així com l'estructura d'aquesta.

3.4 Comunicació amb l'usuari

El robot no pot comunicar-se amb l'usuari fent ús del port sèrie perquè aquest està ocupat amb l'adquisició de dades, per tant s'utilitza una combinació entre els LEDs i l'altaveu per indicar els diferents estats.

A la Figura 170 es pot veure una representació del robot amb tots els LEDs encesos en blau, aquest és un estat que indica a l'usuari, juntament amb una curta melodia generada per l'altaveu, que la inicialització de tots els components s'ha realitzat correctament.

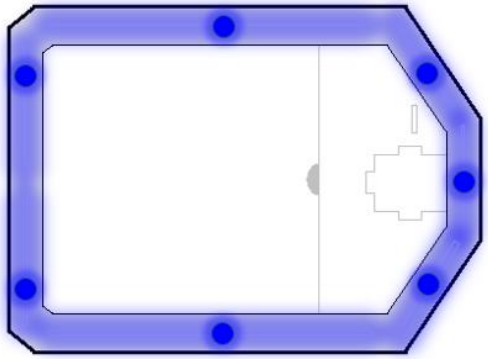


Figura 170. Representació del robot indicant que s'ha inicialitzat correctament.

Un cop Carpher s'ha inicialitzat, s'espera fins que l'usuari envia una resposta al sistema indicant si vol un control manual o automàtic. Per indicar aquest estat, s'encenen un per un els LEDs en color blanc i després, seguint el mateix ordre, s'apaguen.

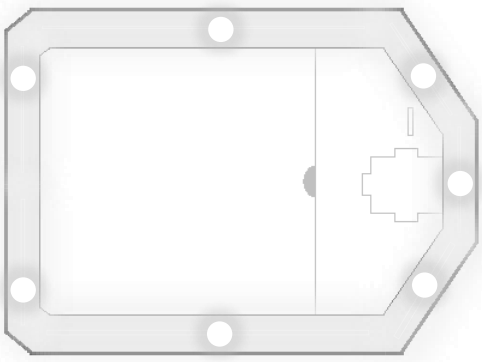


Figura 171. Representació del robot esperant una resposta de l'usuari.

A l'hora del mapatge, tant manual com automàtic, el robot mostra les mesures dels diferents sensors mitjançant quatre colors en els LEDs col·locats a la part superior dels sensors. A la Taula 12 es mostra un resum dels colors que s'utilitzen i el seu significat.

Color	Estat Mesura	Rang (per defecte)
Verd	Distància segura	> 15 cm
Groc	Distància de risc	15 - 10 cm
Vermell	Distància mínima	< 10 cm
S/C	Fora de Rang	> 430 cm

Taula 12. Significat dels colors dels LEDs quan el robot està en funcionament.

A la Figura 173 es pot veure la representació del robot quan tots els sensors mesuren una distància superior a 15 cm. Per altra banda, la Figura 151 mostra de forma simultània els quatre estats possibles en diferents LEDs.

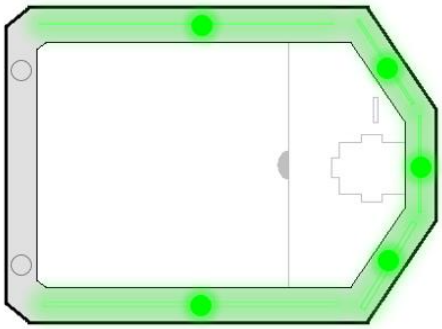


Figura 173. Representació del robot indicant que tots els sensors mesuren per sobre de 15 cm.

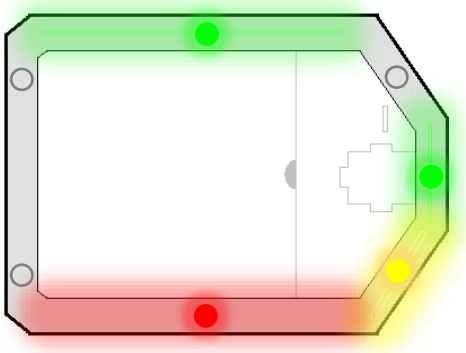


Figura 172. Representació del robot indicant diferents mesures en cada un dels sensors.

Estat de la bateria

Si l'usuari utilitza el robot sense tenir en compte la bateria, hi haurà un punt en què Carpher començarà a comportar-se de forma anormal i es reiniciarà automàticament quan el consum sigui gran. Per evitar aquest comportament s'avisa constantment a l'usuari de l'estat de la bateria mitjançant un LED. Els estats s'indiquen amb tres colors:

- **Verd** (Figura 151): La bateria es troba en un voltatge normal d'operació, entre 7,2 V i 6,2 V. El LED és mostra en verd.

- **Groc** (Figura 152): Avisa a l'usuari que queda poc per la descàrrega de la bateria (entre 6,2 i 6,0 V). Es continua amb un funcionament normal del sistema però el LED de la bateria parpelleja en groc i cada cinc segons s'emet un "bip" per l'altaveu.

- **Vermell** (Figura 153): La bateria s'ha esgotat i utilitzar-la podria provocar inestabilitat al sistema. S'aturen els motors i s'apaguen els LEDs, menys el de la bateria, i es reprodueix un "bip" amb l'altaveu de forma constant.

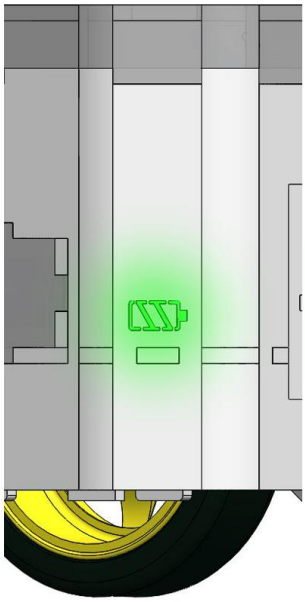


Figura 174. Bateria carregada per un funcionament òptim.

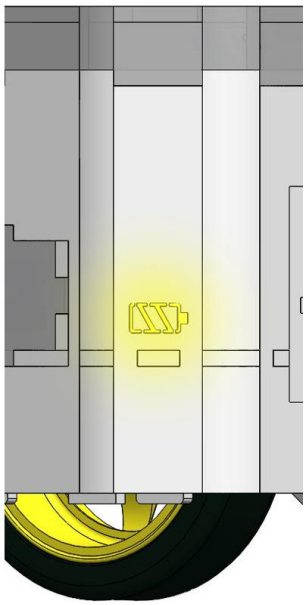


Figura 175. Bateria apunt d'esgotar-se.

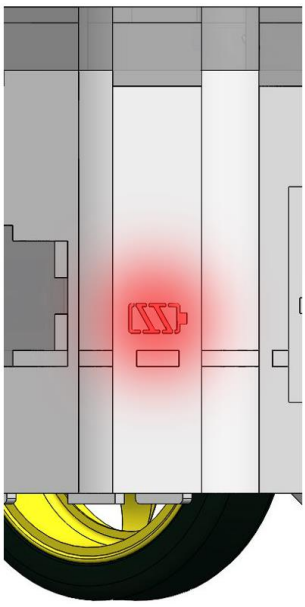


Figura 176. Bateria esgotada.

Tenint en compte que els LEDs són la part del sistema que més bateria consumeix, s'ha creat una variable dins del programa anomenada *intensitat* que permet regular, tal com diu el seu nom, la intensitat dels LEDs. Com menys intensitat, menys bateria es consumirà. Aquesta variable és un coma decimal que pot tenir valors entre 0.0 i 1.0. A l'hora de configurar els LEDs s'utilitza de la següent forma:

```
float intensitat = 0.5;

pixels.setPixelColor(7, pixels.Color(r * intensitat,
                         g * intensitat, b * intensitat));

pixels.show();
```

Això permet conservar el color però fent que el LED brilli menys i consumint menys bateria.

CAPÍTOL 4: MAPATGE EN TEMPS REAL

En aquest capítol s'introdueix el motor de creació de videojocs Unreal Engine 4, desenvolupat per Epic Games. S'explica el sistema de programació amb *blueprints* i l'entorn gràfic. No s'entrarà en detalls específics sinó que es veuran els aspectes més generals, ja que és un programa amb molt potencial i detall, i per tant molta complexitat, que en aquest projecte no s'arribarà a desenvolupar. S'explica l'obtenció de les dades del port sèrie així com el tractament d'aquestes. També es veu la creació dels diferents objectes gràfics que s'utilitzaran.

4.1 Preparació de l'entorn UE4

Per descarregar el motor és tan senzill com anar a la pàgina oficial d'Unreal Engine i fer clic al botó de *Download.* Un cop s'ha instal·lat i executat l'aplicació és mostra la pantalla de la Figura 177. Per crear un nou projecte s'han de seguir els passos marcats a continuació:

1.- Pestanya per crear un projecte nou.

2.- Tipus de programació. Si s'escull C++, es necessita el Visual Studio 2017.

