Universitat de Girona
Escola Politècnica Superior

Treball final de grau

Estudi: Grau en Enginyeria Electrònica Industrial i Automàtica

Títol: Robot i pràctiques per aprendre a programar sistemes encastats

Document: 1. Memòria

Alumne: Anna Planas Bahí

Tutor: Albert Figueras Coma
Departament: Enginyeria Elèctrica, electrònica i automàtica
Àrea: Enginyeria de sistemes i automàtica

Convocatòria (mes/any): Juny/2020

ÍNDEX

1 INTRODUCCIÓ

1.1 Antecedents

Un dels projectes que es van realitzar durant l'estada a l'entorn laboral al Grup Agents Reserch Laboratory (ARLab) de la Universitat de Girona, va ser el Disseny i realització d'un Ardubot millorat. En aquest projecte es va dissenyar el circuit electrònic i la placa de circuit imprès de l'Ardubot 4.1, que es va enviar a fabricar però no es van provar. Posteriorment a l'assignatura d'Aplicacions Industrials dels Microcontroladors, es van implementar el disseny de l'Ardubot 4.1 en alguna de les pràctiques. Per altra banda, cada dia és mes freqüent trobar elements quotidians connectats a internet i que es poden controlar del de qualsevol punt del món, és el conegut internet de les coses.

1.2 Objecte

Per aquest motiu s'ha decidit adaptar l'últim disseny de l'Ardubot 4.1 per facilitar la connexió del dispositiu a internet i que a més a més permeti fer pràctiques de més alt nivell, per exemple incorporant un Raspberry Pi. També s'aprofitarà per fer millores mecàniques en el sistema motor-roda i aprofitar les característiques especifiques del nou microcontrolador. S'estudiaran mètodes per allargar l'autonomia del robot, ja sigui amb un altre sistema d'alimentació o reduint el consum. Per altra banda s'actualitzaran les pràctiques actuals i se'n dissenyaran de noves per a fer comunicacions entre el robot i internet.

1.3 Abast

Així doncs, s'haurà de dissenyar una placa que permeti adaptar el nou controlador al robot existent, de manera que es substituirà l'Arduino Uno per aquest nou dispositiu. Amb el nou controlador s'haurà de poder actuar sobre tots els actuadors i s'hauran de poder llegir tots els sensors de l'Ardubot 4.1. S'haurà d'adaptar el sistema d'alimentació al nou controlador i es faran les millores mecàniques imprimint en 3D una roda amb el codificador rotatiu incorporat. També s'han d'actualitzar les pràctiques actuals i se n'han de crear de noves per a poder ensenyar als alumnes a utilitzar el nou dispositiu i a comunicar-se amb internet.

2 ARDUBOT 4.1

L'Ardubot 4.1 és un robot didàctic, pensat per a facilitar l'aprenentatge de la programació, sobretot de sistemes encastats. És l'evolució de L'Ardubot 3.2, crea des del grup de recerca de la Universitat de Girona Agents Research Lab (Arlab), que treballa en robots de rescat i des del departament d'EEEA. La motivació de la seva creació va ser la de realitzar tallers als instituts sobre robots de rescat i com a eina d'aprenentatge en electrònica, informàtica, robòtica... Posteriorment es va adaptar per a fer classes en l'àmbit universitari, a l'assignatura d'Aplicacions industrials dels microprocessadors de la UdG. En l'última fase es van solvatar error de disseny i es van afegir pins lliures per ampliar funcionalitats. El creador del robot original va ser en Rafel Hesse juntament amb l'Albert Figueras. La versió amb què es treballarà en aquest projecte es va dissenyar durant l'estada en l'entorn laboral feta per l'autora del prenent projecte (Anna Planas) en el grup de recerca de la UdG ARLab.

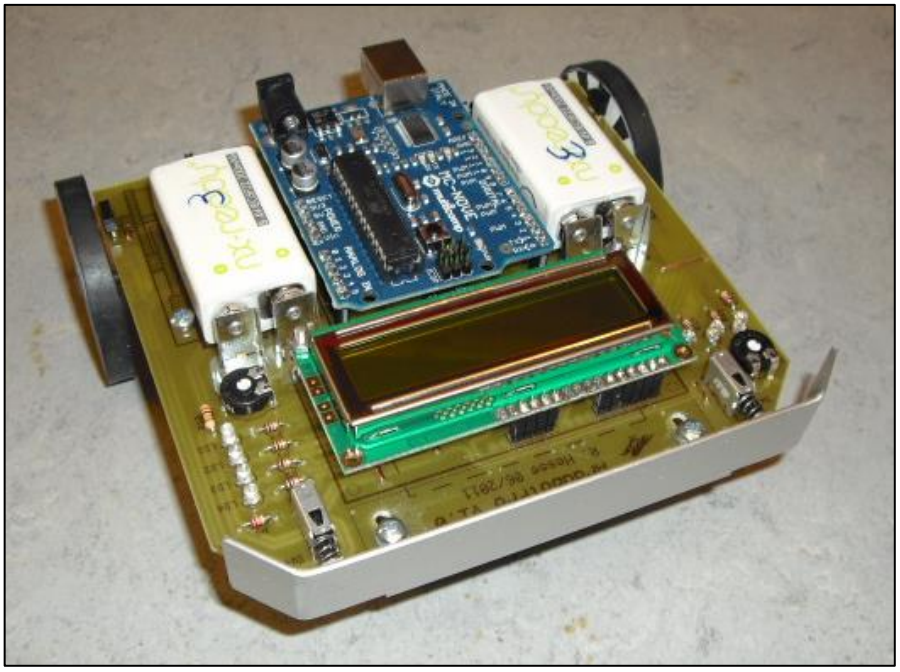


Figura 1. Ardubot 4.1

L'Ardubot és una eina molt interessant vist des de dos punts de vista diferents. Des del punt de vista d'un nen que comença a programar per primera vegada a la seva vida, tenir un robot que pugui programar de manera fàcil, amb el que pugui interaccionar, és més atractiu que tenir únicament un programa a una pantalla. Un robot és un element físic, que a part d'ajudar a assimilar conceptes d'una manera visual ho fa amb tots els altres sentits. A més a més és fàcilment programable pels més petits, ja que existeixen moltes interfícies de programació per blocs per a Arduino (el microcontrolador de l'Ardubot). Analògic i rellotge

Per altra banda, des del punt de vista d'un informàtic que vol aprendre a programar sistemes encastats, és molt recomanable començar per un conjunt assemblat. Programar sistemes encastats té diferents dificultats: la memòria, els recursos, la bateria (si en té), l'espai... Si a més a més s'afegeix una mala connexió (per culpa d'un cable que ha saltat o una placa de proves malmesa), problemes amb la instrumentació dels sensors o actuadors, problemes produïts per sobre consums d'intensitat, etcètera; aquest aprenentatge augmenta molt de dificultat. Si es té un robot compacte, la major part dels errors seran del programa realitzat i no del sistema físic.

2.1 Bases mecàniques i electròniques

La part mecànica es caracteritza per tractar-se d'un robot diferencial, això significa que té dues rodes no direccionables i un punt de suport. Consta de dues plaques, la principal i la del seguidor, unides de forma mecànica i elèctrica. La placa principal és la que conté la gran part dels sensors, tots els actuadors i també el microcontrolador. La placa del seguidor conté els sensors de contrast (per a poder fer un seguidor de línies) i la seva instrumentació. Aquesta placa està situada a la part davantera del robot i també s'hi uneix el punt de suport. L'aparell motriu del robot són dos motors de corrent continua, de 12 Volts i amb control de velocitat. Aquests motors van units a la part inferior de la placa principal. A l'eix primari del motor hi ha dos engranatges reductors que redueixen la velocitat de l'eix secundari, on es troba la roda.

El seguidor de línia consta de 4 sensors de contrast situats a la Placa del seguidor. Amb les dades que proporcionen els sensors de contrast es poden fer actuar els motors que accionen les dues rodes motrius. Per facilitar aquesta feina hi ha quatre LEDs, cada un assignat a un dels sensors, que s'encenen quan el sensor detecta un color reflectant.

A cada roda hi ha un adhesiu amb franges radials blanques i negres, que conjuntament amb un sensor de contrast creen un codificador rotatiu. Així doncs es pot conèixer amb facilitat i exactitud al desplaçament i posició de cada una de les rodes. Fer-ho amb un adhesiu permet explicar de manera visual com funciona un codificador rotatiu i fins i tot modificar-lo per a obtenir resultats diferents.

A la part davantera del robot hi ha dos sensors de contacte, protegits per una barrera mòbil metàl·lica. Aquest conjunt està ideat per a ser un para-xocs, així es pot saber si s'ha col·locat amb un objecte. Tot i això aquests elements es poden utilitzar amb altres finalitats. Per acabar amb els elements d'entrada falta comentar que hi ha un polsador, l'element més bàsic per a poder interacciona amb un robot. A més a més es pot saber l'estat de les bateries.

Per altra banda, existeixen diferents actuadors. Als paràgrafs anteriors ja s'ha parlat del sistema motriu del robot, dues rodes unides a dos motors de corrent continua de velocitat variable i de control independent. Aquests motors se situen a la part posterior i inferior de la placa principal, a dreta i a esquerra. L'Ardubot també disposa de tres LEDs que es controlen des del microcontrolador. Un LED és vermell, l'altre és taronja i l'últim és verd. El LED és un dels elements més bàsics per a tenir una comunicació maquina-home. Finalment hi ha una pantalla LCD de tecnologia I2C, amb una matriu de dos per setze caràcters.

A la Figura 2 es mostra la distribució que tenen els diferents elements de l'Ardubot. Es poden observar amb diversos colors els diferents tipus de sortides i entrades que hi ha. Les fletxes indiquen si és una entrada pel microcontrolador, una sortida o un bus de comunicacions dels dos sentits.

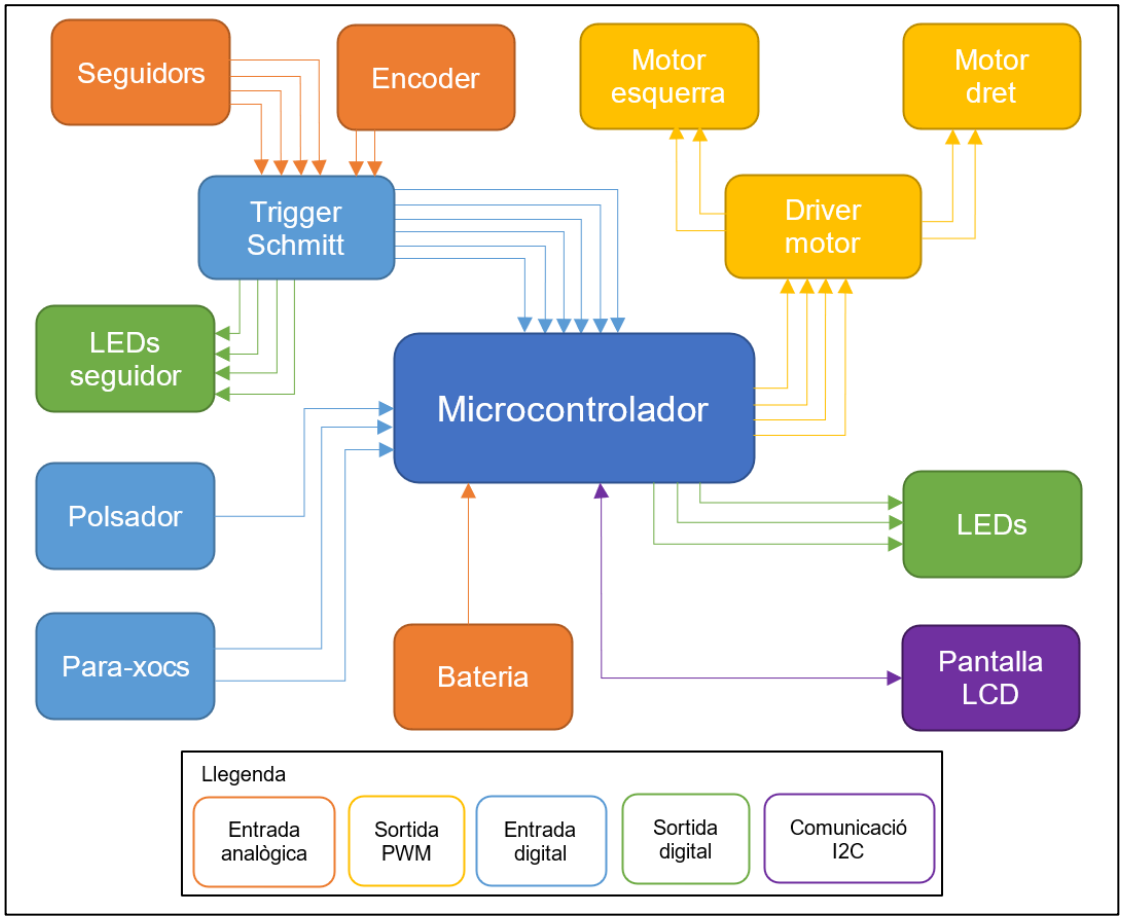


Figura 2. Diagrama dels diferents elements d l'Ardubot.

El microcontrolador de l'Ardubot és un ATmega328P amb el format d'Arduino Uno. Permet una fàcil manipulació i està molt protegit, cosa que el fa perfecte com a element educatiu. A més a més l'IDE de programació és gratuït i amb un munt d'exemples i ajudes de la comunitat.

Per a programar la placa únicament es necessita un ordinador amb l'Arduino IDE i un cable USB A a USB C. L'alimentació de l'Ardubot es pot fer a través de l'Arduino, si aquest està connectat a l'ordinador, o a través d'un parell de bateries prismàtiques tupis PP3 de 9 V i 800 mAh cada una.

En el present projecte es vol crear una placa adaptativa que ampliï les prestacions electròniques i de connectivitat del robot, per a crear pràctiques de més alt nivell. Per fer-ho es planteja substituir o complementar l'Arduino Uno per un altre element que permeti la connexió a internet. També algunes millores mecàniques com pot ser la roda impresa en 3D i l'estudi d'una millora en l'autonomia del robot.

3 HARDWARE PER A LA CONNEXIÓ A INTERNET

Actualment, amb l'expansió de la internet de les coses, existeixen molts de dispositius que permeten connectar qualsevol tipus d'elements a internet, des d'endolls fins a rellotges. En el cas que ens ocupa, al tractar-se d'un robot mòbil, aquesta connexió s'ha de fer mitjançant tecnologia sense fils i amb un consum baix d'energia. Amb les premisses anteriors s'han buscat diferents opcions.

Primer de tot es va pensar amb un adaptador de l'Arduino, que únicament li dóna connexió a internet a través del WiFi, però això no ampliaria la resta de prestacions, per tant es va descartar ràpidament. Seguidament es van pensar alternatives que substituïssin l'Arduino Uno que s'utilitza actualment. Es van trobar dues tecnologies molt populars i amb més prestacions que complien el requisit de la connexió a internet sense fils, per una banda l'ESP32 i per altra la Raspberry Pi. El benefici que tenen aquestes dues tecnologies enfront d'altres de menys conegudes és que existeix una comunitat molt gran a internet que dòna suport en fòrums, llibreries, programes d'exemple... fet que beneficia l'autoaprenentatge dels alumnes més avançats. Una altra placa similar a la Raspberry i molt coneguda però sense tanta comunitat és la BeagleBoard, de Texas Instruments.