3.- Inicia el projecte amb una plantilla o en blanc.

4.- Selecció de la plataforma per la qual està orientat el projecte, pot ser ordinador o mòbil.

5.- Qualitat en què es veurà l'editor del projecte.

6.- Definir si es vol començar amb contingut addicional (cubs, partícules, etc.).

7.- Ruta on es guardarà el projecte.

8.- Nom del projecte

9.- Botó per crear el projecte.

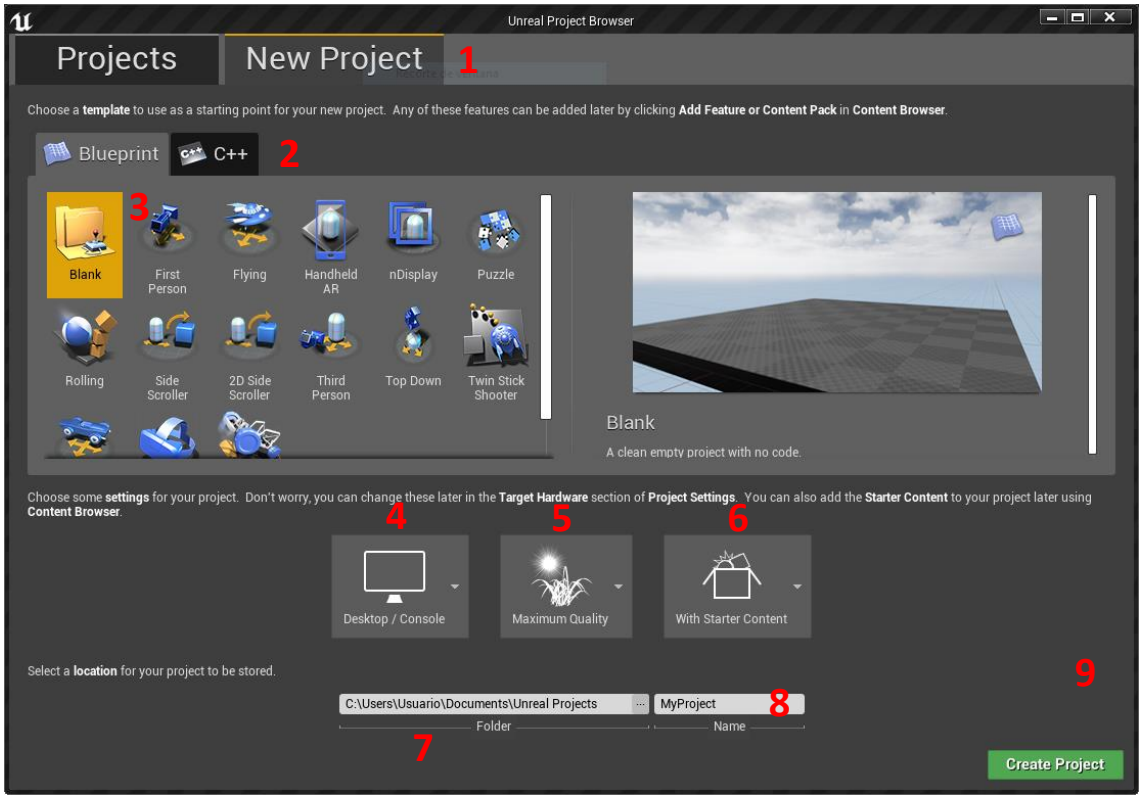


Figura 177. Pantalla de creació d'un nou projecte de UE4.

En aquest projecte s'escull com a base la plantilla adaptada pel control d'un vehicle i la plataforma escollida és ordinador. Un cop s'ha creat el projecte aquest s'obre automàticament (Figura 178). Les principals parts de l'editor es mostren a continuació:

1.- Espai on es mostren les opcions i característiques dels objectes.

2.- Explorador del projecte. Mostra tots els arxius que es vagin creant (textures, etc.).

3.-Escena actual o nivell carregat.

4.- Llista de tots els objectes que es troben en l'escena o nivell actual.

5.- Barra d'eines, aquí hi ha accés directe a guardar, crear nous objectes, configuracions, etc.

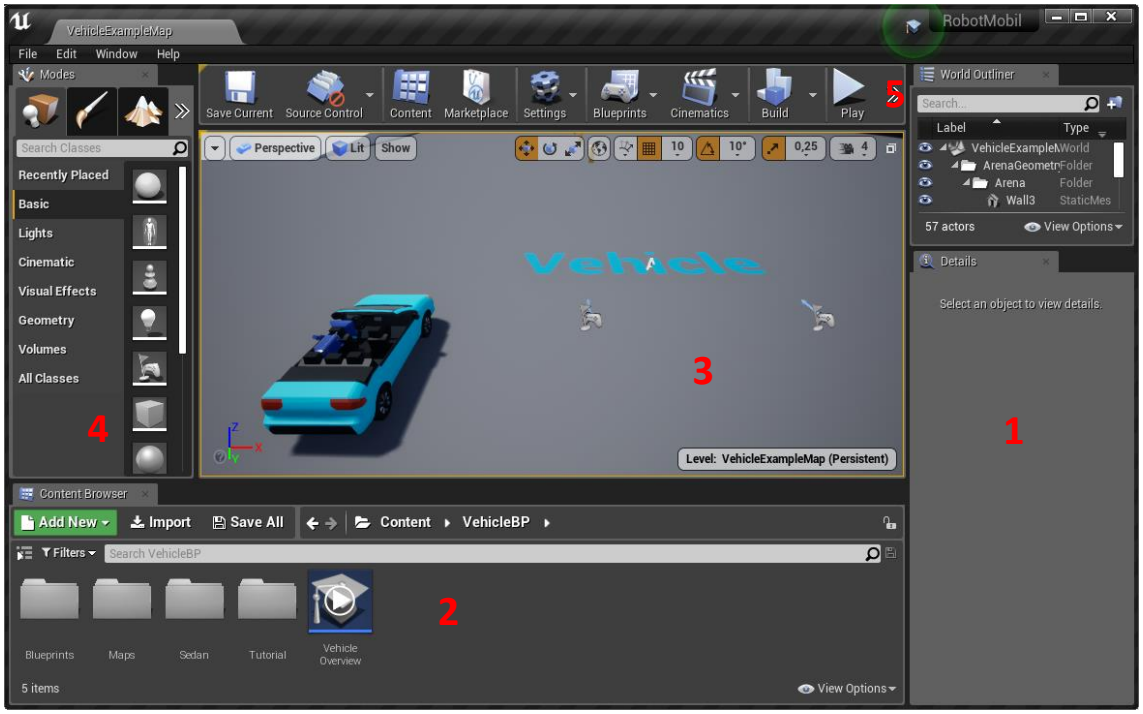


Figura 178. Captura de pantalla de l'editor UE4

4.2 Sistema de Blueprints

El sistema de blueprints permet desenvolupar programes de forma senzilla i visual, aquests són blocs que s'uneixen entre si mitjançant nodes. Tenen multitud d'opcions i funcionalitats com seria crear, implementar o modificar objectes, creació de malles, materials i personatges, etc. Aquest sistema no substitueix a la programació en C++, present en quasi tots els motors de videojocs, sinó que actua com a complement aquesta.

Tipus de blueprints

Existeixen diversos tipus de blueprints que permeten executar diverses funcionalitats. A continuació es nomenaran els més importants i es descriuran breument

Level blueprint

És l'espai on es crearan els blocs de programació del nivell actual. Cada nivell té el seu propi LevelBlueprint. Els esdeveniments del nivell (ex. Entrar al nivell, sortir del nivell, etc.) s'utilitzen per activar seqüències en forma de controls de flux. A la Figura 179 es pot veure la interfície d'un LevelBlueprint i la programació en blocs del moviment d'un objecte en iniciar el nivell.

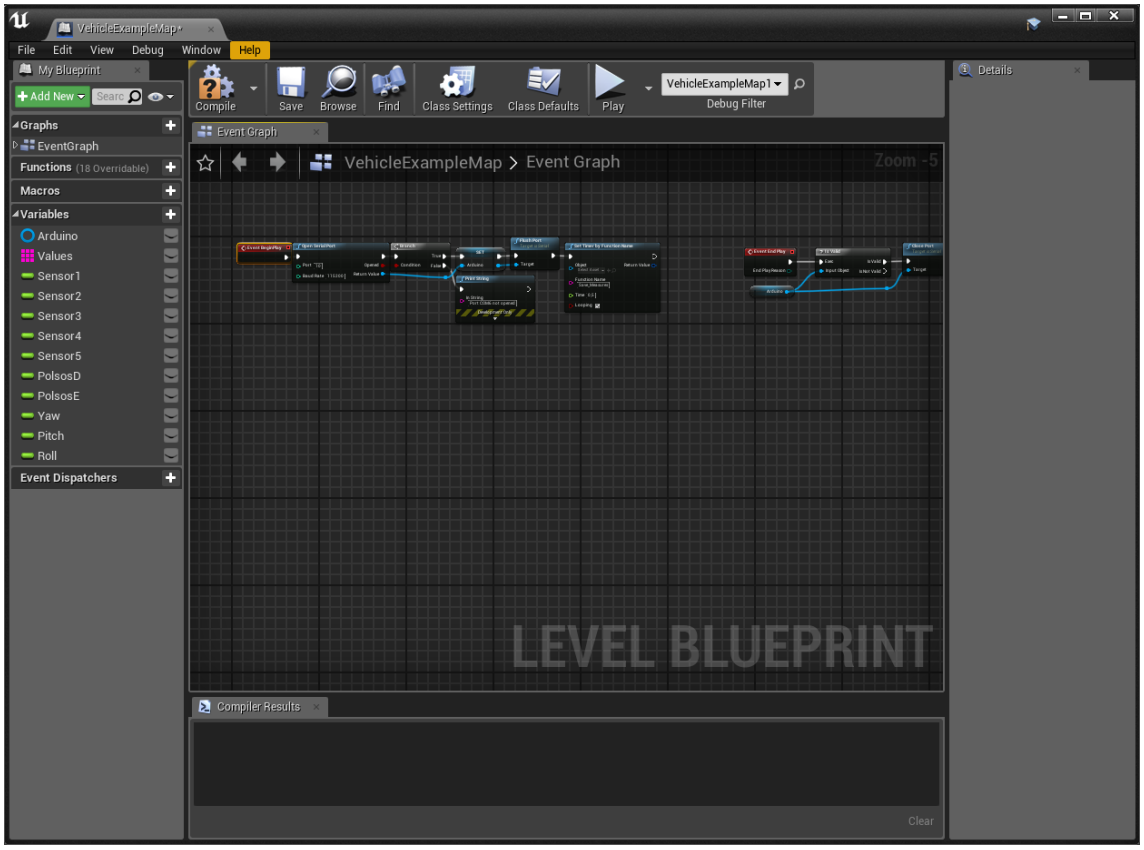


Figura 179. Captura de pantalla del LevelBlueprint.

Blueprint Class

Són blueprints creats a partir d'objectes presents en la part visual del programa, un exemple podria ser una porta. Per tal de fer que aquesta s'obri i es tanqui es crea un blueprint d'aquesta especificant la classe de "Pare" en la que es basarà el blueprint. En seleccionar la classe el blueprint heretera les propietats d'aquest i podrà desenvolupar les accions que se li assignin, en aquest cas obrir-se i tancar-se. Hi ha molts tipus de classes "pare" però en aquest projecte només se n'utilitzaran tres:

- Actor: objecte que pot ser col·locat en el món. Ex. Parets, terra, etc.
- Personatge: actor capaç de realitzar accions com caminar, córrer, saltar, etc. En aquest projecte equivaldria al robot que es mouria per la pantalla a partir de les mesures rebudes de l'Arduino.
- Jugador: és l'usuari capaç de controlar el món, en aquest cas l'usuari de Carpher.

Widget Blueprints

Amb aquest tipus de blueprints es poden crear menús.

4.3 Recepció de dades

Per rebre les dades del port sèrie es necessita instal·lar un *plugin* en el projecte d'UE4. Aquest porta per nom UE4Duino i està creat per Rodrigo Villani. Per descarregar-lo es pot fer directament des de la pàgina de GitHub. Per incloure'l en el projecte s'ha d'anar a la carpeta del PC on està descarregat l'UE4, i crear una nova carpeta. En aquesta carpeta nova s'ha de copiar la carpeta descomprimida del *plugin* que s'ha descarregat.

Per obrir el port sèrie quan es comenci el nivell s'uneix el node *EventBeginPlay* al d'*OpenSerial* (Figura 180). En aquest segon s'indica el port sèrie en el qual està connectat l'Arduino i el *baud rate*. En el cas d'aquest projecte, el port sèrie és el COM6 i el *baud rate* 115200, ambdós definits anteriorment en l'aplicació del Wixel de Polulu. Amb el *blueprint SET* creem un objecte del port sèrie que anomenarem Arduino. *FlushPort* llegeix tot el que hi ha al port sèrie per tal de netejar-lo.

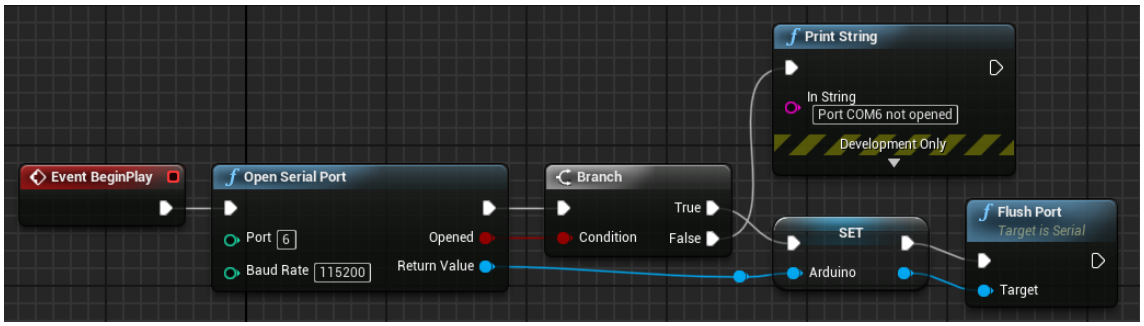


Figura 180. Blueprints per obrir un port sèrie determinat i especificació del tipus de lectura que es farà d'aquest.

És molt important tancar el port sèrie en parar l'execució del programa, ja que si es volgués tornar a engegar, o bé comunicar-se amb la placa a través d'Arduino, sortiria un missatge d'error indicant que el port sèrie està ocupat i no s'hi pot accedir. Per tancar-lo s'uneix el node d'esdeveniment *EventEndPlay* amb el de *CloseSerial* tal com es mostra a la Figura 181. *IsValid* evitaria que tanquéssim el port si aquest no s'ha pogut inicialitzar. Si per algun cas no es tanca el port, i surt el missatge d'error, es pot solucionar desconnectant l'USB de l'ordinador i tornant-lo a connectar.

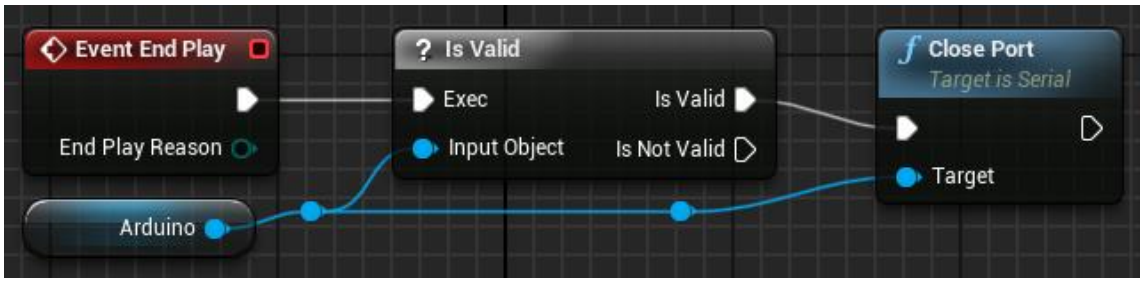


Figura 181. Blueprints per tancar el port sèrie una vegada se surt del programa.

Per tal de fer la lectura del port sèrie, cada 500 ms s'ha de crear un temporitzador. El blueprint *Set Timer by Function Name* permet crear un temporitzador i associar-lo a una funció. Cada vegada que s'arribi al temps establert la funció es posarà a 1. També dona l'opció que aquest es repeteixi indefinidament marcant la casella de *Loop.* A la figura 182 es pot veure el blueprint configurat per tal de llegir les mesures del port sèrie cada 500 ms de forma indefinida.

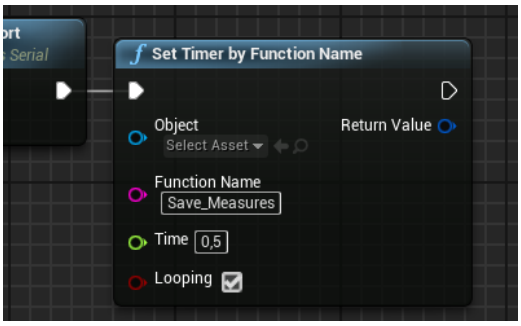


Figura 182. Creació d'un temporitzador que crida a una funció.

Com s'ha dit abans, la funció anomenada *Save_Measures,* que s'ha definit al temporitzador, és la que dóna inici a la lectura del port sèrie. Per llegir les dades s'utilitza el blueprint *ReadLine,* aquest llegeix tots els caràcters fins a trobar-se amb el caràcter de control "\r\n" que equival al final d'una cadena escrita amb `Serial.println()`. Com s'ha descrit anteriorment, els valors de les diferents mesures se separaven entre si mitjançant una coma, per tant, la cadena rebuda, se separa per aquestes i és guarda en una llista, és a dir, que ara cada mesura correspon a una posició de la llista. A la Figura 183 es poden veure les parts descrites.