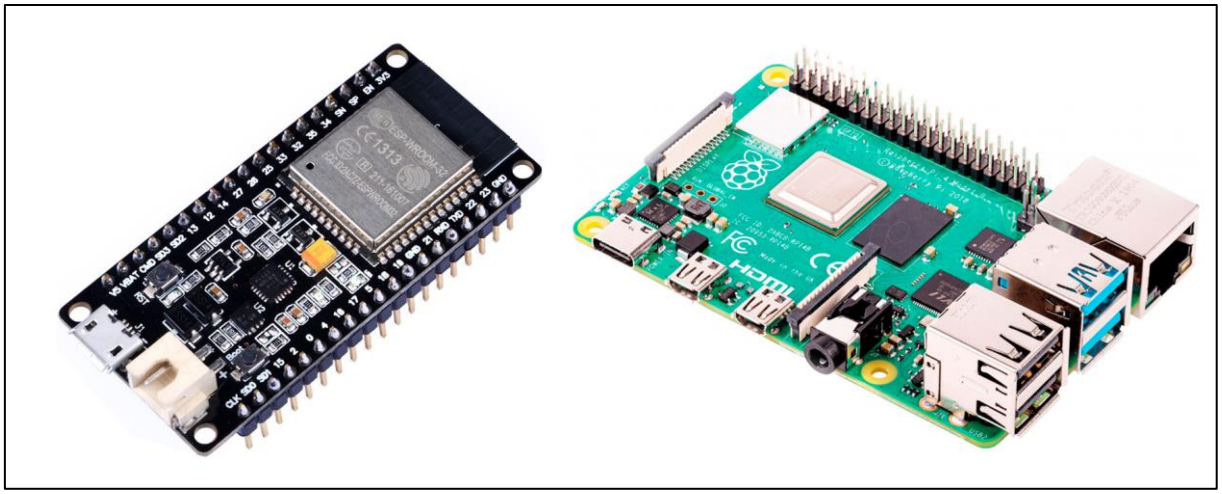


Figura 3. ESP32 i Raspberry Pi 4.

A la Figura 3 es mostra a l'esquerra la versió més coneguda de l'ESP32 i a la dreta la última versió del Raspberry Pi. A simple vista es pot veure que tots dos som força diferents, però en el seu interior encara ho són més. L'ESP32 és el xip platejat que es veu a la part superior de la placa esquerra. Aquest xip disposa d'un microcontrolador de 32 bits, amb tecnologia WiFi i entrades i sortides analògiques i digitals. Es programa a través del mini USB utilitzant un

ordinador extern. Per altra banda, el Raspberry Pi és un concepte totalment diferent, es tracta d'un ordinador, però, amb entrades i sortides físiques, de propòsit general. Disposa d'un microprocessador de 32 bits del tipus ARM i requereix un sistema operatiu per a funcionar. Aquest SO, que normalment és el Raspbian de la família Linux, es carrega en una targeta SD, que és compatible amb tots els models de Raspberry Pi. A més a més disposa de connectors USB, HDMI, ethernet o de càmera, per a instal·lar perifèrics fàcilment. El dispositiu no requereix cap element extern per a ser programat, únicament es necessita instal·lar un IDE i un compilador. A més, es poden executar diversos programes a l'hora.

Després d'investigar quines solucions hi havia amb cada tecnologia es van trobar dues plaques que complien tots els requisits. Per la part de l'ESP32 la Wemos Mega i per la part del Raspberry Pi la versió 0 W. A la Figura 4 es mostren els dos aparells que posteriorment es compararan.

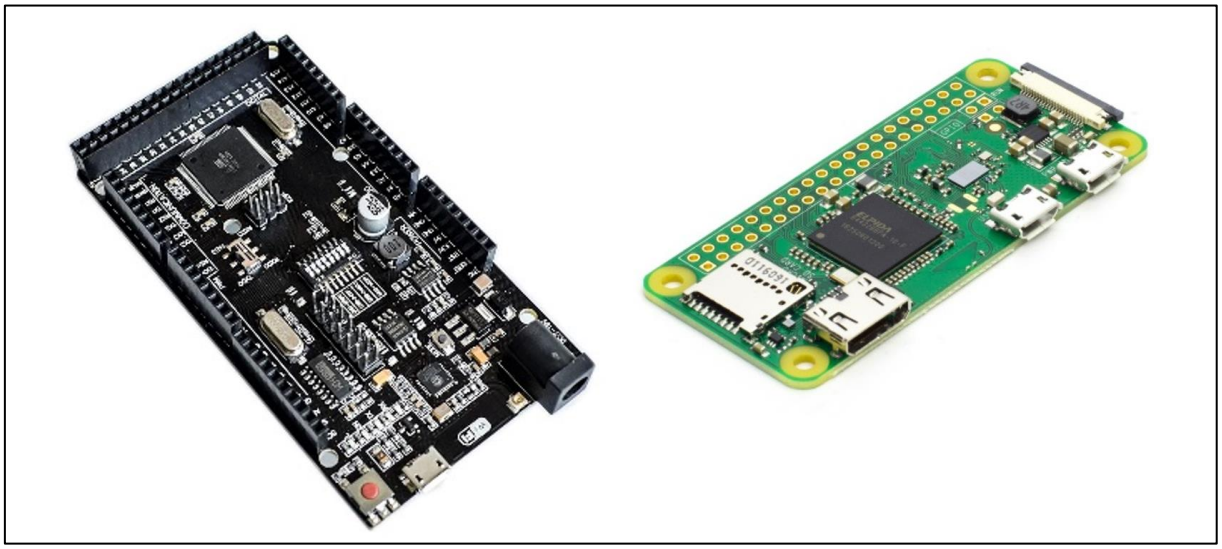


Figura 4. Wemos Mega i Raspberry Pi 0 W.

A l'esquerra es veu la placa Wemos Mega, que uneix la tecnologia de l'Arudino Mega i de l'ESP32, creant un element amb moltes entrades i sortides, WiFi, diversos ports UART, capacitat de crear interrupcions internes i comptadors i amb tecnologia de comunicació I2C i SPI. A la dreta hi ha la versió de Raspberry Pi 0 W dissenyada per a desenvolupar aplicacions de IOT. Disposa de connexions sense fils de WiFi i Bluetooth, més entrades i sortides que l'Arduino Uno, un HDMI, connector per a càmera i un USB. A la Taula 1 es mostra la comparació entre els dos elements i l'Arduino Uno que hi ha actualment a l'Ardubot.

	Arduino Uno	Wemos Mega (ESP32)	Raspberry Pi 0 W
Microcontrolador	AtMega328p	ESP32 + AtMega2560	B4432BBPA
Consum (mA)	46	178 (93 + 85)	200
Voltatge d'alimentació (V)	7 a 12	7 a 12	5
Nº de sordines i entrades digitals	14	60	26
Nº d'entrades analògiques	6	16	0
Nº de sortides PWM	6	15	2 hardware / 26 software
Intensitat de sortida (mA) Max. pin / Rec. pin / Max. total	40 / 20 / 200	20 / 10 / 200	16 / 8 / 50
Comunicacions	I2C, SPI, 1 x UART	I2C, SPI, 4 x UART, WiFi	I2C, SPI, 2 x UART, WiFi, BT,
Mida (cm)	7,6 x 6,4 x 1,9	10,16 x 5,33 x 1,9	6,5 x 3 x 1
Multitasca	NO	SI (2 CPU diferents)	SI (temps compartit)
Preu (€)	20	40	25

Taula 1. Comparació entre l'actual Arduino Uno i els possibles substituts Weamos Mega i Raspberry Pi 0 W.

A l'encapçalat de la taula es veuen els noms dels tres dispositius comparats, un a cada columna, a les files es mostren diferents característiques. Primer el microcontrolador, es veu que el Wemos Mega en té dos, que es poden programar de forma independent o conjuntament i el Raspberry Pi només un de la marca Elpida, model B4432BBPA. Seguidament, a les dades relacionades amb l'alimentació, es marca el consum i el voltatge. L'Arduino Uno, que s'utilitza actualment, té el consum més baix i les dues plaques proposades tenen un consum similar, però gairebé 3 vegades superior. Pel que fa al voltatge, la Wemos té les mateixes característiques que l'Arduino, però el Raspberry Pi 0 W s'ha d'alimentar a 5 V constants, ja que no disposa d'un regulador intern.

De les files 4 a 7 s'indiquen característiques dels pins físics com són el nombre de sortides i entrades digitals, analògiques, PWM i intensitat per a cada pin. La Wemos, en tenir la mateixa tecnologia que un Arduino Mega i a més a més l'ESP32 té moltes sortides i entrades, 60 de digitals, 15 de les quals poden tenir una sortida PWM i 16 d'analògiques (que només es poden fer servis d'entrada). El Raspberry Pi té molts menys pins, tots 26 són digitals, dels quals només 2 sortides són PWM per hardware. Malgrat això, tots el GPIO poden crear un PWM per software. Finalment a la intensitat de sortida se n'indiquen 3: la intensitat màxima de sortida per a cada pin, la recomanada i la màxima que es pot donar en total. El Raspberry és la que té més limitacions en aquest sentit.

A les quatre últimes columnes es mostren les comunicacions que té cada dispositiu, les dimensions de la placa, si permet multitasca i el preu oficial. La Wemos té les mateixes tecnologies que l'Aurduino, però amb 3 comunicacions tipus UART i a més el WiFi que li proporciona l'ESP32. El Raspberry té un UART més, WiFi i a més a més un Bluetooth integrat. Pel que fa a la mida la Wemos és la placa més gran i la Raspberry la més petita. Respecte a la multitasca, actualment l'Arduino Uno no permet executar dos programes a l'hora. La Wemos, al tenir dos microcontroladors, permet executar un programa a cada CPU, el Raspberry pot fer multitasca amb temps compartit. Per acabar, el preu que es veu a la taula és l'oficial de cada una de les cases, juntament amb els elements addicionals, com són cables de connexió, carregadors, carcassa, targetes SD, adaptadors...

Amb les característiques vistes anteriorment, les dues opcions presenten millores a l'Arduino Uno i entre elles. La Wemos Mega té menys consum, el mateix voltatge i més pins digitals i analògics. La integració seria absoluta, ja que les característiques són molt similars a l'Arduino Uno i a més a més es podrien utilitzar els mateixos programes. El Raspberry Pi té unes característiques computacionals molt més grans, incorpora Bluetooth, és més petit i més econòmic, però es requereix una placa que adapti les característiques electròniques. A més a més, s'hauria de regular el voltatge d'entrada a 5 volts, s'hauria de controlar la intensitat de les sortides i s'hauria d'afegir un mòdul d'entrades analògiques per a poder llegir el voltatge de la bateria, ja que l'RPi no en disposa cap.

Un cop sospesats tots els pros i contres dels dos dispositius s'ha decidit utilitzar el Raspberry Pi 0 W. Tot i els avantatges que aporta la Wamos Mega, les característiques computacionals són molt similars a l'Arduino i no es podrien ampliar molt les pràctiques. El Raspberry, al ser un ordinador, obre un ventall de possibilitats enorme. Amb aquest element es poden realitzar pràctiques de processament d'imatge, càlculs matemàtics complexos, PIDs de més precisió, crear servidors per emmagatzemar i visualitzar-les des de qualsevol punt del món... Les opcions són pràcticament il·limitades.

A més a més, la Raspberry té un Bluetooth incorporat, que permet una fàcil comunicació amb altres dispositius; no augmentarà molt el volum del robot, ja que és més petita; és més econòmica tot i el condicionament que s'ha de fer i és més versàtil en molts sentits. Tots aquests fets fan decantar la balança cap al model 0 W de la família Raspberry Pi. Seguidament s'indiquen més característiques tècniques d'aquest dispositiu.

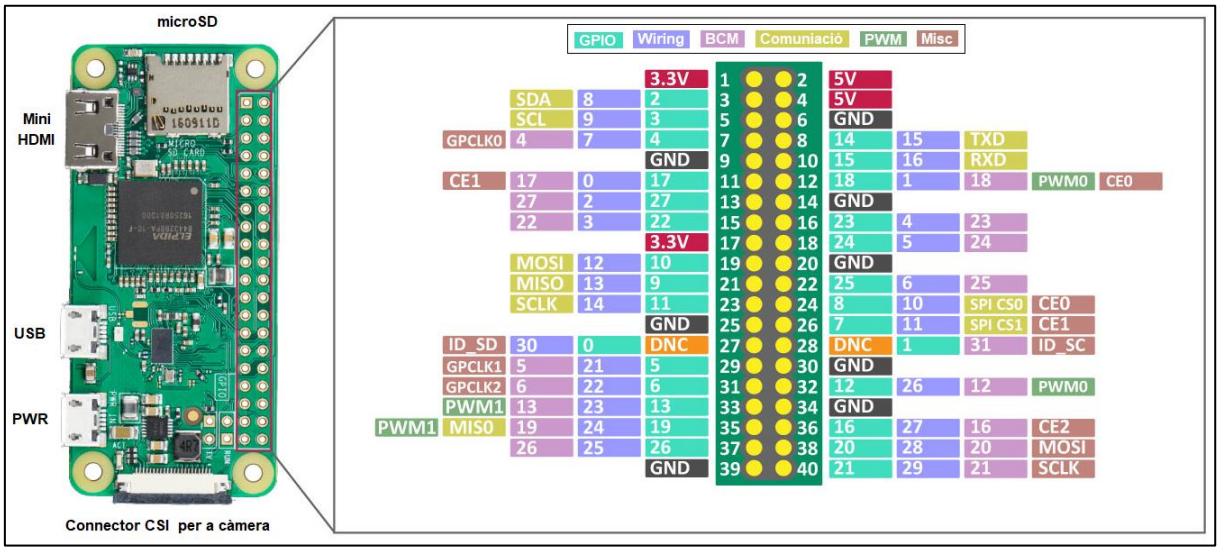


Figura 5. Raspberry Pi 0 W.

A la Figura 5 es mostren les parts del Raspberry Pi 0 W i les característiques de les seves sortides. A la part superior es troba el connector per a col·locar-hi la targeta microSD, que incorporarà el sistema operatiu i la memòria del Raspberry, funciona com un disc dur. Si seguim cap a l'esquerra, hi ha el connector mini HDMI, que permet transmetre so i imatge. A sota es troben dos connectors mini USB, el de dalt és de comunicació, per a poder connectar-hi un teclat o un ratolí i el de baix de potència, per alimentar el Raspberry Pi amb 5 V. A la part inferior es troba un connector CSI per a càmera. Al centre del dispositiu es troben els xips del microcontrolador (a prop del connector mini HDMI) i el de les comunicacions sense fils. A la part dreta es troben els 40 pins del dispositiu.

A l'ampliació dels pins es veu també quines característiques té cada un. No tots els pins són programables, ja que hi ha quatre pins d'alimentació per a sensors (en color vermell se'n marques dues de 5 V i dues de 3,3 V), set GNDs que es marquen de color negre i dos pins que es recomanen no connectar, s'indica DNC de color taronja. Aquests últims queden reservats per a la comunicació I2C de la memòria EEPROM.

Una de les peculiaritats que té el Raspberry Pi és que cada llibreria que s'utilitzi per a programar les sortides anomenades GPIO tenen una nomenclatura diferent. A la part més propera als pins, es troba el número del pin físic, que va de l'1 al 40, aquest número només s'utilitza per a saber on es troba cada pin. En verd clar, hi ha el número de GPIO, que és el que fa servir internament al kernel de l'aparell. Aquesta és la més eficient, ja que accedeix als registres interns. Amb blau i rosat hi ha els números de les llibreries WiringPi i BCM, dues de

les més utilitzades i que faciliten la programació. Als exemples i pràctiques desenvolupades en aquest treball s'utilitzarà la llibreria WiringPi, per tant, sempre que no es digui el contrari, s'estarà fent referència a aquesta numeració.

Per altra banda, a la figura també es marca quins pins tenen altres funcions. Amb verd fosc hi ha dues sortides PWM doblades, als pins físics 33 i 35 n'hi ha una i als 12 i 32 l'altre. En color groc hi ha les que fan referència a comunicació sèrie com són l'UART, l'I2C i SPI. Als pins físics 8 i 10 hi ha la transmissió i recepció de la comunicació UART. Als pins físics 3 i 5 s'hi troba els rellotges i les dades de la comunicació I2C. Finalment, als pins 19, 21, 23, 24, 26 i 35 s'hi troben els elements que s'utilitzen per a la comunicació SPI. Per acabar, amb color marró s'indiquen pins que tenen funcions de més baix nivell com els rellotges de propòsit general anomenats GPCLK i les comunicacions sèrie internes de la placa.