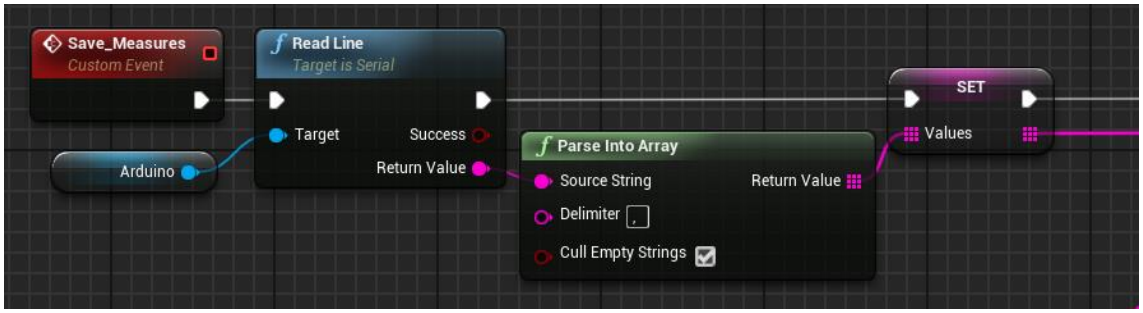


Figura 183. Configuració de blueprints per llegir caràcters del port sèrie i guardar-los en una llista separats per un delimitador.

A continuació validem els caràcters de sincronització. A la Figura 184 es pot veure com.

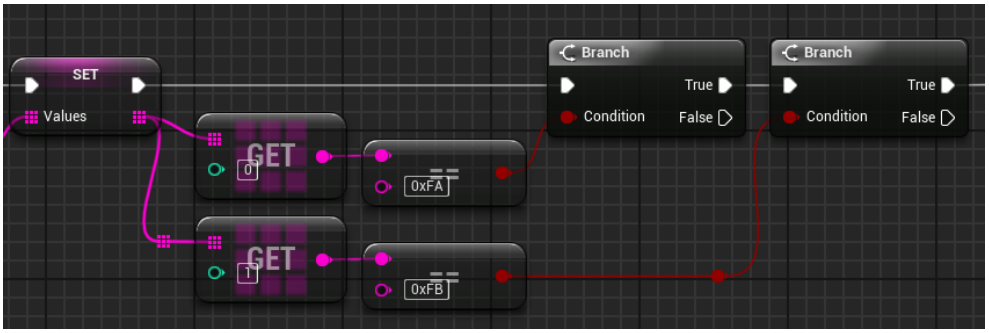


Figura 184. Validació dels caràcters de sincronització.

Per últim s'ha de guardar cada una de les mesures en una variable *float* o *int* per tal de poder realitzar els càlculs posteriors. Sabent la posició que té cada mesura, senzillament s'ha d'agafar aquesta i guardar-la en una nova variable de coma flotant. A la Figura 185 es pot veure el procediment seguit. Aquí no es pot veure el càlcul del cheksum, ja que aquest es farà en C++.

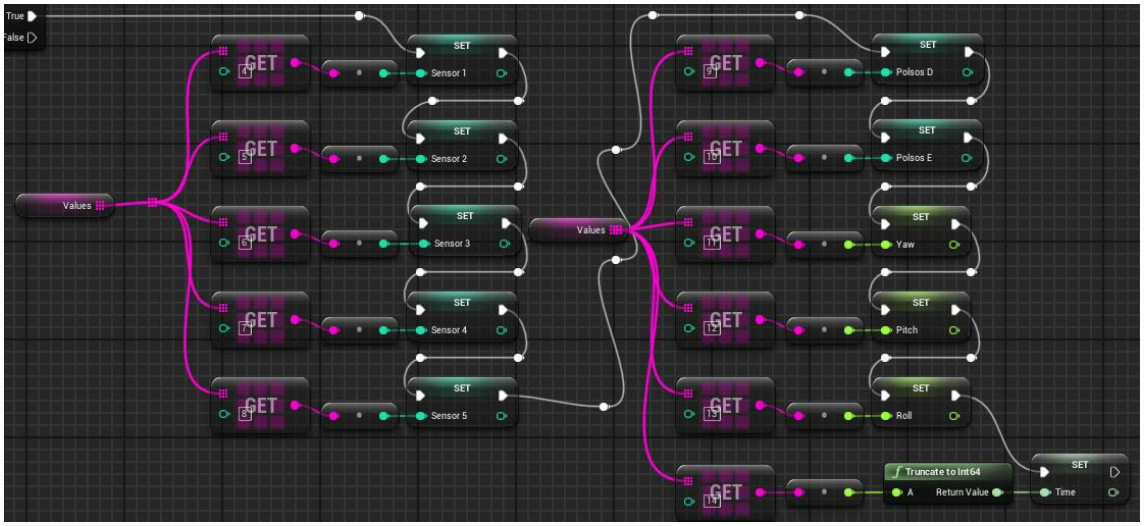


Figura 185. Sistema de blueprints per guardar en variables float les dades d'una llista de strings.

4.4 Càlcul i tractament de les dades

Per obtenir el punt (x, y) del robot s'ha seguit el model d'odometria. Es comença coneixent, $\mathrm{L} = 62\ \mathrm{mm}$ (distància del centre de l'eix de les rodes respecte d'una d'aquestes) i $\mathrm{r} = 33\ \mathrm{mm}$ (radi de les rodes).

Els càlculs necessaris per obtenir (x, y) del robot mòbil són:

1r.- Càlcul del diferencial de temps, equival al temps entre mostra i mostra:

$$T_s = t_{k+1} - t_k$$

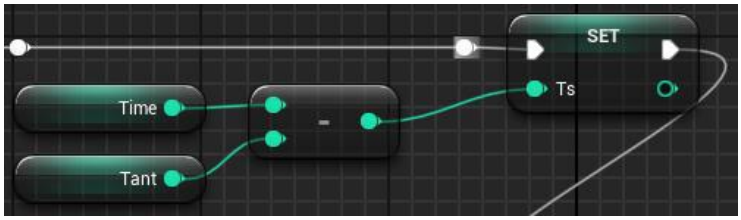


Figura 186. Càlcul amb nodes del diferencial de temps.

2n.- Expressar els polsos/segon de les rodes en graus

$$\dot{\theta}_{d\_g} = \frac{P_d \cdot 360}{N_c} \qquad \dot{\theta}_{e\_g} = \frac{P_e \cdot 360}{N_c}$$

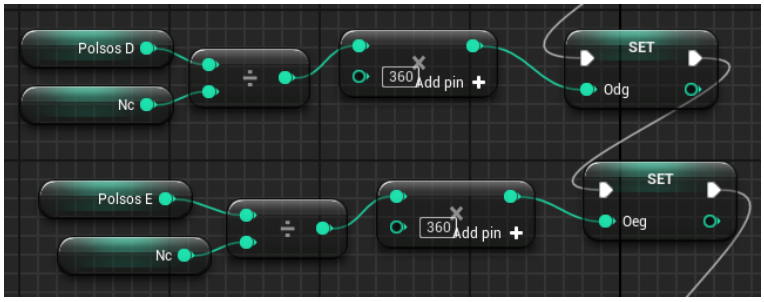


Figura 187. Càlcul amb nodes dels graus girats de les rodes a partir dels polsos/s.

3r.- Passar de graus a radians, obtindrem

$$\dot{\theta}_d = \dot{\theta}_{d\_g} \cdot \frac{\pi}{180} \cdot \frac{1}{T_s} \qquad \dot{\theta}_e = \dot{\theta}_{e\_g} \cdot \frac{\pi}{180} \cdot \frac{1}{T_s}$$

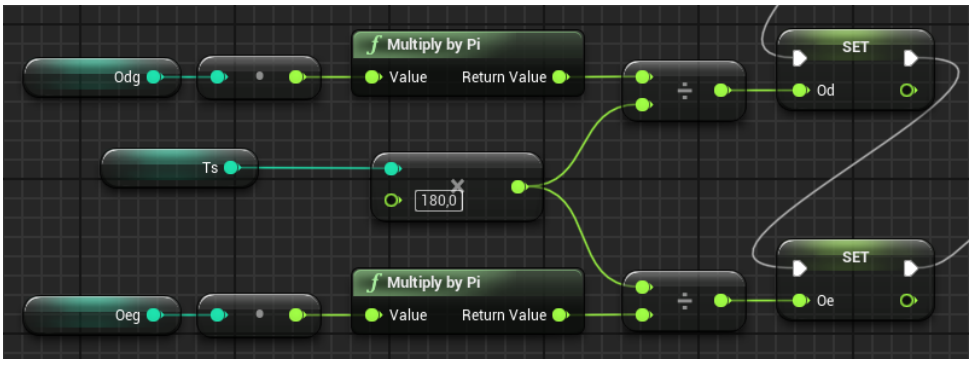


Figura 188. Conversió de graus a radians amb nodes.

4rt.- Càlcul de la velocitat lineal i angular del robot:

$$v = r\left(\frac{\dot{\theta}_d + \dot{\theta}_e}{2}\right) \qquad w = r\left(\frac{\dot{\theta}_d - \dot{\theta}_e}{2L}\right)$$

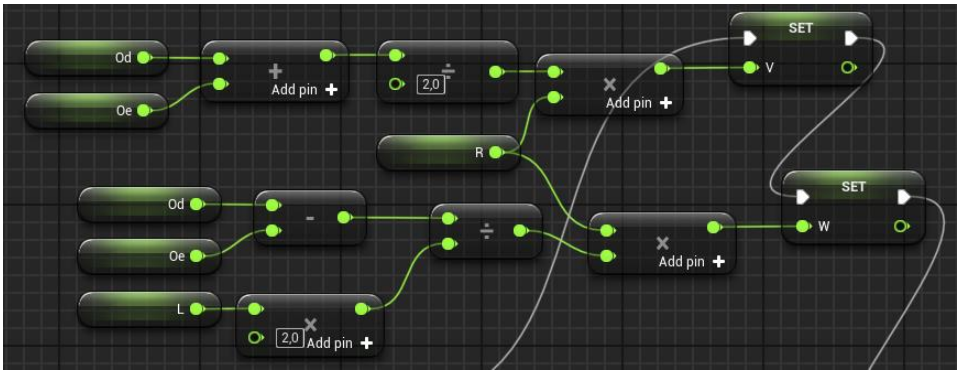


Figura 189. Càlcul amb nodes de la velocitat lineal i angular.

5è.- Càlcul de (x, y) en l'espai, càlcul de la velocitat angular:

$$x_{k+1} = x_k + v_k T_s \cos(\theta_k)$$

$$y_{k+1} = y_k + v_k T_s \sin(\theta_k)$$

$$\theta_{k+1} = \theta_k + T_s w_k$$

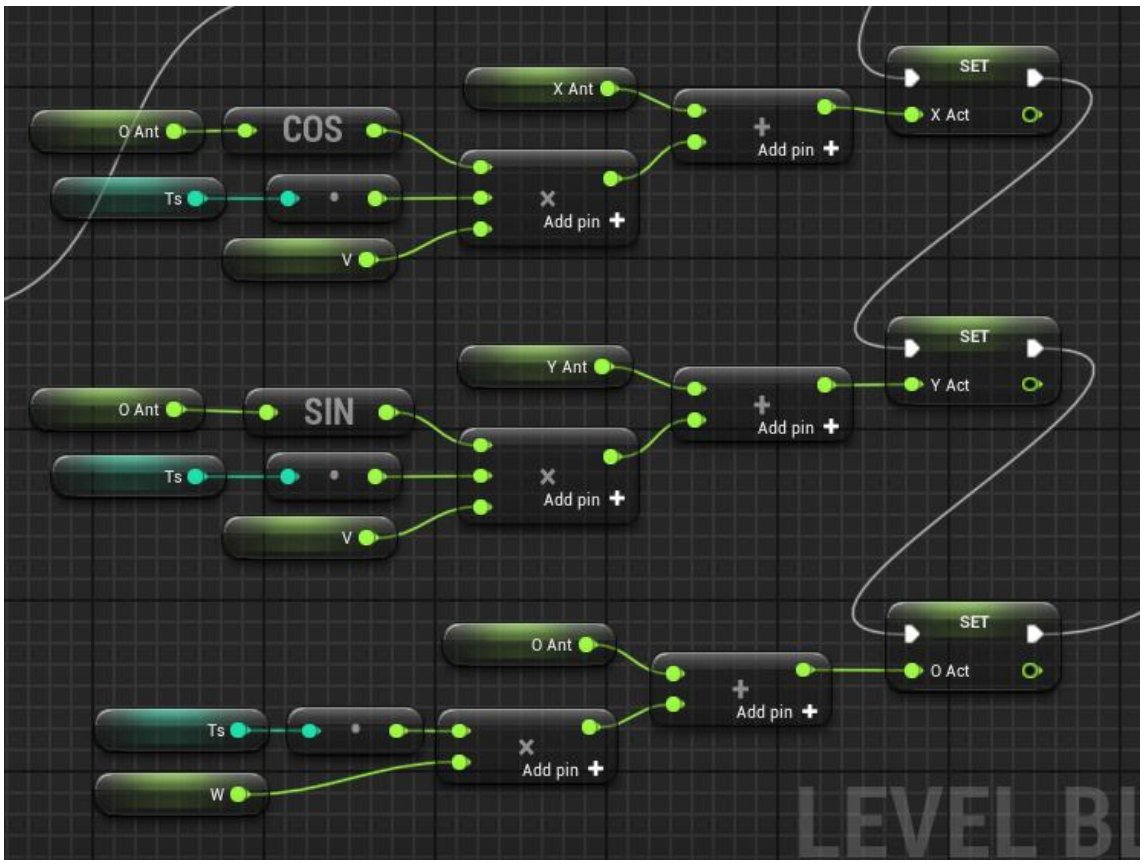


Figura 190. Càlcul amb nodes de les posicions x,y i de la velocitat angular

Finalment es guarden les variables actuals com a les anteriors per poder realitzar el següent càlcul.

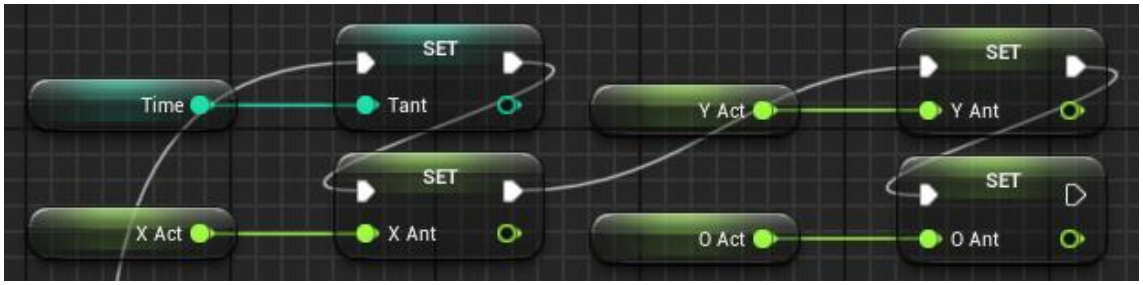


Figura 191. Assignació de les variables actuals com a les anteriors amb nodes.

4.5 Creació d'objectes

4.5.1 Pla

El món que es crearà serà de dimensions proporcionals al rang del Wixel de Polulu. Sabent que aquest pot arribar, de forma segura, fina a uns 5 m, es crearà un pla que farà de terra. Al voltant d'aquest es crearà una malla per evitar que el Carpher o la càmera que pot controlar l'usuari caiguin al buit. A la figura 192 es pot veure el món un cop s'ha creat el pla.

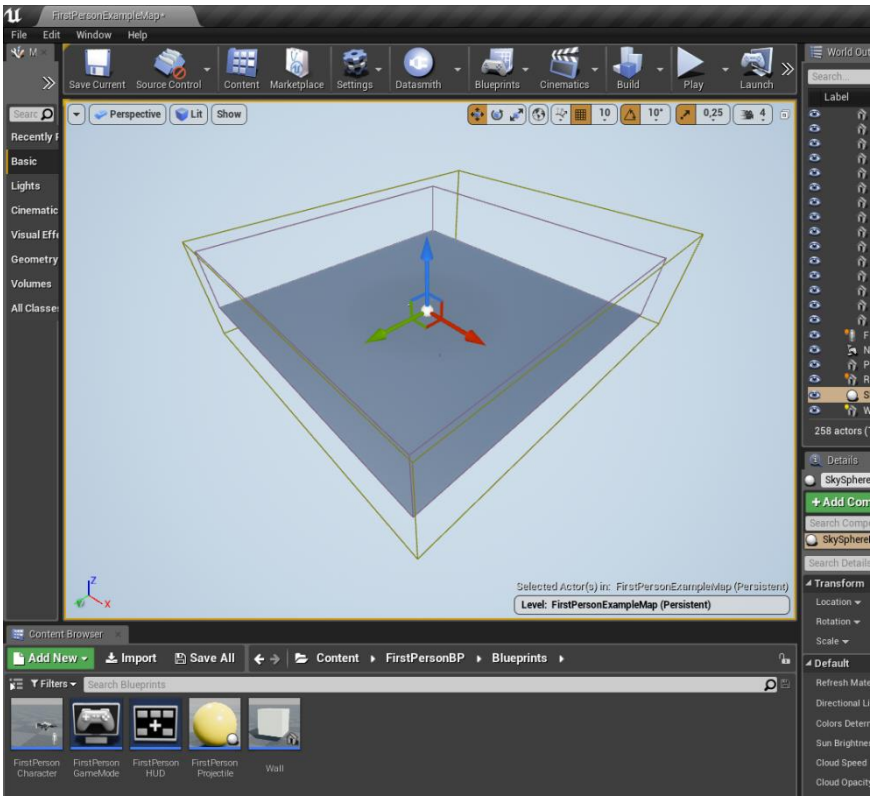


Figura 192. Pla creat a l'entorn d'UE4 què tindrà la funció de fer de terra.