3.1 Placada adaptadora

Per a poder utilitzar el Raspberry Pi 0 W es requereix crear una placa adaptadora per a connectar-lo amb el robot, Per una banda tindrà les sortides amb la forma de l'Arduino Uno i per l'altra la del Raspberry Pi 0 W. També es necessitarà incorporar un sistema per a adaptar l'alimentació als 5 V que necessita el Raspberry, un mòdul d'entades analògiques i adaptar les sortides per a no superar la intensitat màxima admissible. A més a més s'han de tenir en compte les limitacions d'espai i altres especificacions dels que requereix l'Ardubot.

3.1.1 Alimentació

A diferència de l'Arduino Uno, que permet alimentar tot l'aparell a través d'un dels seus pins, el Raspberry Pi només es pot alimentar a través del connector mini-USB amb 5 V rectificats. Com s'ha explicat anteriorment, l'Ardubot s'alimenta a través de dues bateries 9 V i 800 mAh. Per a poder utilitzar aquestes mateixes bateries s'ha de regular la sortida a 5 V i s'ha d'idear una manera de portar l'alimentació al connector de potència del Raspberry. A més a més, es dissenyarà un sistema per a poder alimentar paral·lelament la placa per no gastar bateria quan s'estigui programant o en cas que no es vulgui utilitzar l'Ardubot com a un robot mòbil.

Per la part de la regulació de voltatge s'ha decidit utilitzar el regulador LM2596S-5.0, que permet una entrada entre 7 i 40 V i dona una sortida de 5 V i fins a 3 A, amb una eficiència del 80 %. Per a utilitzar aquest xip es necessiten altres elements com dos condensadors electrolítics, una bobina i un díode Schottky, com es veu a l'esquema de la Figura 6.

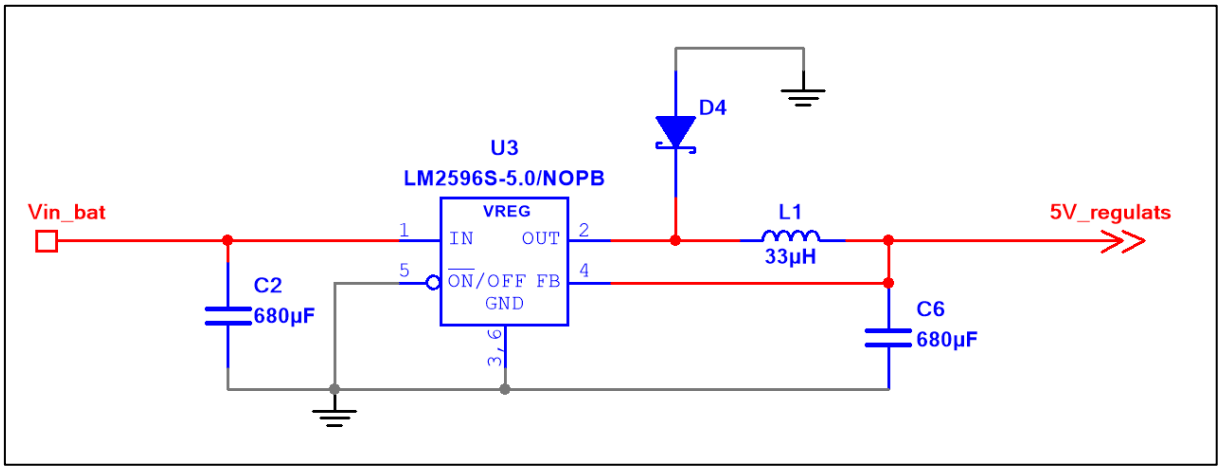


Figura 6. Esquema del regulador.

Per a poder alimenta el robot per una font externa sense haver de connectar i desconnectar les bateries s'ha utilitzat un esquema que funciona de selector automàtic. Amb l'Arduino aquest problema no existeix, ja que tan bon punt s'alimenta comença a executar el programa que se li ha programat. El Rasberry Pi és multitasca i requereix un sistema operatiu per a funcionar, per tant si es talla el subministrament quan s'està executant el programa aquest es pararà encara que es torni a alimentar. A la Figura 7 es mostra el sistema esmentat.

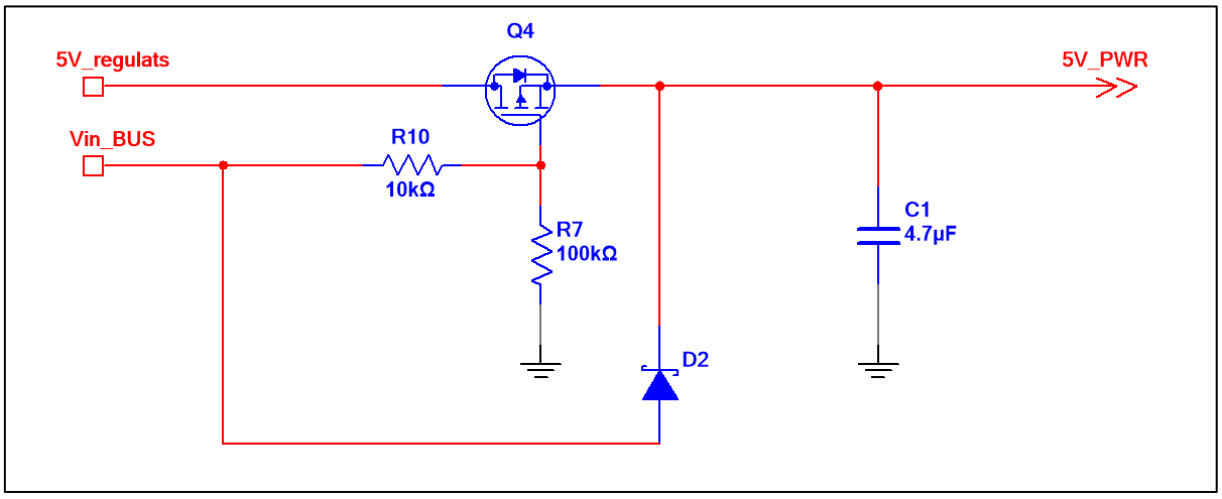


Figura 7. Esquema del selector de l'alimentació.

A la part esquerra de la figura hi ha les dues entrades d'alimentació, la de 5 V regulada, que prové de les piles, i l'anomenada Vin_BUS, que vindria a través d'un regulador extern, com podria ser un carregador de mòbil. L'element Q4 és un MOSFET de canal P que s'utilitza per a fer la selecció del voltatge. Aquest element funciona de tal manera, que si s'excita la porta impedeix el pas de corrent a través de la sortida. Quan això passa, el díode permet el pas de

corrent del voltatge de bus i impedeix corrents inverses. Quan no hi ha alimentació a través de Vin_BUS, el MOSFET permet el pas de voltatge que prové de la bateria. El condensador integra el senyal i evita talls durant la connexió i desconnexió. Per alimentar el Raspberry Pi no es poden utilitzar els pins i s'ha de fer servir el connector USB de la placa. Per portar la potència en aquest connector s'ha decidit utilitzar un cable USB-miniUSB comercial de 10 cm, que comuniqui la placa adaptadora amb el microcontrolador.

El Raspberry Pi 0 W no disposa d'un LED per indicar que s'està alimentant correctament. Per aquest motiu, se n'ha instal·lat un a la placa adaptadora. Aquest LED s'alimenta a través dels 3,3 V que proporciona el Raspberry Pi, així ens assegurem que li arriba la potència i els reguladors interns de la placa treballen correctament.

3.1.2 Mòdul d'entrades analògiques

Una de les entrades de l'Ardubot és l'estat de la bateria, per a poder llegir aquesta variable es necessita una entrada analògica i el Raspberry Pi no en té cap. Un dels mòduls que recomana la comunitat de Raspberry és el PCF8591, amb comunicació I2C. Aquest mòdul disposa de 4 entrades analògiques i una sortida analògica de 8 bits amb un consum màxim de 20 mA. Es comunica amb el Raspberry a través d'I2C i pot tenir fins a 8 adreces diferents.

L'I2C en el Raspberry és una comunicació multimaster i multislave que utilitza 2 cables. Això vol dir que hi pot haver més d'un element que doni ordres i més d'un que les rebi. Per aquest motiu és molt important saber l'adreça de l'element. El PCF8591 té tres entrades que s'utilitzen per a definir-la. Els dos cables de la comunicació són l'SCL que és el rellotge (sincronitza les dades) i l'SDA que és el senyal de les dades. Aquests dos senyals són binàries, per assegurar uns valors lògics correctes, es recomana una resistència pull up que ja existeix a la placa de l'Ardubot. A la Figura 8 es mostra l'esquema de connexió d'aquest element.

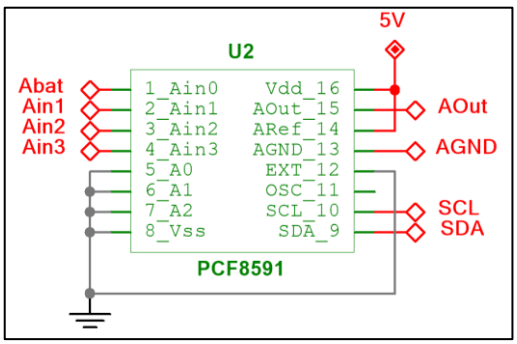


Figura 8. Esquema del mòdul analògic.

El mòdul s'alimenta amb 5 V, que també és el voltatge de referència per a fer totes les conversions. L'entrada analògica 0 es connecta a l'entrada de la lectura de la bateria. Aquest senyal ja està condicionada a l'Ardubot perquè no superi els 5 V. Les altres entrades analògiques queden lliures i es connectaran a un connector per a poder-li acoblar senyals externs, igual que la sortida analògica i el GND analògic.

Les entrades que s'anomenen A0, A1 i A2 es poden connectar a GND o al voltatge d'alimentació i defineixen l'adreça I2C del xip. Es connectaran a terra per a obtenir l'adreça amb hexadecimal 0x48. A la Taula 2 es mostra com s'obté aquest valor. L'últim bit de l'adreça, es modifica si es vol llegir o escriure en el dispositiu, per tant a l'hora de donar l'adreça es menysprea.

	Obtenció adreça del dispositiu								**Adreça Hexa.**
Bit	**6**	**5**	**4**	**3**	**2**	**1**	**0**	R/$\overline{W}$	
Adreça	1	0	0	1	A2	A1	A0		
Ex (tot GND)	1	0	0	1	0	0	0	X	0x48

Taula 2. Obtenció de l'adreça I2C del mòdul analògic PCF8591.

3.1.3 Distribució de pins

El Raspberry Pi 0 W disposa de 26 pins d'entrada/sortida (anomenats GPIO), són sis més que l'Arduino Uno, per tant en sobraran. Tanmateix, com s'ha vist anteriorment a la Figura 5, la majoria d'aquests pins tenen altres possibles funcions. Tot això i les necessitats específiques de cada element de l'Ardubot s'ha de tenir en compte per a distribuir les funcions assignades a cada pin del Raspberry Pi 0 W.

Primer de tot, s'han assignat les sortides PWM a cada un dels motors, ja que només poden funcionar amb aquest tipus de sortida si se'n vol regular la velocitat. L'Ardubot està dissenyat de tal manera que se'n necessiten dos per a cada motor (un per girar a dretes i l'altre per girar a esquerre). Tot i que hi ha quatre pins de sortida, són doblades i només es poden programar dos PWM per hardware. Existeixen diverses llibreries que creen un PWM per software que es pot executar a qualsevol pin GIPO, per tant s'eliminen les limitacions. Tot i això, per a poder fer una pràctica aprofitant el PWM per hardware s'ha decidit que cada sortida de cada motor s'alimentaria amb un de diferent, tal com es mostra a la Figura 9. Com que la placa de l'Ardubot disposa d'un driver per a controlar els motors, la intensitat consumida per aquestes sortides ja és molt petita, per tant no s'ha d'adaptar el senyal.

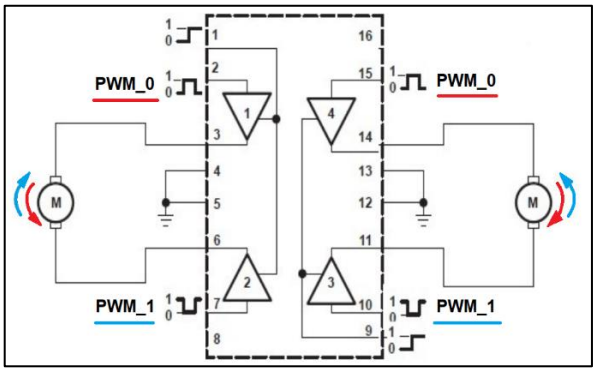


Figura 9. Driver del motor i alimentació dels PWM per hardware.

La pantalla LCD que utilitza el robot funciona amb tecnologia I2C així que és important que els pins físics 3 i 5, que porten aquesta tecnologia es connectin als connectors que porten la comunicació a la pantalla. Un altre element de l'Ardubot que requereix una consideració especial són els sensors que s'utilitzen com a encoders, ja que necessiten una entrada que permeti crear interrupcions asíncrones. Un dels avantatges del Raspberry Pi és que qualsevol de les seves entrades es pot programar com a tal, per tant en aquest cas no es requereix cap pin en concret.

Un cop definits tots els elements que tenen necessitats especials, la resta de sensors i actuadors són digitals (hi ha LEDs, polsadors i seguidors de línia). Per a connectar aquests elements amb el Raspberry Pi s'intentaran fer servir els pins que només disposin de GPIO, per poder aprofitar al màxim totes les possibilitats que ofereix la placa.

S'ha començat a distribuir els pins per les sortides, és a dir, els 3 LEDs. A la placa de l'Ardubot, dos aquests elements s'alimentaven a través dels pins de comunicació de l'Arduino Uno, per tant comparteixen pin amb un dels connectors auxiliars de la placa que s'utilitza per a connectar el Bluetooth. A més a més, el tercer LED té un pont que permet a l'usuari decidir si la sortida de l'Arduino alimenta el LED o un connector que permet connectar-hi un element extra. Aquestes peculiaritats es van fer per a poder aprofitar al màxim les reduïdes sortides de l'Arduino Uno. Com que el Raspberry té més sortides, quan s'utilitzi la placa adaptadora que s'està dissenyant en aquest projecte, els connectors de l'Ardubot quedaran anul·lats. Així doncs, els LEDs s'han de tractar com unes sortides sense cap requisit especial. Només s'ha tingut en compte que la intensitat màxima de totes les sortides del Raspberry pot no ser suficient i per alimenta els LEDs, tant s'han col·locat uns transistors per a poder-los alimentar correctament. A la Figura 10 es mostra l'esquema que s'ha fet servir per a cada una de les sortides que alimenten un LED, en total hi ha 3 conjunts de transistors.

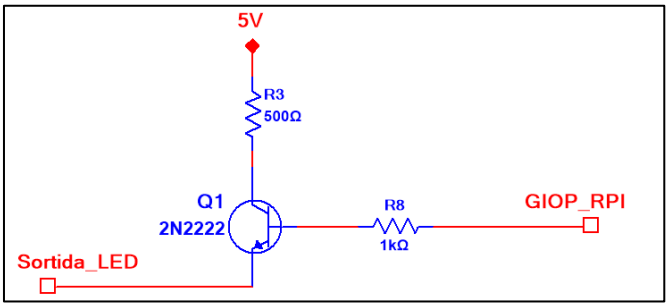


Figura 10. Esquema del transistors que alimenta un LED.