4.5.2 Robot Carpher

Per col·locar el Carpher dins del món s'aprofitarà el model creat amb 3D en SolidWorks. UE4 proporciona l'eina Datasmith que permet importar objectes en diferents formats CAD. Per accedir aquesta s'han de descarregar dos *plugins* i registrar-se a la versió beta del complement, ja que encara està en fase de desenvolupament.

El model 3D ocupa molt i per tal d'evitar pes innecessari s'eliminaran tots els components que no siguin visibles des de l'exterior. Un cop eliminats es guardarà l'assemblatge com una sola part i ja es podrà importar al UE4. Per fer-ho se seleccionarà l'eina Datasmith CAD i seguidament es buscarà l'arxiu a l'ordinador. Un cop seleccionat s'ha d'escollir la carpeta on es vol guardar, o crear-ne una de nova, i finalment escollir les opcions d'importació. A la Figura 193 es veu el Carpher un cop importat dins l'entorn UE4.

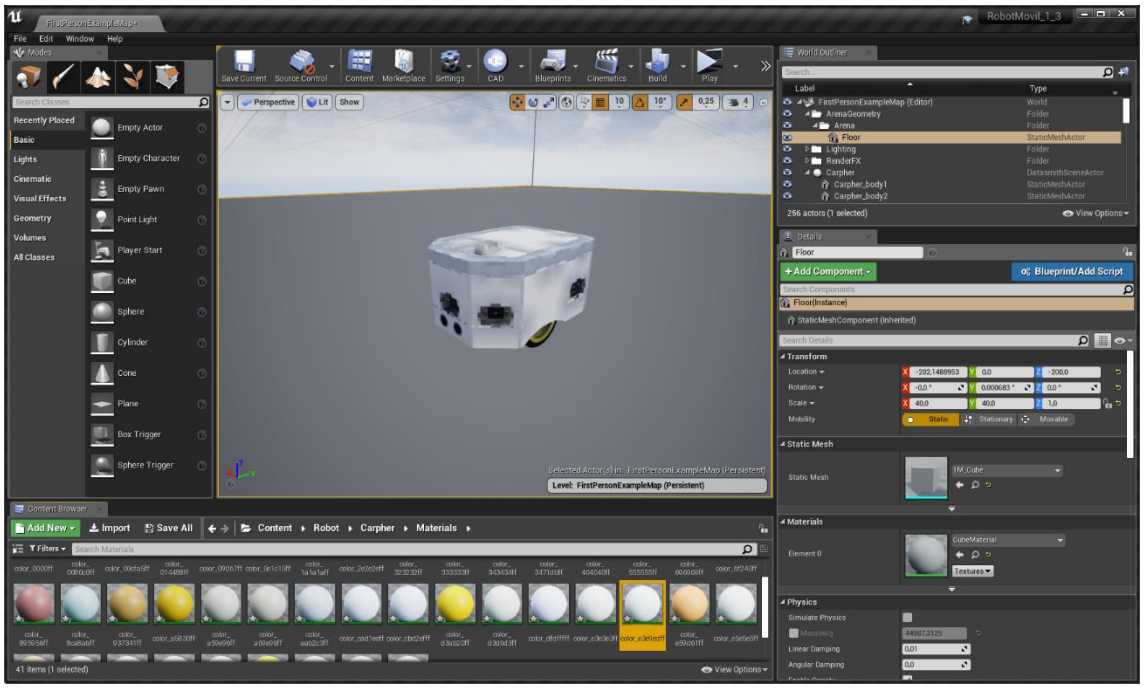


Figura 193. Model 3D de Carpher importat a UE4.

Per tal de poder moure l'objecte dins del món durant l'execució del programa s'hauran d'agrupar totes les parts i crear-ne un Blueprint.

4.5.3 Secció de paret

A partir de les mesures dels sensors làser es coneix la situació de les parets o objectes que envolten al robot, per tant per reproduir-les en el món senzillament es col·locarà un actor en forma de paret en les coordenades que aquests indiquen, utilitzant com a centre de referència el robot.

Per crear l'actor s'ha d'arrossegar des de la pantalla *Modes* al món, Figura 194.

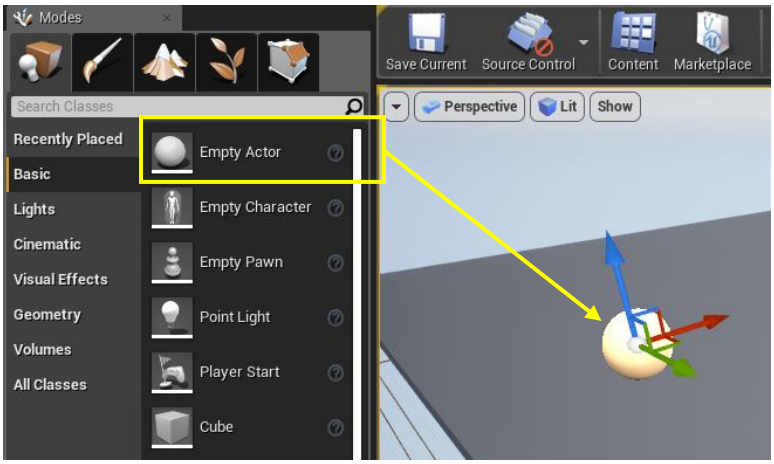


Figura 194. Creació d'un actor en l'entorn UE4.

Per defecte té forma d'esfera i el que es vol és que sigui en forma de cub. Per canviar-ho s'ha de clicar *+ Add Component* a la finestra de *Details* del actor, aquesta surt automàticament a la dreta de la pantalla en fer clic sobre l'actor, i finalment, seleccionar *Cube*. Després, en la pestanya *Transform,* s'escala el cub per tal que s'assembli més una paret. A la Figura 193 es pot veure l'actor un cop configurat amb la forma desitjada.

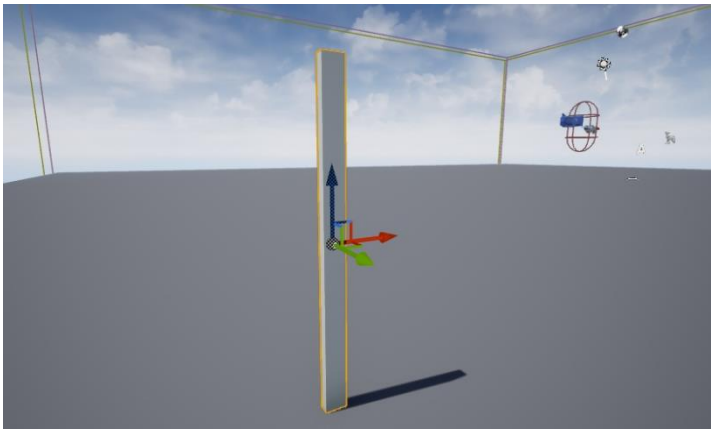


Figura 195. StaticMeshActor creat per simular les parets detectades pels sensors en l'entorn UE4.

Com s'ha fet abans amb el Carpher es crearà un blueprint del actor i ja estarà disponible per controlar i copiar tantes vegades com es vulgui en el món.

4.5.4 Càmeres

S'han creat fins a tres càmeres diferents per tal que l'usuari es pugui moure pel món. Per canviar entre elles s'ha de pressionar la tecla C.

1.- Càmera en primera persona que el jugador pot moure pel món utilitzant les tecles W, A, S i D. També se li permet saltar pressionant l'espai. (Figura 196)

2.- Càmera que segueix el robot durant el recorregut, no pot ser controlada. (Figura 197).

3.- Càmera fixa que mostra tot el món. (Figura 198).

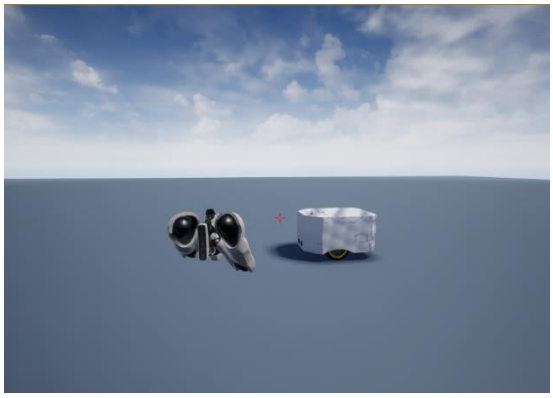


Figura 196. Càmera en la qual el jugador es pot moure lliurement per l'entorn creat.

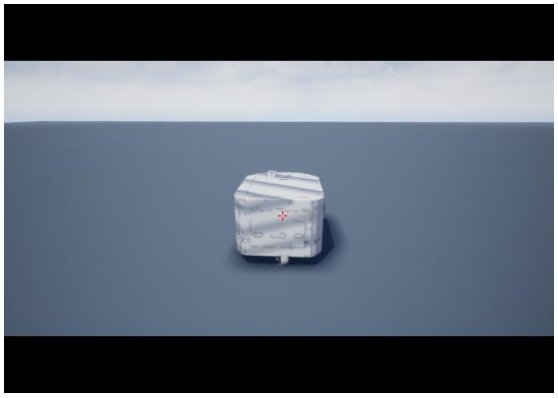


Figura 197. Càmera fixe en el robot què es mourà a la vegada que aquest ho faci.

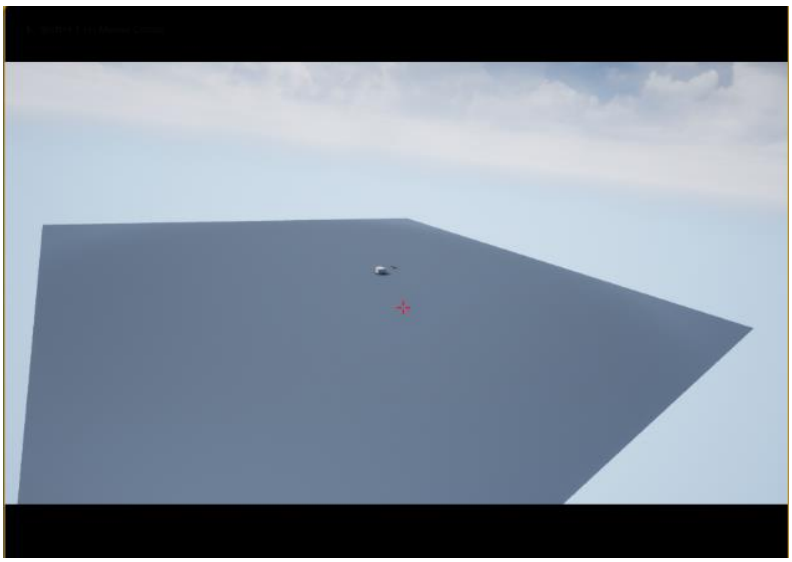


Figura 198. Càmera fixa en la que es pot veure tot el món.

4.6 Moviment dels objectes a partir de les mesures dels sensors

El Carpher pot moure's de posició de forma directa una vegada realitzats els càlculs d'odometria. Per altra banda, la rotació que s'hauria de fer a partir dels càlculs del giroscopi no es podrà fer, ja que no s'ha aconseguit uns valors nets. A la Figura 123 es poden veure els nodes utilitzats per tal de moure el robot. Els nodes per l'orientació tenen les variables de *yaw*, *pith* i *roll* com a constants a 0.

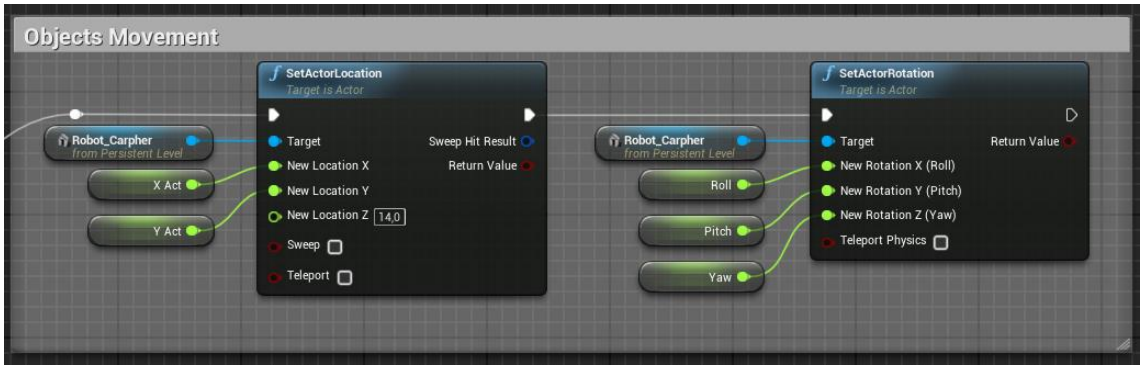


Figura 199. Nodes utilitzats per tal de moure el robot dins l'entorn d'UE4.

Les parets es posaran al món un cop s'hagi mogut el robot. S'utilitzarà com a punt de referència aquest i per cada sensor s'afegirà un *offset* diferent, aquest bé donat per la distància entre el centre del robot i el sensor. Per acabar se sumarà la mesura presa pel làser. A la Figura 200 es poden veure els nodes pel sensor 1.

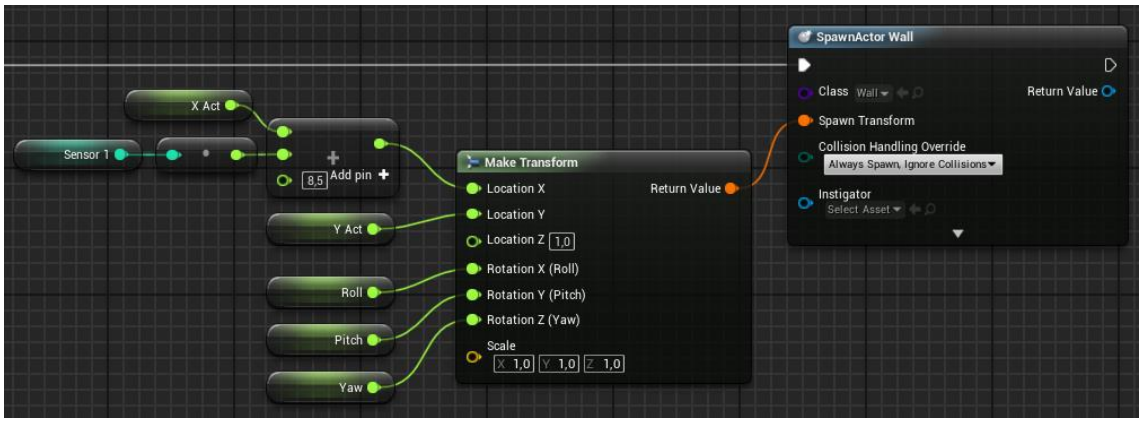


Figura 200. Nodes per col·locar en el món una paret a partir de la mesura del sensor làser 1.

CAPÍTOL 5: MILLORES

En aquest capítol es llisten les millores que es poden dur a terme en el projecte així com les coses que s'han intentat i necessiten una millora per tal de ser implementades.

5.1 Aspectes mecànics

- Millorar el disseny i les fórmules de curvatura per tal de poder imprimir les peces en metacrilat. Això també influiria en els càlculs d'odometria, ja que la fusta de contraplacat amb el pes tendeix a doblegar-se i fa que la posició dels motors no sigui la ideal.

5.2 Aspectes elèctrics i electrònics

- Col·locació d'una càmera per tal que l'usuari pugui veure en temps real l'entorn del robot.
- Col·locar un sensor de color per poder definir els colors de les parets i el terra que envolten al robot.
- Implementar un circuit que talles automàticament l'alimentació al detectar que la bateria és baixa.
- Estudiar més a fons el giroscopi per tal de poder treure'n valors més clars i poder-ho reflectir després a l'entorn UE4.

5.3 Programació en Arduino

- Fer que sigui capaç de tornar a l'origen del seu recorregut de forma autònoma.

5.4 Entorn Unreal Engine 4

- Implementar un sistema de memòria en el qual es pogués accedir als mapes anteriorment creats.
- Crear un menú per tal de poder veure les diferents opcions (control manual. Automàtic, etc.) de forma més visual.

CONCLUSIONS

En aquest projecte s'han enfrontat problemes típics en els robots mòbils, com serien la cartografia d'espais i el posicionament. S'ha vist que, un mal posicionament dels motors en la plataforma, crea errors en els càlculs odomètric difícils de corregir un cop processat. Respecte al mapatge un dels objectius era aconseguir un bon funcionament del robot, és a dir, que aquest fos capaç de recórrer tot l'entorn sense passar pel mateix lloc i evitant girs i moviments bruscos. S'ha arribat a la conclusió que, per aconseguir aquesta meta, s'ha de posar a prova el robot durant hores i anar corregint al mateix temps petits errors de moviment, així com afegir més codi per tal que pugui resoldre noves situacions.