Les entrades del robot que resten són els dos encoders, els quatre seguidors de línia, el polsador i els dos para-xocs. Tot l'Ardubot està alimentat a 5 V per tant l'1 lògic és de 5 V. Tot i que el Raspberry s'alimenti a 5 V té un voltatge lògic de 3,3 V i per tant s'ha d'adaptar la senyal. Per fer-ho s'han fet un divisor de tensions a cada una de les entrades. Per a calcular el valor de les resistències s'ha utilitzat l'Equació 1 i s'ha trobat que R2 ha de ser de 3,3 kΩ i R1 de 1,7 kΩ.

$$V_{OUT} = \frac{R_2}{R_2+R_1} \cdot V_{IN} \qquad \text{(Eq. 1)}$$

On: V_{OUT} és el voltatge de sortida que ha de ser 3,3V.
V_{IN} és el voltatge d'entrada que és de 5 V.
R_1 és la resistència que es troba entre Vin i Vout.
R_2 és la resistència que es entre Vout i GND.

Les entrades condicionades es connecten als GPIO que no tinguin altres funcions i intentant no fer encreuaments, per a poder tenir a distribució de pistes més fàcil. Un cop fet això queden lliures 8 pins GPIO del Raspberry i 4 del mòdul analògic. Aquests, juntament amb els de la comunicació I2C, s'han distribuït en diferents connectors per facilitar el seu ús, tal com es mostra a la Figura 11.

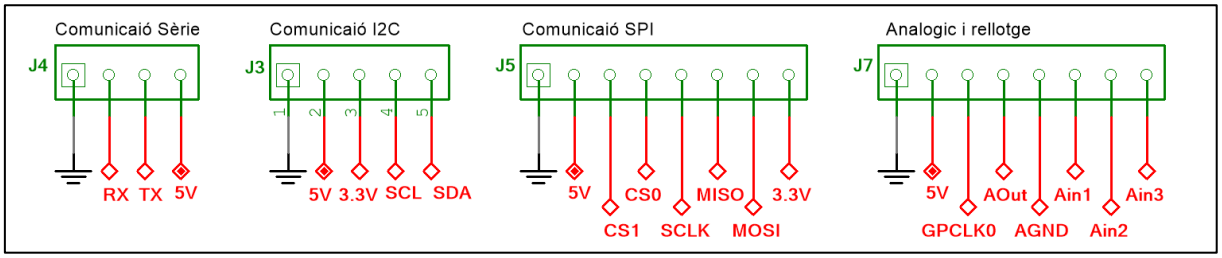


Figura 11. Connectors de pins lliures.

Es pot observar que s'han distribuït els pins per tipus de comunicació i ús. A tots els connectors s'hi troba 5 V i GND. Començant per l'esquerra hi ha els de comunicació sèrie, distribuïda de tal manera que s'hi poden connectar diferents tips de sensors com GPS que utilitzen aquesta tecnologia o ultrasons que té de dues sortides digitals. Seguidament es troba el connector de l'I2C, que només es pot fer servir per a aquesta comunicació, ja que s'utilitza a la resta del robot. En tercera posició hi ha el connector SPI, disposa dels CS (Chip Selector per utilitzar el Raspberry com a esclau), el rellotge SCLK i la comunicació MISO i MOSI. Aquestes dos connectors tenen una sortida de 3,3 V. Finalment es troba el connector del mòdul analògic, amb les entrades, la sortida i el terra analògic. També hi ha un GPIO del Raspberry que pot ser utilitzat com a rellotge. S'ha de recordar que totes aquestes pins (menys els de l'I2C) es poden utilitzar com a entrades o sortides digitals. A la Taula 3 es mostra la distribució final de tots els pins del Raspberry Pi 0 W, juntament amb la seva funció, on estan connectats i el número que tenen a la llibreria de WiringPi.

Element	**Funció**	**Numero de pin físic**	**Numero al WiringPi**
Elements de l'Ardubot			
LED 1 (Verd)	Sortida digital	11	0
LED 2 (Taronja)	Sortida digital	13	2
LED 3 (Vermell)	Sortida digital	15	3
Encoder Esquerra	Entrada digital	29	21
Encoder Dret	Entrada digital	31	22
Motor Dret	Sortida digital (PWM1)	33	23
Motor Dret	Sortida digital (PEM0)	32	26
Motor Esquerra	Sortida digital (PWM1)	35	24
Motor Esquerra	Sortida digital (PEM0)	12	1
Seguidor Lateral Esquerra	Entrada digital	37	25
Seguidor Centre Esquerra	Entrada digital	40	29
Seguidor Centre Dret	Entrada digital	38	28
Seguidor Lateral Dret	Entrada digital	36	27
Polsador	Entrada digital	22	6
Para-xocs Esquerra	Entrada digital	18	5
Para-xocs Dret	Entrada digital	16	4
Lectura Bateria	Mòdul analògic (Entrada)	- (comunicació I2C)	A0
Connectors a la placa adaptadora ArduPi			
Connector Sèrie	RX	10	16
Connector Seriè	TX	8	15
Connector I2C	SDA	3	8
Connector I2C	SCL	5	9
Connector SPI	CS1	26	11
Connector SPI	CS0	24	10
Connector SPI	SCLK	23	14
Connector SPI	MISO	21	13
Connector SPI	MOSI	19	12
Connecotr Analògic i Clock	GPCLK0	7	7
Connecotr Analògic i Clock	Mòdul analògic (Sortida)	- (comunicació I2C)	A0
Connecotr Analògic i Clock	Mòdul analògic (Entrada)	- (comunicació I2C)	A1
Connecotr Analògic i Clock	Mòdul analògic (Entrada)	- (comunicació I2C)	A2
Connecotr Analògic i Clock	Mòdul analògic (Entrada)	- (comunicació I2C)	A3

Taula 3. Distribució dels pins del Raspberry

3.1.4 Circuit imprès

A l'hora de dissenyar el circuit imprès primer s'ha definit la mida màxima que pot tenir la placa. La placa adaptadora ha de substituir l'Arduino Uno que està molt encaixat entre els altres elements del robot. A la Figura 12 es veu la distribució dels diferents elements del robot educatiu.

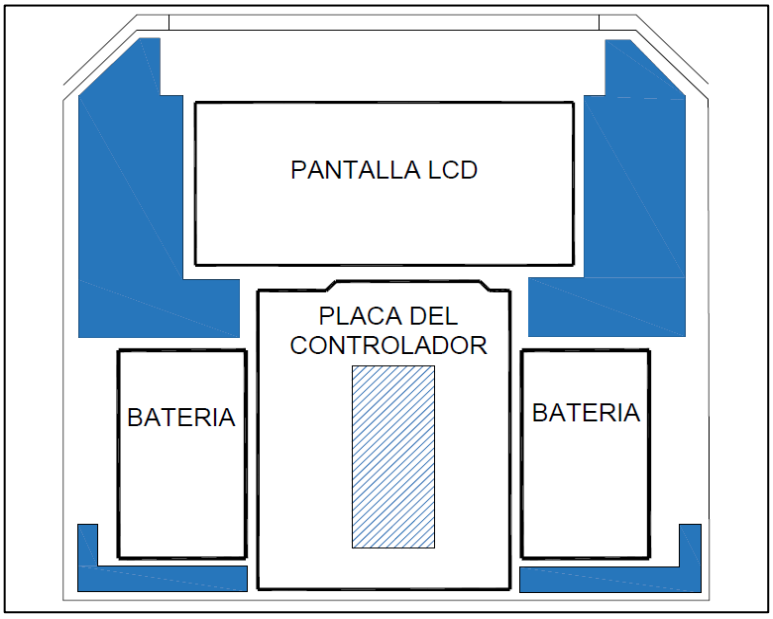


Figura 12. Distribució de l'Ardubot.

S'han marcat alguns elements principals que tenen una alçada igual o superior a la placa de l'Arduino com són les bateries i la pantalla LCD. Al centre de la placa s'hi troba la placa del controlador, que anteriorment era un Arduino Uno. Amb color blau s'han marcat les àries que estan ocupades per altres elements. A la part central es veu una zona ratllada, aquí s'hi troben els drivers del motor i del seguidor de línia, que també tenen una alça considerable. Tot i cabre perfectament sota de l'Arduino, no es podrà ficar cap component a la cara inferior de la placa en aquesta zona.

Així doncs, la placa no pot créixer pels costats i per la part de davant només 3 mil·límetres. Per la part de darrere, però, es pot allargar el que sigui necessari, tenint en compte que sobresortirà de la placa principal de l'Ardubot. Finalment s'ha decidit fer una placa rectangular, ja que és la forma més bàsica i econòmica de produir. Tindrà unes dimensions de 81x53 mm amb un disseny compacte i una alçada de 16 mm que sobresortirà uns centímetres per la part posterior de l'Ardubot.

La part posterior de la placa adaptadora també és l'única que pot tenir connectors USB, ja que a la resta de costats hi ha elements que impediries la seva connexió. En aquesta part es distribuiran els dos connectors USB i el regulador de voltatge. El Raspberry Pi es trobarà al centre de la placa, suficientment elevat perquè hi càpiguen elements a sota. A la Figura 13 es mostra la distribució de la placa.

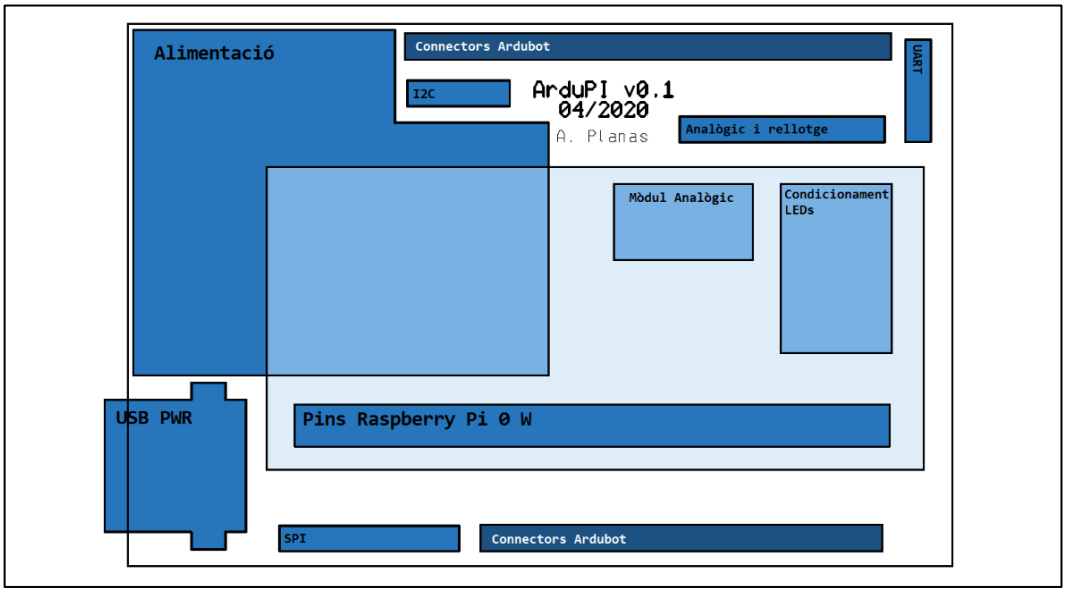


Figura 13. Distribució de la placa adaptadora.

Cada una de les zones marcades amb blau tenen el nom al què estan destinades. A la part esquerra s'observa la zona que cobreix el condicionament de l'alimentació, ocupa quasi un 50 % de la superfície. Amb aquest mateix to es marca on es troba cada un dels connectors de sortides extra i del Raspberry. Amb el blau més clar s'indica la superfície que cobreix el Raspberry Pi. A sota, amb un altre to de blau, s'hi troben el mòdul analògic i el condicionament dels LEDs que no seran visibles perquè quedaran per sota del Raspberry. Amb color blau marí es mostren els connectors que es troben a la part inferior de la placa i que la connecten a l'Ardubot. A la cara inferior, a la zona de l'alimentació, hi ha el mini USB per a l'alimentació externa i el MOSFET de la selecció automàtica d'alimentació.

A l'hora de dissenyar el circuit s'han tingut en compte peculiaritats com la dissipació del regulador de voltatge. Per tenir una millor dissipació s'han fet el que es coneixen com a vies tèrmiques sobre un pla màssic. El regulador és SMD i per tant no pot portar un dissipador extern. Per contra té un gran pad de terra que gràcies a les vies dissipa l'escalfor del regulador cap a la capa inferior de la placa, tal com es mostra a la Figura 14.

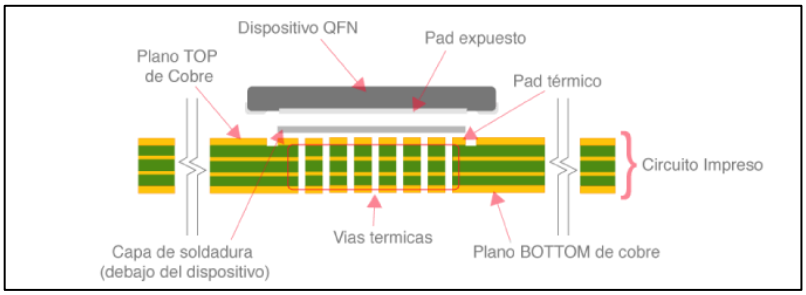


Figura 14. Vies de dissipació

També s'ha distribuït un gran pla màssic sota el mòdul analògic, per evitar sorolls i millorar el senyal. Aquest element està molt proper al seu connector, ja que si les pistes són més curtes, hi ha menys soroll. Finalment s'han fet els forats per a subjectar correctament el Raspberry Pi a la placa adaptadora. Un cop dissenyada la placa adaptadora s'ha fet una simulació per a mostrat com quedarien integrats tots els elements, es mostren tres vistes de la simulació a la Figura 15.

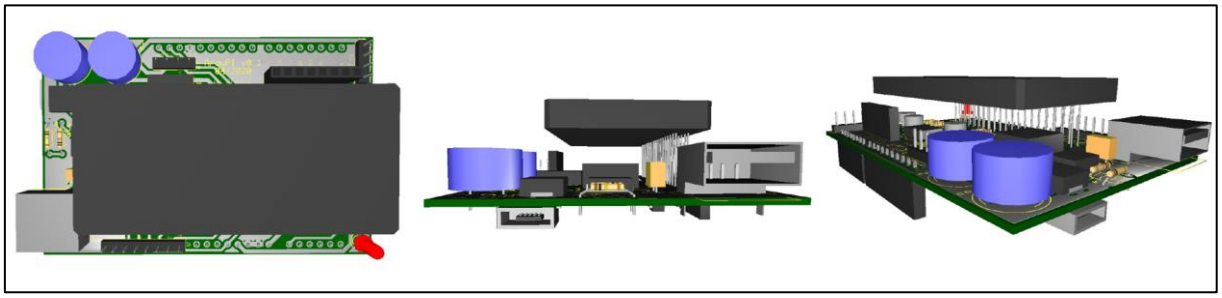


Figura 15. Simulació 3D de la placa adaptadora.

El Raspberry Pi és l'element negre que està més elevat. A la vista central es veuen els connectors USB, un per sota i l'altre per sobre la placa. La última vista, amb més angle permet apreciar les diferents altures, els connectors inferiors i la majora dels elements de la placa.

Finalment, per a facilitar la feina a l'usuari final, a la serigrafia de la placa s'han escrit totes les característiques de cada sortida en els diferents connectors. Com que la placa adaptadora també pot ser utilitzada fora del robot, també s'han escrit els números i característiques a la part inferior de la placa, on s'indica que tots els números fan referència als que utilitza la llibreria WiringPi.

4 ESTUDI D'AUTONOMIA I MILLORES MECÀNIQUES

Per a plantejar les millores mecàniques i d'autonomia, primer es realitza un estudi de l'autonomia de l'Ardubot amb la placa adaptadora. A més a més, es resoldrà el problema de desgast que es té actualment amb l'adhesiu de l'encoder.

Actualment, el robot s'alimenta a través de dues bateries de 9 V i 800 mAh cada una, connectades en paral·lel, per tant, la capacitat total de les bateries és de 1600 mAh. Per a saber el consum que tindrà el robot s'ha realitzat un programa que activa totes les teves sortides i entrades. A la Taula 4 es mostren els consums en diversos casos.