Pel que fa a Arduino, ha demostrat ser una eina molt potent al ser fàcil d'utilitzar i amb un cost zero, això ha permès crear un robot mòbil autònom controlat per un microcontrolador i amb un pressupost inferior a tres-cents euros. El sistema Wixel de Polulu ha permès programar el robot i enviar dades a través del port sèrie sense haver de connectar el mòbil en cap moment a l'ordinador, això ha generat un gran estalvi de temps, ja que s'ha evitat haver d'anar connectant i desconnectant el robot a l'hora de programar.

Respecte a UE4 no s'ha arribat a desenvolupar el projecte lo suficient per a poder esprémer al màxim el motor. Tot i això, en un futur, es preveu continuar amb el desenvolupament i millora de Carpher, tant en hardware com en software, per tal d'aconseguir les metes que, per temps, no s'han pogut dur a terme.

BIBLIOGRAFIA

[1] Andrew Hurley. *Camera switching in Blueprints.* S/D. [Consulta 1 de Juliol de 2019]. Disponible a: < https://wiki.unrealengine.com/Camera_Switching_in_Blueprints>

[2] Andromina robot. *Encoder y Arduino. Tutorial sobre el módulo sensor de velocidad IR con el comparador LM393 (Encoder FC-03).* Nueva York, EE.UU: Novembre del 2018. [Consulta: 16 de Març de 2019]. Disponible a: <http://androminarobot.blogspot.com/2016/07/en-este-tutorial-mostramos-como-usar-el.html >

[3] Arduino. ARDUINO MEGA 2560 REV3. [Consulta 22 d'Abril de 2019]. Disponible a: < https://store.arduino.cc/mega-2560-r3>

[4] Arduino. *ARDUINO UNO REV3.* [Consulta: 22 d'Abril de 2019]. Disponible a: < https://store.arduino.cc/arduino-uno-rev3>

[5] Arduino. *Getting Started Introduction.* [Consulta: 21 d'Abril de 2019]. Disponible a: < https://www.arduino.cc/en/Guide/Introduction#>

[6] Arduino. *i2c_scanner.* Novembre del 2018. [Consulta: 16 de Març de 2019]. Disponible a: < https://playground.arduino.cc/Main/I2cScanner/>

[7] Army Technology. *TALON Tracked Military Robot.* Març de 2017. [Consulta: 9 de Juny de 2019]. Disponible a: < https://www.army-technology.com/projects/talon-tracked-military-robot/>

[8] Askik.com. *Cómo hacer un vehículo de tres ruedas [chasis robot].* Octubre de 2014. [Consulta: 11 de Juny de 2019]. Disponible a: <https://www.askix.com/como-hacer-un-vehiculo-de-tres-ruedas-chasis-robot.html>

[9] B.Beauregard traducció de J. Moyano. *Arduino PID – Guía de uso de la librería.* Abril del 2011. [Consulta 19 de Juny de 2019]. Disponible a: <http://brettbeauregard.com/blog/wp-content/uploads/2012/07/Gu%C3%ADa-de-uso-PID-para-Arduino.pdf>

[10] BostonDynamics. *Spot,* Setembre de 2017. [Consulta: 9 de Juny de 2019]. Disponible a: < https://www.bostondynamics.com/spot>.

[11] CORDIS. *Un pequeño paro para los robots y un salto gigantesco para la robótica.* Octubre del 2016. [Consulta 25 de Juny de 2019]. Disponible a: < https://cordis.europa.eu/news/rcn/126430/es>

[12] ElecFreaks. *Ultrasonic Ranging Module HC-SR04.* [Consulta: 7 de Març de 2019]. Disponible a: <https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf>

[13] G.Podnar. *Neptune.* Febrer de 1985. [Consulta: 11 de Juny de 2019]. Disponible a: <http://www.cs.cmu.edu/afs/cs/user/gwp/www/robots/Neptune.html>

[14] GeekFactory. *Alimentar el Arduino la guia definitiva.* Setembre 2018. [Consulta: 31 de Gener 2019]. Disponible a: <https://www.geekfactory.mx/tutoriales/tutoriales-arduino/alimentar-el-arduino-la-guia-definitiva/>

[15] GeekFactory. *Sensor ultrasónico HC-SR04 y Arduino.* Mayo del 2014. [Consulta: 7 de Març de 2019]. Disponible a: < https://www.geekfactory.mx/tutoriales/tutoriales-arduino/sensor-ultrasonico-hc-sr04-y-arduino/>

[16] Github, usuari flybrianfly. *MPU9250.* Desembre de 2018. [Consulta 26 de Juny del 2019]. Disponible a: < https://github.com/bolderflight/MPU9250>

[17] Hideakitai. *MPU9250 Library.* Juliol del 2018. [Consulta 25 de Juny de 2019]. Disponible a: < https://github.com/hideakitai/MPU9250>

[18] Juan Castromil. *Toyota Home Assistant Robot, el chaco robot doméstico.* Gener de 2018. [Consulta: 9 de Juny de 2019]. Disponible a: <https://clipset.20minutos.es/toyota-home-assistant-robot-el-chacho-robot-domestico/>

[19] Liquid Robotics. *Wave Glider Overview.* [Consulta: 9 de Juny de 2019]. Disponible a: <https://www.liquid-robotics.com/wave-glider/overview/>

[20] Luis Llamas. *Comprobar la integridad de datos en Arduino con checksum.* Agost del 2017. [consulta 3 de Juliol de 2019]. Disponible a: <https://www.luisllamas.es/arduino-checksum/>

[21] Luis Llamas. *Controlar motores de corriente continua con Arduino y L298N.* Mayo del 2016. [Consulta: 20 de Febrer de 2019]. Disponible a: < https://www.luisllamas.es/arduino-motor-corriente-continua-l298n>

[22] Luis Llamas. *El bus I2C en Arduino.* Mayo del 2016. [Consulta: 15 de Març de 2019]. Disponible a: <https://www.luisllamas.es/arduino-i2c/>

[23] Luis Llamas. *Medir temperatura y humedad con arduino y sensor DHT11-DHT22.* Març del 2016. [Consulta: 18 de Juny de 2019]. Disponible a: <https://www.luisllamas.es/arduino-dht11-dht22/>

[24] Luis Llamas. *Teoría de control en Arduino: El control PID.* Abril del 2019. [Consulta: 19 de Juny de 2019]. Disponible a: < https://www.luisllamas.es/teoria-de-control-en-arduino-el-controlador-pid/>

[25] Luis Llamas. *Tipos de motores rotativos para proyectos de Arduino.* Agost del 2016. [Consulta: 18 de Febrer de 2019]. Disponible a: <https://www.luisllamas.es/tipos-motores-rotativos-proyectos-arduino/>

[26] Matthew Hallberg. *Room mapping Arduino robot with Unity 3D.* Agost del 2018. [Consulta 23 de Gener de 2019]. Disponible a: <https://www.instructables.com/id/ROOM-MAPPING-Arduino-Robot-With-Unity-3D/>

[27] Miguel Vedoya. *Introducción a Unreal Engine, C++ y Blueprints.* Novembre de 2017. [Consulta 28 de Juny de 2019]. Disponible a: <https://www.miguelvedoya.com/2017/11/28/introduccion-a-unreal-engine-c-y-blueprints/>

[28] Real Academia Española. *Robot* (s.f.a.). . [Consulta: 18 de Febrer de 2019]. Disponible a: <https://dle.rae.es/?id=WYRlhzm>

[29] Robert Alley. *¿Cuál es la diferencia entre 4WD y 2WD?.* Novembre del 2016. [Consulta: 18 de Febrer de 2019]. Disponible a: < https://www.puromotores.com/13103814/cual-es-la-diferencia-entre-4wd-y-2wd>

[30] Roberto Villani. *UE4Duino.* Abril de 2019. [Consulta 28 de Juny de 2019]. Disponible a: <https://github.com/RVillani/UE4Duino/releases>

[31] Robotnik. *Robot móvil SUMMIT-XL.* [Consulta: 9 de Juny de 2019]. Disponible a: < https://www.robotnik.es/robots-moviles/summit-xl/>

[32] ROVINA. Febrer del 2013. [Consulta 25 de Juny de 2019]. Disponible a: < http://www.rovina-project.eu/>

[33] TuElectrònica.es. *Qué es la protoboard (breadboard).* Febrer del 2016. [Consulta: 5 de Febrer 2019]. Disponible a: <https://tuelectronica.es/que-es-la-protoboard/>

[34] Unreal Engine 4. *Blueprints Visual Scripting.* S/D. [Consulta 1 de Juliol de 2019]. Disponible a: < https://docs.unrealengine.com/en-US/Engine/Blueprints/index.html>

[35] Unreal Engine 4. *Datasmith Overview.* S/D. [Consulta 1 de Juliol de 2019]. Disponible a: <https://docs.unrealengine.com/en-US/Studio/Datasmith/Overview/index.html>

[36] Vidaltec.com. *Historia de Arduino.* [Consulta: 21 d'Abril de 2019]. Disponible a: <http://www.videltec.com/arduino/historia_arduino>