Elements	Consums
Raspberry (amb el WiFi treballant)	200 mA
LEDs + Pantalla Ardubot	140 mA
LEDs + Pantalla Ardubot + Motors	254 mA
LEDs + Pantalla Ardubot + Motors amb carrera	314 mA
LEDs + Pantalla Ardubot + Motors bloquejats	394 mA

Taula 4. Consums de l'Ardubot.

El consum del Raspberry Pi 0 W, utilitzant l'escriptori remot i el Code::Blocks (que s'utilitzarà per a programar el robot), és de 200 mA. El consum de diferents elements de l'Ardubot es mostra a l'apart inferior, i fa referència al consum sense microcontrolador. Les proves s'han fet amb l'Arduino Uno que consumeix 46 mA, per tant s'ha restat aquest valor del consum total del robot per obtenir-los. El consum de tots els LEDs i la pantalla LCD és de 140 mA, si se li afegeixen els dos motors al 60 % de la velocitat màxima, augmenta a 254 mA. Quan aquests motors tenen carrega (el robot es mou normalment) el consum és de 314 mA, però si aquests motors queden bloquejats, arriba fins a 394 mA.

Es considera que el 100 % de la sessió de pràctiques s'està utilitzant el Raspberry, el 40 % només funcionen els LEDs i el 60 % restant també funcionarà el motor. D'aquest 60 %, un 70 % els motors tindran una càrrega normal, un 20 % no tindran carrega i un 10 % els motors quedaran bloquejats. Utilitzant aquests percentatges i els consums anteriors s'aplica l'Equació 2 i s'obté que el consum mitjà a les pràctiques és de 442 mA.

$$200 + 140 \cdot 0{,}4 + (314 \cdot 0{,}7 + 254 \cdot 0{,}2 + 394 \cdot 0{,}1) \cdot 0{,}6 = 442 \text{ mA} \qquad \text{(Eq. 2)}$$

Amb aquest consum i la capacitat de les bateries ja es pot saber quantes hores pot estar funcionant l'Ardubot durant una pràctica normal. Per fer-ho només s'ha de dividir la capacitat

amb mAh pel consum amb mA i s'obtindrà l'autonomia del robot amb hores, tal com es mostra a l'Equació 3. S'obté que l'autonomia és de 3,62 hores.

$$\text{Autonomia (h)} = \frac{\text{Capacitat (mAh)}}{\text{Consum (mA)}} \qquad \text{(Eq. 3)}$$

Una sessió de pràctiques de l'assignatura d'Aplicacions Industrials dels Microprocessadors és de 3 h, per tant, encara que s'augmenti el consum, es podrà fer tota la pràctica sense haver de canviar o recarregar les bateries. A més a més, la placa adaptadora, disposa d'un connector mini USB, per alimentar el robot en cas que no es facin servir els motors. Així doncs, no és necessari realitzar una millora d'autonomia del robot.

Pel que fa a la millora mecànica, es planteja canviar la roda i l'encoder. Actualment, les rodes de l'Ardubot es tallen d'un tub massís de PBC, es marquen, es foraden, se'ls hi adhereix la goma i finalment se'ls hi enganxa un paper amb el dibuix de l'encoder. Aquestes rodes s'uneixen al motor per pressió. Un dels problemes que apareix molt sovint a l'hora de fer les pràctiques és que els papers de l'encoder es desenganxen o es gasten i perden la seva funcionalitat. A la Figura 16 es veuen dues rodes diferents desgastades, on queda la part superior gris i per tant el sensor de contrast no es distingeixen les franges.

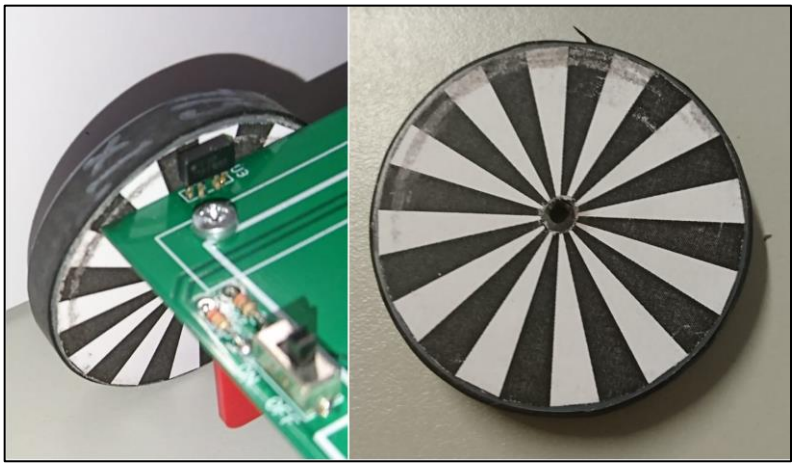


Figura 16. Roda actual de l'Ardubot desgastada.

Per evitar això s'ha disseny una roda amb l'encoder incorporat per a poder-la imprimir amb tecnologia 3D. S'ha fet amb el software OpenSCAD, un programa gratuït de disseny 3D de codi estructurat. Aquest programa permet crear els dissenys parametritzats, de forma que canviant algunes variables es poden modificar parts de l'element. En el cas de la roda, s'ha

definit el número de forats que es volen a l'encoder, l'ampla de la roda i el gruix del pneumàtic. A la Figura 17 es mostra el disseny final renderitzat.

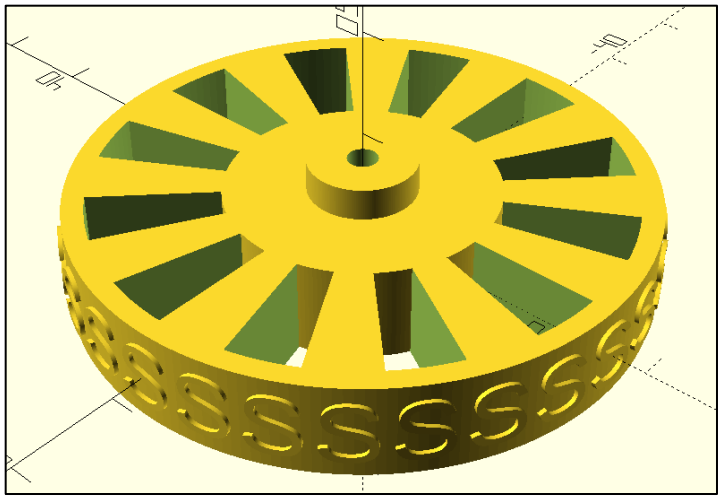


Figura 17. Disseny 3D de la nova roda.

Es pot observar que a la part central de la roda es troba un reforç que sobresurt 3 mm per a millorar la unió entre l'eix i aquests element. Seguidament hi ha 12 forats radials, de la meixa mida que la part opaca. A la part més exterior s'hi troba el pneumàtic, amb un gruix de 4 mm i un dibuix amb "S" per a millorar la tracció. Aquesta roda està pensada per ser impresa amb TPU, un plàstic tou altament resistent. D'aquesta manera l'eix entra a pressió al forat central i s'espera una gran durabilitat. A la Figura 18 es veu el resultat un cop impresa i també muntada a l'Ardubot. S'ha comprovat que l'encoder funciona correctament i el gruix de la roda és suficient perquè, si es tapen els forats per la part exterior, continuï detectant bé les franges. A la última imatge s'aprecia el sensor diferencial entre el forat superior de l'encoder.

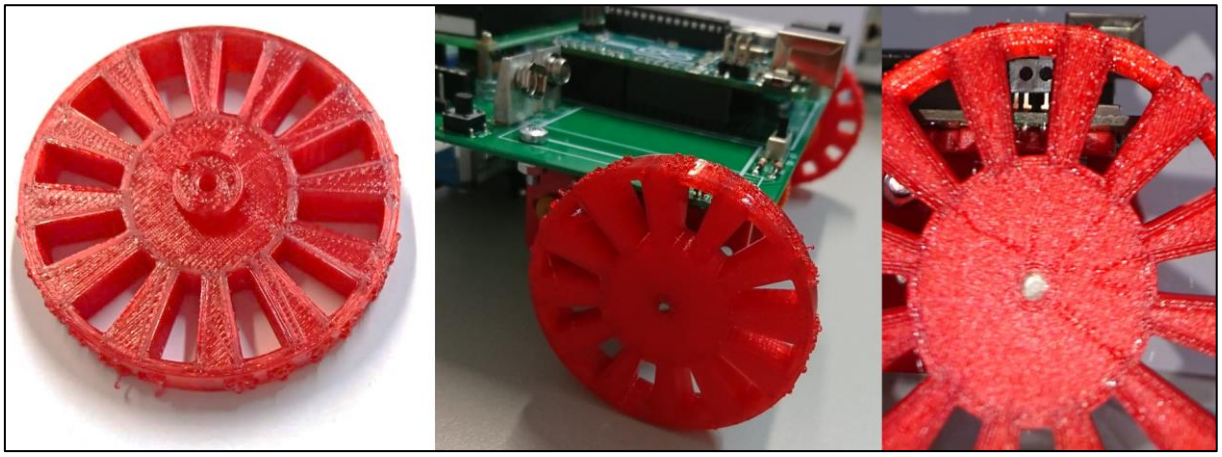


Figura 18. Roda impresa amb TPU i instal·lada al robot.

Amb la configuració i la impressora que s'ha utilitzat per a fer els prototips s'han gastat 15 g de plàstic TPU. Actualment, un quilogram d'aquest plàstic flexible costa uns 30 € per tant, el material per a cada roda, costa uns 45 cèntims d'euro.

A la part exterior de la roda actual, una CNC marca el logotip de la UdG, amb la roda impresa també s'ha volgut fer però hi ha hagut alguns problemes. Després de vectoritzar el logotip, importar-lo al programa i crear la marca al centre de la roda, es va renderitzar la imatge i el logotip no apareixia. Es va investigar i aquest programa no permet unir logotips amb altres elements, de forma que no es va poder fer la marca. De totes maneres, s'ha creat un mòdul que et permet escriure a la part central de la roda, amb una profunditat de dos mil·límetres, suficient per distingir les paraules sense malmetre la robustesa estructural de la roda. A la Figura 19 es mostra un exemple del que es podria fer.

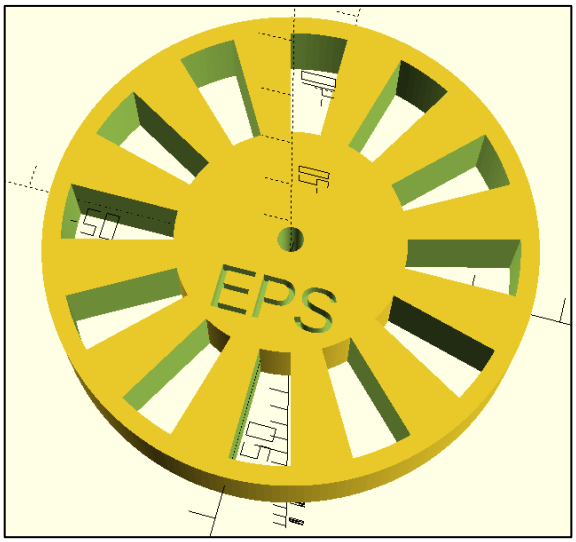


Figura 19. Disseny de la roda amb un escrit al centre.

5 FUNCIONAMENT I CONFIGURACIÓ

El Raspberry Pi 0 W és un petit ordinador de placa única de la família dels Raspberry Pi. És un hardware de disseny lliure i baix cost desenvolupat per estimular l'aprenentatge de les ciències de la computació. Per a funcionar correctament requereix un sistema operatiu, tot i que n'existeixen diversos de compatibles es recomana utilitzar el Raspbian. Aquest és un sistema operatiu amb un nucli amb distribució de Linux basat en l'arquitectura Debian i optimitzat per al hardware del Raspberry. A més a més està compost íntegrament per programari lliure i això el fa idoni per a utilitzar-lo en les pràctiques de la universitat.

El SO es pot descarregar gratuïtament a través de la pàgina oficial de Raspberry. Existeixen tres versions: la més completa que té escriptori i softwares recomanats com per exemple l'Scratch i el Libre Office; la mitjana que només té escriptori i la més lleugera que no disposa d'escriptori, és a dir, no té cap interfície gràfica i s'ha de fer tot mitjançant terminal. El model de Raspberry Pi 0 W no és prou potent per acollir tots els programes i s'escull la versió mitjana. Un escriptori fa més amable la programació i després s'instal·laran els programes desitjats.

Un cop descarregat el fitxer a l'ordinador s'ha de copiar la imatge del sistema operatiu a la targeta SD. Aquesta targeta es pot introduir a qualsevol dispositiu de la família Raspberry i funcionarà correctament. Per a fer la còpia i convertir la SD en un disc dur, s'ha utilitzat el programa Win32 Disk Imager. Primer s'ha de descomprimir el fitxer per obtenir un ".img". Seguidament s'ha d'introduir la targeta a l'ordinador (normalment es requereix un adaptador). A l'obrir el programa Win32 Disk Imager s'ha de seleccionar la imatge del sistema operatiu i la targeta SD on es vol introduir el SO. Finalment s'ha de prémer Write i esperar que acabi el procés, a la Figura 20 es veu la interfície del programa.

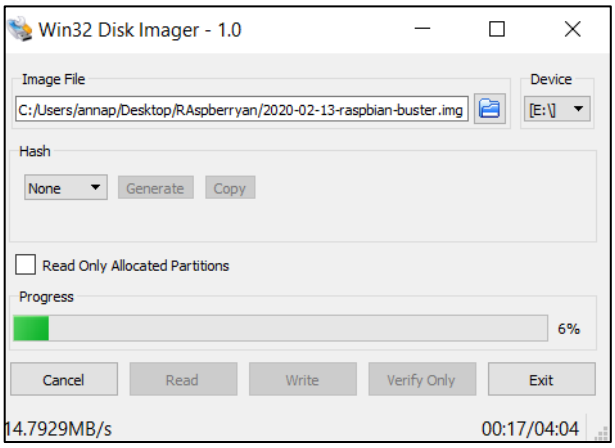


Figura 20. Interfície del programa Win32 Disk Imager mentre es crea una imatge a la targeta SD

Un cop es té la targeta a punt s'introdueix a la Raspberry i s'inicia el sistema. Per a fer-ho es necessita connectar el dispositiu a una pantalla amb connexió HDMI, a un teclat i a un ratolí a través d'un USB. Tan bon punt s'obre el sistema operatiu apareix una finestra per triar el país, la llengua, la zona horària, la contrasenya del sistema (s'ha triat AppluP), l'aparença de l'escriptori i si es vol connectar el dispositiu al WiFi. Després es preparà el sistema i es reinicia el dispositiu diverses vegades, aquest procés pot trigar uns minuts.

Seguidament s'obre el terminal i com a súper usuari s'actualitza el sistema operatiu i les llibreries. També s'instal·la l'entorn de programació que s'utilitzarà per a programar, juntament amb les llibreries, el programa per a poder obrir un escriptori remot, utilitzar la càmera i la tecnologia MQTT. Es configura la IP fixa i s'activen les interfícies dels diferents tipus de connexions, com la càmera, l'SSH, l'SPI o l'I2C.

A la Taula 5 es mostren totes les accions i comandes que es necessiten per a tenir l'entorn del a punt per les pràctiques, a la Figura 21 hi ha la interfície que apareix a l'obrir les configuracions del Raspberry per activar les interfícies.

Acció	Comandes al terminal
Actualització del sistema	sudo apt-get update sudo apt-get upgrade
Instal·lació de llibreria i programes	sudo apt-get install wiringpi sudo apt-get install xrdp sudo apt-get install codeblocks sudo apt-get install luvcview sudo apt-get install motion
Instal·lació de llibreria i l'entorn per a l'MQTT	sudo apt-get install build-essential gcc make cmake cmake-gui cmake-curses-gui sudo apt-get install fakeroot fakeroot devscripts dh-make lsb-release sudo apt-get install libssl-dev duso apt-get install doxygen graphviz Descarregar el zip de https://github.com/eclipse/paho.mqtt.c i descomprimir-lo make sudo make install
Configuració de la IP fixa	sudo nano /etc/dhcpcd.conf Afegir al final interface wlan0 static ip_address=192.168.1.10/24 static routers=192.168.1.XX
Activar les interfícies (càmera, VCN, SSH, SPI, I2C, serial, 1-wire i remote GPIO)	sudo raspi-config Obrir: "5 Inferacting Options" Activar elements

Taula 5. Accions i comandes al terminal per a iniciar el Raspberry.

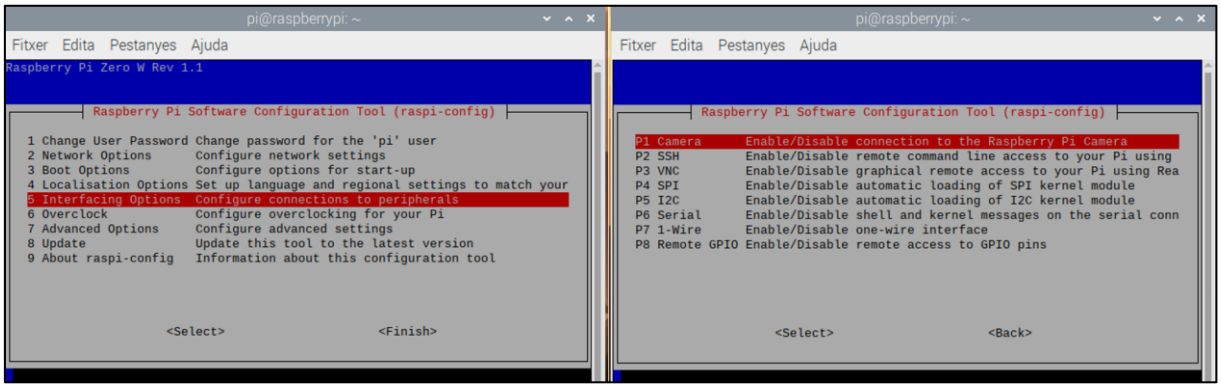


Figura 21. Interfície de la configuració del Raspberry Pi per activar les interfícies.

Per a fer les pràctiques es podria utilitzar el mètode que s'ha fet servir per configurar el sistema, és a dir, connectar una pantalla, un teclat i un ratolí al Raspberry, però no es farà així. Connectar i desconnectar tants elements cada cop que es volgués fer una modificació al programa seria molt ineficient i poc pràctic. S'ha optat per a utilitzar un escriptori remot, que permet tenir tota la interfície del sistema operatiu sense cap connexió física. Per fer-ho es permeten les connexions remotes i s'ha instal·lat el programa xrdp per a facilitar la connexió. Per connectar-se la Raspberry des de qualsevol ordinador que es trobi amb xarxa local només és necessari obrir l'aplicació de Windows "Connexió a escriptori Remot", indicar la IP del dispositiu i iniciar secció amb l'usuari i la contrasenya correcte.

Al laboratori de sistemes intel·ligents, on es fan les pràctiques amb l'Ardubot, disposa d'un punt d'accés WiFi que crea una xarxa local d'internet. Per a facilitar la feina de saber quina adreça té cada robot es marca una IP fixa que anirà des de la 192.168.1.11 a la 192.168.1.18, un per a cada robot. L'usuari és "pi" i la contrasenya "AppIuP". A la Figura 22 es mostren els passos per a obrir l'escriptori remot des d'un ordinador amb sistema operatiu Windows.

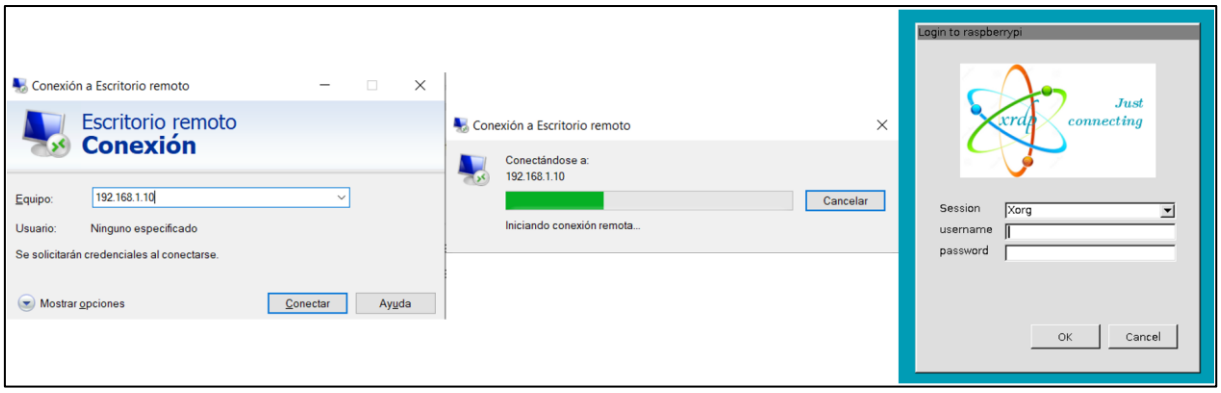


Figura 22. Passos per obrir l'escriptori remot del Respberry des d'un ordinador amb sistema operatiu Windows.

Si no es vol utilitzar aquesta tecnologia, per exemple, perquè ocupa massa amplada de banda de WiFi i no és tan àgil com es volia, es poden fer servir altres programes com el FileZilla. Aquest programa utilitza tecnologia FTP, que permet enviar fitxers remotament, de forma que es pot programar el dispositiu a l'ordinador i enviar-lo al Raspberry desitjat. Aquest programa, juntament amb el PuTTY, que utilitza la tecnologia SSH, permet creat un terminal remot i dóna un control absolut del Raspberry utilitzant pocs recursos i amplada de banda del WiFi.

Finalment, per a incloure la llibreria WiringPi a Code::Blocks per doner compilar i executar correctament s'ha de posar la direcció de la llibreria a l'IDE.. Per fer-ho només s'ha d'obrir l'IDE Code::Blocks, a la barra de menú superior, prémer Setings, Compiler, Link Settinh i prémer Add, en aquest punt s'ha d'incloure la direcció de la llibreria (/user/lib/libwiringPi.so). A la Figura 23 es mostra com queda el menú un cop s'ha inclòs la direcció.

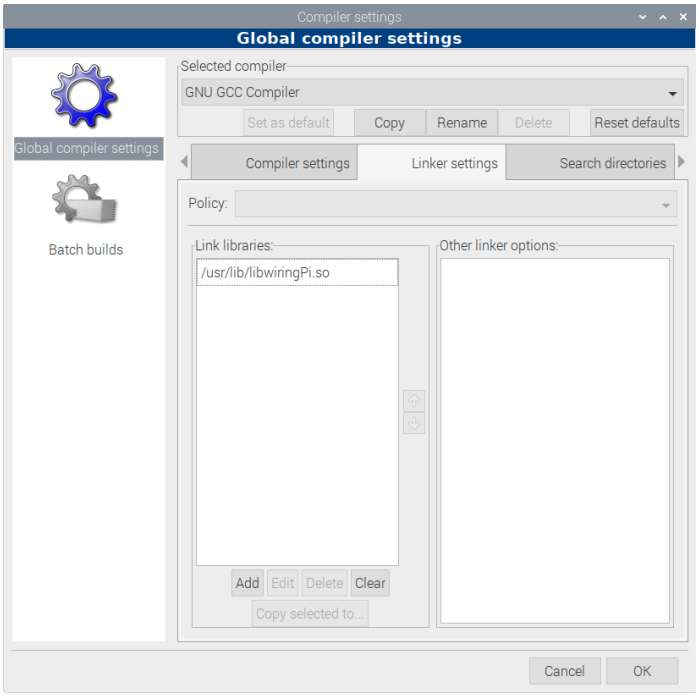


Figura 23. Configuració del compilador un cop s'ha afegit l'adreça de la llibreria WirinPi.

5.1 Útil per a la pantalla LCD

L'Ardubot 4.1 disposa d'una pantalla LCD que es troba a la part davantera del robot. Es tracta d'una pantalla de 2 files amb 16 caràcters cada una. El tipus de comunicació és I2C, per tant s'utilitza un bus de comunicació per enviar-li la informació. La pantalla és una interfície maquina-humà molt útil. Concretament és una pantalla de la marca MIDAS, del model MC21605C6W-SPTLYI-V2. S'alimenta a 5V i cada caràcter té 5x8 píxels.

S'ha creat una un útil per a utilitzar la pantalla LCD amb comunicació I2C perquè els alumnes tinguin l'oportunitat d'entendre el funcionament d'aquesta i poder modificar i ampliar fàcilment les funcions de la pantalla. Aquest útil disposa de 6 funcions bàsiques, iniciar la comunicació I2C de la pantalla, iniciar la pantalla, escriure, moure el cursor, netejar i tancar comunicació.

La funció LCD_InitI2C(void) inicia la comunicació I2C amb la pantalla i no té cap paràmetre d'entrada. Retorna un identificador que s'utilitzarà a totes les funcions per a indicar que es vol fer servir aquesta pantalla.

La funció sendLcdInit(int) té un paràmetre d'entrada que és d'identificador de la comunicació, ha de ser un enter i té cap retorn. A la inicialització es recorren tots els registres necessaris.

La funció sendLcdString(int,*str) té dos paràmetres d'entrada i cap retorn. El primer paràmetre ha de ser l'identificador de la comunicació que ha de tenir el format d'un nombre enter. El segon paràmetre ha de ser un punter a un string que es recorre fins que no queda cap element per enviar-lo al registre correeponent.

La funció sendLcdCursorLocat(int,int,int) té tres paràmetres d'entrada i cap de sortida. El primer paràmetre és l'identificador, el segon és la fila i el tercer la columna on es vol col·locar el cursor, s'envien les dues posicions als registres corresponents.

La funció sendLcdClear(int,int) té dos paràmetres d'entrada i cap de sortida. El primer paràmetre és l'identificador i el segon la fila que es vol esborrar. Si el segon paràmetre és 0, s'esborra la primera fila, si és 1 la segona i si és 2 les dues files. Per esborrar la pantalla s'omplen totes les posicions amb espais. Al final de la funció es posa el cursor a la primera posició de la pantalla.

Finalment, la funció LCD_ClosI2C(int) tanca la comunicació I2C de la pantalla, té un paràmetre d'entrada que és d'identificador (ha de ser un enter) i no té cap retorn.

Si es vol utilitzar aquest útil, només s'ha d'incloure la llibreria WirngPi i el nom de l'útil que és "Lcd_I2C_Ardubot.h". Aquest útil s'ha de trobar a la mateixa carpeta que el programa realitzat. Seguidament s'ha d'inicialitzar la comunicació I2C de la pantalla i s'ha d'inicialitzar la pantalla. Un cop es té la pantalla inicialitzada es poden fer servir totes les funcions indiferentment.

5.2 Comprovació de funcionament

Per assegurar un correcte comportament del robot s'han fet dos tipus de proves. Primer de tot s'han fet diverses proves unitàries per comprovar que totes les parts funcionin correctament. En el món de la programació, les proves unitàries o de components consisteixen a realitzar proves sobre unitats més petites de codi.

Aquestes proves són petits codis independents que han de funcionar per ells mateixos i que han de donar un resultat clar. En el cas que ens ocupa s'ha comprovat el correcte ús de totes les funcions necessàries per a utilitzar l'Ardubot 4.1. S'ha testejat el correcte funcionament de les funcions per a les entrades i sortides digitals, entrades i sortides analògiques, sortides PWM, interrupcions per hardware i ús de la pantalla LCD.

Un cop s'han creat, passat i aprovat totes les proves unitàries es pot crear un test funcional. Aquest test serveix per avaluar un producte en un punt de la seva creació i la seva finalitat sempre serà mostrar l'estat del producte. Amb l'Ardubot i la placa adaptadora s'ha creat un test funcional que demostra el correcte funcionament de tots els seus sensors i actuadors. A la Figura 24 es mostra la diferència entre aquests dos testos.

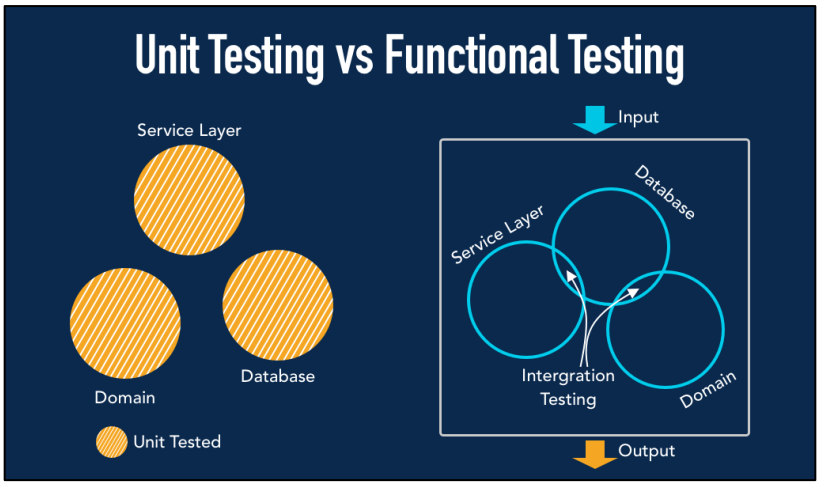


Figura 24. Diferència entre test unitari i test de funcionament.

En aquest programa, el robot actua com un seguidor de línia, utilitzant els 4 sensors i els dos motors. A més a més s'utilitzen les interrupcions dels sensors del codificador rotatiu per a fer un comptador; la pantalla amb comunicació I2C per a visualitzar l'estat de la bateria i els comptadors i els para-xocs per a fer una interacció amb els tres LEDs. A la Figura 25 es veu el diagrama de flux de tot el test.

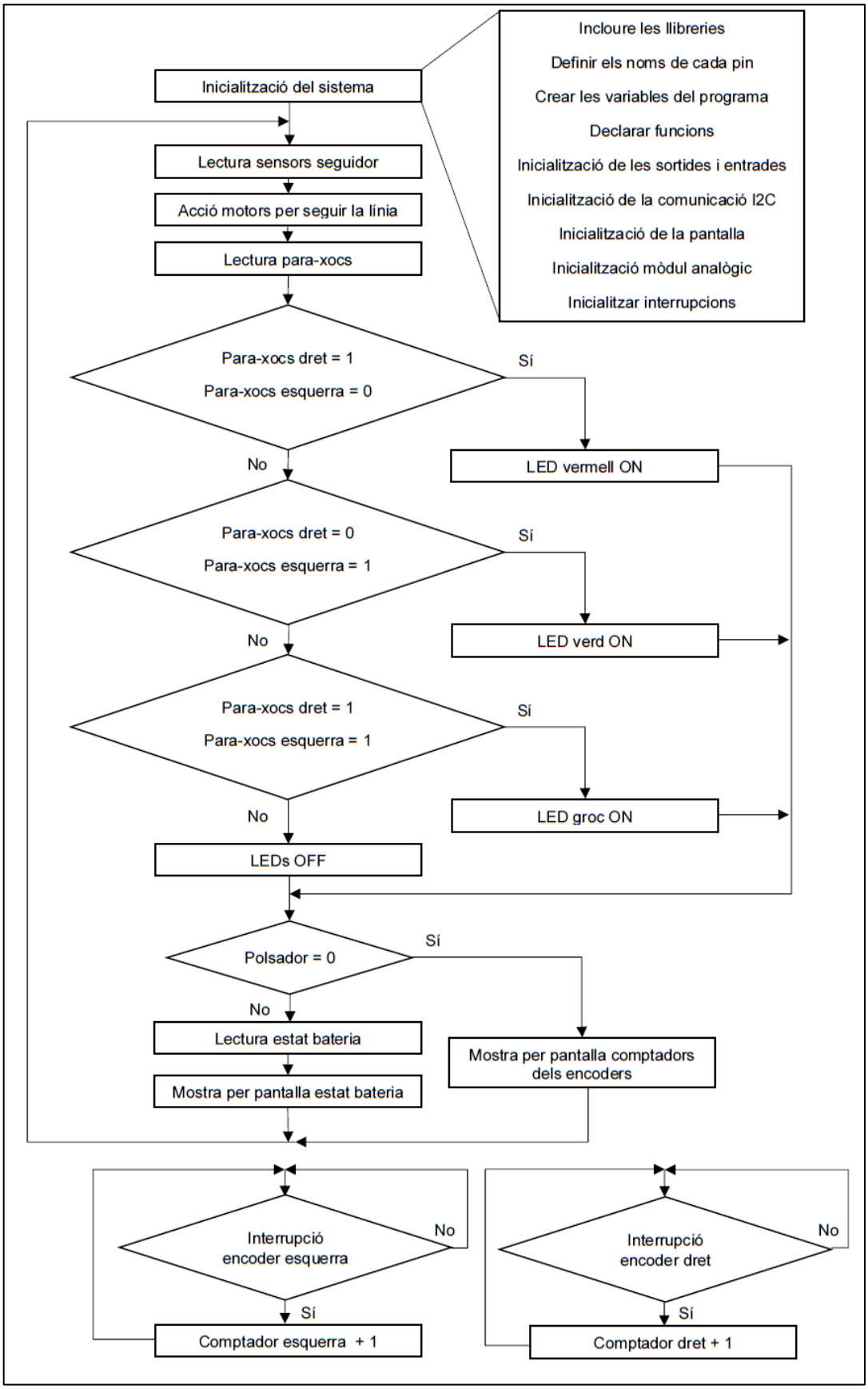


Figura 25. Diagrama de flux del programa de test funcional.

Quan s'inicia un codi, primer s'inclouen les llibreries stdio.h, stdlib.h, WiringPi.h, WiringPiI2C.h, error.h i l'útil per a la pantalla LCD. Seguidament es defineixen les constants i variables del programa. A continuació hi ha el bloc d'inicialització de les comunicacions diferents elements. Que és la part del programa que només s'executa una vegada (al principi) i on s'han de definir si els pins són de sortida o entrada. En aquest punt també es descriuen les interrupcions externes, que s'utilitzaran per a gestionar els dos codificadors rotatius de la roda dreta i esquerra. Es defineix el pin en qüestió, el nom de la interrupció i si es tracta d'una interrupció per canvi o per valor.

Tot seguit hi ha el bucle while(1), on es troba el programa principal que s'anirà executant sempre. Es veuen tres accions diferenciades dins el bucle. Primer de tot hi ha el seguidor de línia, seguidament la interacció entre el para-xocs i els tres LEDs i per acabar l'escriptura de la pantalla en funció del polsador.

La part del seguidor està feta de tal manera que se segueix una línia clara sobre un fons negre. Com que el seguidor té 4 sensors existeixen 16 combinacions possibles, de les quals només se n'han definit 6. Hi ha moltes combinacions que són impossibles d'obtenir en un seguidor convencional, de totes maneres hi ha combinacions possibles que no s'han tingut en compte, com per exemple que només es detectés la línia en una de les puntes. Quan es troba un cas que no s'ha contemplat es para el robot. Perquè el robot segueixi recte es mantenen les dues rodes a la mateixa velocitat. Per a fer girar el robot cap a un costat es fa girar més ràpid la roda del costat contrari al que es vol girar, com més gran sigui la diferència entre de dues rodes més ràpid girarà.

La interacció entre els para-xocs i els tres LEDs és molt senzilla, quan es toca el para-xocs esquerra s'encén el LED verd, quan es toca el para-xocs dret s'encén el LED vermell, quan es toquen els dos para-xocs s'encén el LED groc i quan no es toca cap para-xocs no se n'encén cap.

Pel que fa a la pantalla LCD, quan no es prem el polsador a la segons fila es mostra l'estat de la bateria. Per obtenir aquest valor es divideix entre 2,5 la lectura analògica, ja que una entrada analògica anirà entre 0 o 255, i el potenciòmetre de la bateria està ajustat de tal manera que proporcioni 5 V quan la bateria està carregada del tot, per tant marxarà 100% de la bateria. Quan es prem el polsador la segona línia de la pantalla mostra el nombre de cops que el codificador rotatiu ha canviat d'estat.

Aquest comptatge es fa a les interrupcions, que es troben fora del main(), just després. Cada interrupció té el seu petit programa, que és un comptador. Quan s'entra a la interrupció se suma 1 a una memòria i seguidament se surt de la interrupció. Les interrupcions ha de ser el més cures possibles, ja que no han d'interferir al programa principal. En total hi ha dues interrupcions, la del codificador dret i la del codificador esquerre. En aquest punt del codi és on hi hauria les funcions si es necessitessin.

Tant aquest programa com les proves unitàries es guarden a l'escriptori del Raspberry juntament amb els exemples, per entendre el funcionament i programació de les diferents parts de l'Ardubot a través de la placa adaptadora per a Raspberry Pi 0 W. A l'Escriptori es troba una carpeta anomenada "Exemples", al seu interior hi ha diverses carpetes amb el nom dels diferents elements, per exemple la carpeta Digitals té tres programes, l'anomenat Entrades, l'anomenat Sortides i finalment blink. Els programes Entrades i Sortides són proves unitàries de les entrades i sortides digitals. El programa blink, és un exemple que s'utilitza a les pràctiques per explicar el funcionament dels elements digitals i les esperes. Finalment la Carpeta anomenada Funcionament hi ha el tes funcional explicat anteriorment.

Aquests arxius poden ser llegits i executats per tothom però només modificats per l'usuari que els ha creat. Per a fer-ho s'han modificat els permisos de tal manera que queden rwxr-xr-x. Per fer-ho s'ha executat la comanda chmod 755 programa.c, per a tots els programes que hi ha a la carpeta d'exemples.

6 PRÀCTIQUES

A l'assignatura de la UdG Aplicacions industrials dels microprocessadors s'utilitza l'Ardubot com a plataforma física d'aprenentatge. A la fitxa de l'assignatura es veu que 20 % dels crèdits de l'assignatura es dediquen a classes teòriques, un 50 % a les pràctiques i un 30 % al treball final amb els robots. Així doncs, el 80 % de l'assignatura gira entorn dels robots. Amb la placa adaptadora es vol augmentar el nivell de les pràctiques mantenint la corba d'aprenentatge.

Per fer-ho es crearà un manual d'usuari per a introduir les característiques bàsiques del Rasberry i l'entorn de programació. Es crearà un dossier de pràctiques on s'actualitzaran les existents i se'n crearan d'específiques per al Raspberry que ensenyin als alumnes a penjar dades a internet i utilitzar la càmera. D'aquesta manera es podrà començar amb el nivell més baix i aprendre com es poden programar tots els elements.

Per a fer les pràctiques s'utilitzarà el llenguatge C i l'entorn de programació Code Blocks. Aquest, és un entorn de desenvolupament integrat de codi obert. Està orientat al llenguatge de C, C++ i Fortain i incorpora diversos compiladors com són GCC, Clang i Visual C++. Des d'aquest programa es poden crear arxius i projectes nous, compilar, executar i depurar sense menester cap altra aplicació. A la Figura 26 es mostra quin aspecte té l'aplicació.

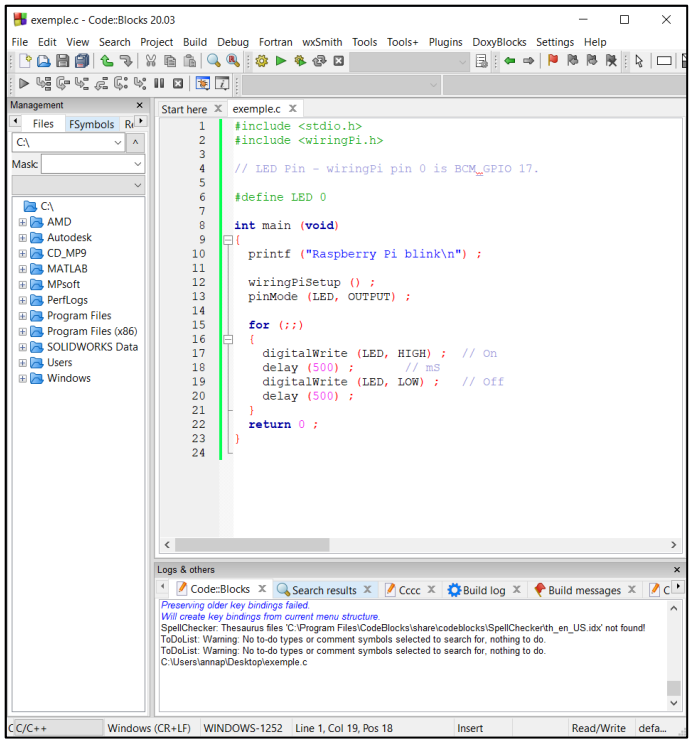


Figura 26. Entorn de programació Code::Blocks.

Es pot veure una barra d'eines a la part superior, un gestor de fitxers a l'esquerra, un visualitzador de missatges a la part baixa i al centre la zona on es pot desenvolupar el codi. Quan s'indica de quin llenguatge es tracta, automàticament es marca el codi de diferents colors per a facilitar la programació. També disposa d'autoemplenat, una eina molt útil que diu quins mots coincideixen amb el que s'està escrivint.

Tot i que el Raspberry es pot programar amb molts de llenguatges (Python, Java, C++, C#...), s'ha decidit utilitzar el C, ja que és molt similar a l'Arduino i a més a més és el que s'aprèn a l'assignatura d'Informàtica Industrial. L'objectiu de l'assignatura d'Aplicacions industrials dels microprocessadors no és la d'ensenyar nous llenguatges de programació sinó la de consolidar coneixements apresos i aplicar-los en entorns nous. Per tant, el C és el més indicat i per a facilitar el seu ús, s'utilitzarà la llibreria WiringPI. Aquesta és una llibreria totalment lliure i gratuïta, que permet gestionar les entrades i sortides de propòsit general del Raspberry amb les mateixes funcions que s'utilitzarien amb l'Arduino.

6.1 Actualització de les pràctiques existents

Actualment a l'assignatura es realitzen 7 pràctiques. A la primera s'utilitzen les entrades i sortides digitals (LEDs, polsador i para-xocs) i la pantalla LCD; a la segona s'introdueix l'entrada analògica i les conversions de l'ACD a voltatge; a la tercera el PWM i els encoders; a la quarta els seguidors de línia, a la cinquena s'implementa un control PID de velocitat, a la sisena es fa una comunicació amb un Bluetooth i a l'última s'explica la comunicació I2C.

6.1.1 Pràctica 1, diàleg home-màquina

El robot disposa de diversos elements d'interfície home-màquina i a la inversa. Els més bàsics són 3 LEDs, un de verd, un de groc i un de vermell i un polsador. A més a més té una pantalla LCD de 16x2 caràcters i dos para-xocs, que també es poden utilitzar a manera de polsador. El Raspberry també té la capacitat de mostrar informació a través del terminat. En aquesta primera pràctica l'objectiu és que els alumnes comencin a desenvolupar petits programes a l'hora que familiaritzin amb l'entorn de programació i amb el robot. En aquests programes s'utilitzaran les sortides digitals i la pantalla LCD amb comunicació I2C.

Com s'ha explicat anteriorment, les sortides digitals del Raspberry no poden aportar més de 16 mA, i es recomana no superar el 8. Per assegurar un bon funcionament, a la placa adaptadora s'han instal·lat tres transistors que aportaran la intensitat necessària. El voltatge

lògic d'1 és de 3,3 V, tant a les sortides com a les entrades. El temps per a canviar l'estat d'una sortida és de 1,7 ns.

Un cop explicades les característiques elèctriques toca parlar de les de programació. Per a utilitzar les sortides i entrades analògiques es faran servir les mateixes funcions que en l'Arduino Uno gràcies a la llibreria WiringPi. Per contra, no es disposa d'una llibreria per a la pantalla LCD amb I2C així que s'ha creat una llibreria amb les funcions bàsiques per a utilitzar-la. Les quatre funcions són la d'inicialitzar la pantalla, la d'escriure a la pantalla, la de moure el cursor i la de buidar la pantalla. S'incorpora també un programa d'exemple de funcionament. Finalment es veurà que amb el Raspberry Pi és molt senzill mostrar informació pel terminal.

6.1.2 Pràctica 2, mòdul analògic

La placa adaptadora incorpora un mòdul analògic de comunicació I2C. A la pràctica dos s'introdueixen les entrades analògiques que s'utilitzarà per a llegir el voltatge de la bateria. També s'introdueix la conversió del valor d'ADC a Voltatge i la creació de funcions. A més a més aquests valors es mostren per la pantalla, per tant consoliden l'ús d'aquest element. A la versió nova de la pràctica també s'utilitzarà la sortida analògica que podrà alimentar un LED.

El mòdul analògic és el PCF8591, és un mòdul de baixa potència i comunicació I2C. Té 4 entrades analògiques i una sortida analògica de 8 bits de conversió. Pot donar un màxim de 20 mA per a la sortida i admet un voltatge màxim a les entrades igual al voltatge d'alimentació més mig volt. La freqüència de conversió del DAC és d'11,1 kHz i la freqüència de mostreig de les entrades també és d'11,1 kHz com a màxim.

Les funcions que s'utilitzaran per a les entrades i sortides analògiques són les mateixes que en l'Arduino, però amb petites diferències. Primer de tot s'ha d'incloure la llibreria del xip i seguidament conèixer l'adreça per inicialitzar el mòdul. Com que es pot tenir més d'un dispositiu, aquesta llibreria t'obliga a donar-li un número major a 64 que identifiqui el mòdul. Aquest valor se suma al número de sortida o entrada per identificar de quin element el tracta.

6.1.3 Pràctica 3, motors amb PWM i interrupcions

El robot disposa de dos motors de corrent continua i dos codificadors rotatius relatius. En aquesta pràctica es vol introduir el concepte de PWM, el seu funcionament i les interrupcions asíncrones i temporitzacions. Per fer-ho primer es configura i es modifica la velocitat dels

motors i seguidament es calcula la velocitat del robot utilitzant les interrupcions externes que creen els sensors de l'encoder i la temporització per saber quan polsos hi ha per segon. A més a més, s'explicarà de diferència entre el PWM per software i hardware en el Raspberry.

El voltatge nominal del motor és de 12 V i té un consum de 0,3 A, la velocitat nominal amb càrrega a aquest voltatge és de 6300 rpm i té un parell de 25 g-cm. Tot i que els motors s'alimenten a 9 V, la velocitat del motor és molt gran. Per aquest motiu hi ha uns engranatges reductors, que redueixen la velocitat alhora que augmenten el parell. Cada un d'aquests engranatges té una relació de 2 i n'hi ha dos, per tant, els engranatges redueixen 4 cops la velocitat i augmenten 8 cops el parell. Així doncs, la velocitat final de la roda és de 1.575 rpm amb un parell de 100 g-cm.

El Raspberry Pi disposa de dos tips diferents de PWM, el físic i el creat per software. La primacial diferència és que el físic no ocupa espai de processament a la CPU, ja que està creat per uns rellotges interns que actives i desactiven la sortida de forma automàtica. La velocitat del PWM per hardware és de 100 MHz programable, la llibreria que s'utilitza per a crear el PWM per software dona una freqüència 100 Hz.

Les connexions a la placa adaptadora estan fetes de tel manera el PWM 0 intern del Raspberry alimenta un sentit de gir dels dos motors i l'1 el sentit de gir contrari. Així doncs es podrà comprovar el funcionament del PWM que té integrat sense requerir cap llibreria. En aquest cas, però, s'haurà d'executar el programa com a súper-administrador. Per altra banda s'utilitzarà la llibreria que crear un senyal PWM a qualsevol sortida GPIO per a moure el robot amb normalitat.

Les interrupcions no requereixen cap llibreria extra i se'n poden crear de tres tipus, de trac de pujada, de flang de baixada o de canvi (que detecta els dos flancs). Quan apareix la interrupció s'executa la funció indicada. Tot i que el Raspberry permet crear rellotges i interrupcions síncrones, s'utilitzarà la funció millis(), que funciona igual que en l'Arduino.

6.1.4 Pràctica 4, seguidor de línies

L'Ardubot té quatre sensors de contrast discretitzats i quatre LEDs que informen de l'estat dels sensors. A la pràctica quatre s'utilitzen per a seguir una línia en el terra i així consolidar el control de velocitat dels motors i la correcta utilització dels bucles i estructures bàsica de

programació en C. A la Figura 27 es mostra com es distribueixen els diferents sensors de contrast i com es sobreposen a la línia en diferents moments.

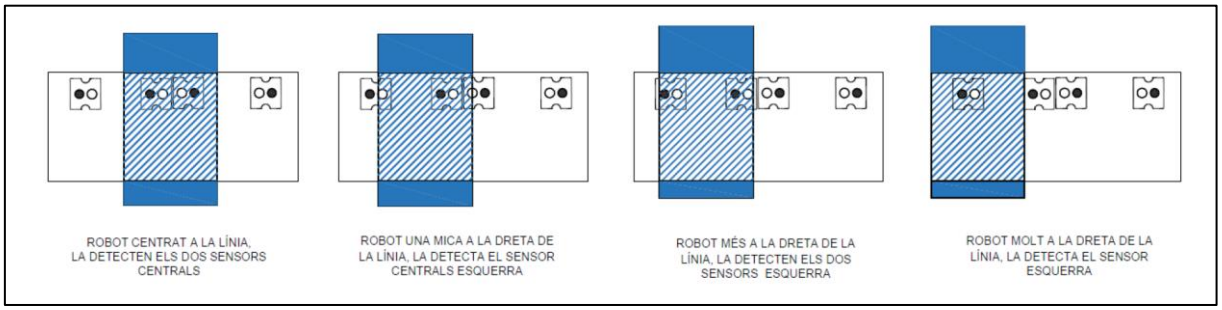


Figura 27. Superposició dels sensors de contrast a la línia en diferents moments.

Es pot observar que la distribució dels sensors està pensada per seguir una línia de 16 mm d'ample, creada amb una cinta aïllant estàndard. Els quatre sensors s'utilitzen per a saber quin grau de desviació té el robot, en funció de quins sensors s'activen es pot actuar més bruscament o menys.

6.1.5 Pràctica 5, control amb PID

Amb el coneixement après i consolidat a les pràctiques 3 i 4, on es modifica la velocitat dels motors, s'implementarà un control PID de la velocitat del robot. En aquesta pràctica es presenta l'equació bàsica d'un control PID i uns valors de control obtinguts a partir d'una prova empírica. Un cop implementat el PID els alumnes poden anar modificant la freqüència de mostreig i les constants de P i de I per observar quin comportament té. Per a crear el PID s'utilitzen les funcions de PWM i interrupcions dels encodes. A més a més també es crearan interrupcions síncrones per a tenir un control precís de la velocitat en cada moment.

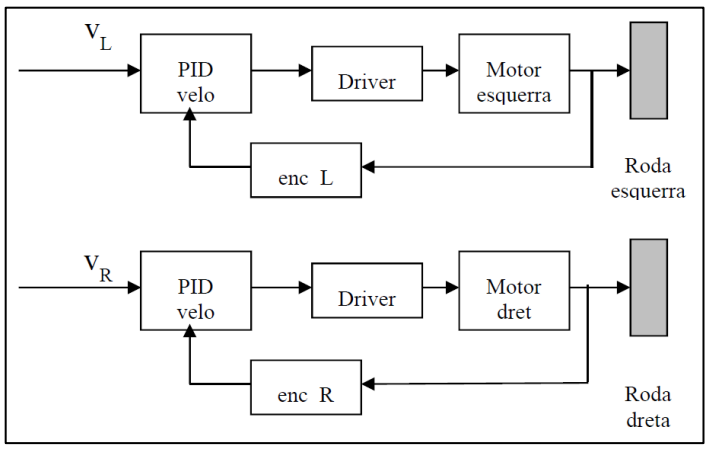


Figura 28. Diagrama de blocs del control dels dos motors.

A la Figura 28 es mostra el diagrama de blocs del control que s'utilitzarà per controlar la velocitat del robot. Es crearan dos controls independents un per a cada roda, ja que encara que s'enviï donar el mateix valor de PWM els dos motors tindran una velocitat diferent. Per a obtenir els valors de les constants del PID s'ha utilitzat el mètode de Zieger-Nichols.

6.1.6 Pràctica 6, comunicació amb BT

Originalment, l'Ardubot requeria un Bluetooth extern per a poder-se comunicar utilitzant aquesta tecnologia. El Raspberry Pi 0 W incorpora un Bluetooth, per tant no es requereix cap element extra. En aquesta pràctica s'utilitza una aplicació de mòbil creada amb AppInventor per connectar-se i controlar el robot a través de Bluetooth. L'Aplicació de mòbil enviarà una lletra diferent per a cada comanda. S'ha de programar el robot perquè actuï correctament al rebre les diferents lletres. Així doncs es coneixerà la comunicació serè Bluetooth, es modificarà una aplicació de mòbil amb AppInventor i es programarà el robot per a ser controlat a distància.

El Raspberry Pi 0 W té un Bluetooth 4.1 Low Energy. Té una banda de freqüència de 2,4 GHz, un consum inferior a 15 mA i un rang teòric superior a 100 m. Per un correcte funcionament primer s'ha d'iniciar la connexió entre el Raspberry i el telèfon mòbil a través del terminal. D'aquesta manera es força la connexió del telèfon i es té la informació de l'adreça del telèfon i del nom en el Raspberry.

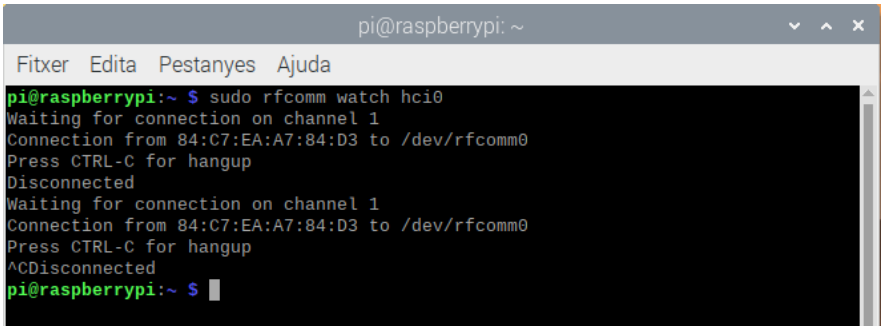


Figura 29. Comanda per a forçar la connexió i la informació de la conneció.

Seguidament s'utilitza la llibreria de comunicació sèrie del WiringPi per iniciar la comunicació, llegir les dades enviades i tancar la comunicació. Amb els missatges que es reben del port sèrie del Bluetooth es comanda el robot perquè es mogui a voluntat. També es poden fer altres accions com visualitzar la bateria o canviar el mode manual a automàtic perquè faci de seguidor de línies.

6.1.7 Pràctica 7, de comunicació I2C

La pràctica 7 s'introduïa la comunicació I2C, en les pràctiques actuals no es requereix fer cap introducció a aquesta tecnologia perquè tant la pantalla com el mòdul analògic es comuniquen amb I2C amb el Raspberry. Tot i això, una part interessant de la pràctica era l'ús d'un mòdul RTC sense llibreria. Això requereix que l'alumne llegeixi amb profunditat el full de característiques de l'element i entengui el seu contingut. S'utilitzarà el rellotge de temps real DS3231 que a més a més permet crear alarmes.

El Raspberry Pi disposa d'una eina molt útil que és el detector d'elements I2C. Aquesta comanda s'utilitzarà per a comprovar que l'adreça del dispositiu RTC és la mateixa que la que es diu al full de característiques. A la Figura 30 es mostra quin aspecte té la funció executada al terminal.

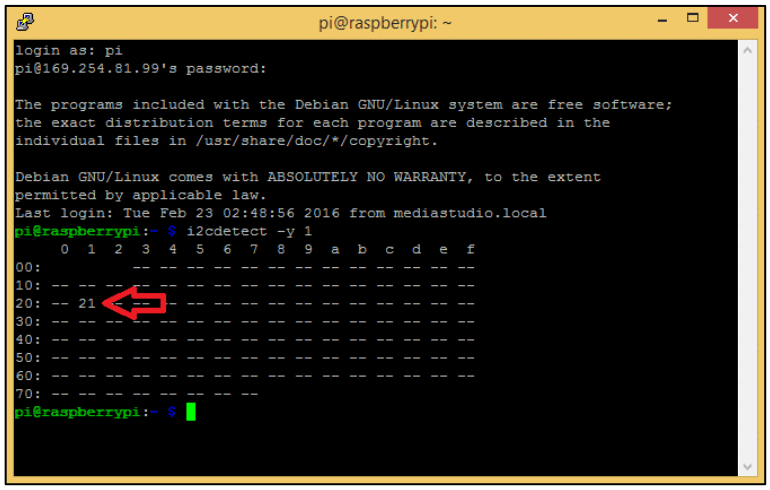


Figura 30. Terminal de Raspberry Pi un cop s'executa la comanda per a detectar adreces I2C.

6.2 Pràctiques noves

Un cop els estudiants hagin assimilat tots els conceptes bàsics de les pràctiques antigues es proposen un parell de pràctiques noves que aprofiten part del potencial del Raspberry. A la primera de les noves pràctiques s'enviaran dades a un broker MQTT per simular una connexió típica de l'IIOT. Amb aquesta pràctica, a part d'enviar informació a un servidor local, els robots podran rebre informació per a modificar el seu comportament i així poder crear una xarxa de comunicació entre ells. A la segona s'utilitzarà el Raspberry Pi i una càmera per a fer un petit sistema de videovigilància amb emmagatzematge d'imatges gràcies a la detecció de moviment, també es podrà visualitzar la imatge en directe a través de qualsevol navegador d'un ordinador que es trobi a la mateixa xarxa.

6.2.1 Pràctica 8, Comunicació a un servidor local MQTT

El Raspberry Pi té connexió a internet gràcies al WiFi que disposa. S'aprofitarà aquesta connexió per a comunicar-se amb un broker MQTT, que és el que gestiona els missatges. Els clients (per exemple un Raspberry) poden estar subscrits a un tòpic o bé publicar-lo. Els tòpics són rutes estructurades amb forma d'arbre, de manera que si s'està subscrit a un tòpic de més jerarquia, rebràs tots els missatges dels tòpics que es troben per sota. A la Figura 31 es mostra un exemple de la jerarquia dels tòpics, on si estàs subscrit al tòpic del laboratori "LabSistInt" rebràs els missatges del laboratori i de tots els robots, però si només estàs subscrit al tòpic del robot "Ardubot2" només rebràs aquests missatges.

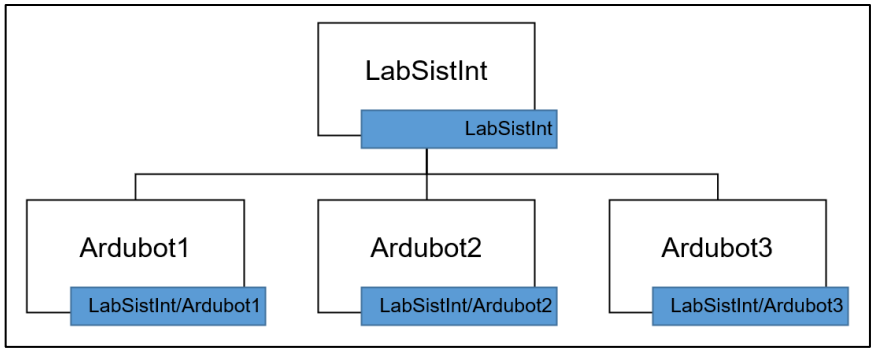


Figura 31. Jerarquia dels tòpics en l'MQTT.

En aquesta pràctica, s'utilitzarà un Raspberry Pi com a broker i la resta de Raspberrys estaran subscrits i publicaran als diferents tòpics. S'ensenyaran les bases de la comunicació MQTT i s'utilitzarà el terminat per a subscriure's i publicar als diferents tòpics. També s'utilitzarà el software MQTT Explorer a l'ordinador del laboratori per comunicar-se a través aquesta tecnologia al Raspberry pi i encendre i apagar un LED de l'Ardubot. Finalment es crearà una comunicació entre Raspberrys utilitzant la comunicació MQTT. Els alumnes publicaran l'estat del seu polsador a un tòpic utilitzant un programa en c i es subscriuran al tòpic del polsador d'un altre robot per encontre un LED.

6.2.2 Sistema de videovigilància

El Raspberry Pi 0 W té un connector mini-CSI, per a instal·lar-hi una càmera. Es disposa d'una càmera de 640x480 píxels i un màxim de 30 mostres per segon. La càmera es connecta al Raspberry Pi estirant la llengüeta negra del connector. Seguidament s'insereix el cable a l'escletxa que apareix, amb la part metàl·lica mirant cap a la placa. Un cop instal·lada queda com es mostra a la Figura 32.

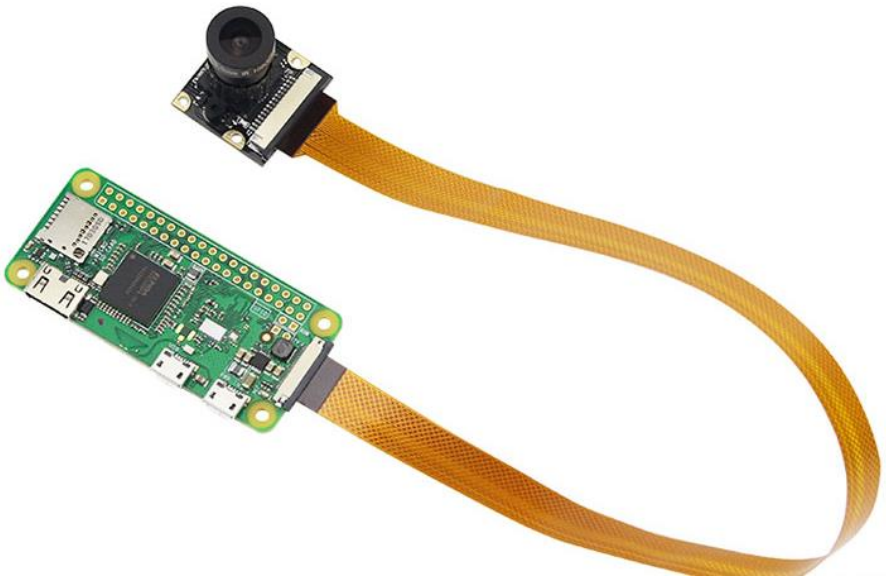


Figura 32. Raspberry Pi 0 W amb una càmera.

Amb aquesta càmera i el programa anomenat Motion, es crearà un sistema de videovigilància. Modificant els paràmetres de configuració ens permetre veure la imatge des de qualsevol ordinador dins la mateixa xarxa. A més a més es detectarà el moviment per a guardar una imatge amb un requadre que marquí on apareix el moviment. Totes les configuracions es faran a través de terminal i pàgina web.

7 RESUM DE PRESSUPOST

Per a crear una placa adaptadora per a substituït l'Arduino Uno a l'Ardubot 4.1 per un Raspberry Pi 0 W. S'ha comprovat que l'autonomia de robot és suficient per a fer una pràctica de tres hores i s'han dissenyat unes rodes que soluciona el problema del desgast de l'encoder. També s'ha adaptat les pràctiques existents i se n'han creat de noves per ensenyar tot el potencial de la placa.

L'import sense IVA del projecte és de cinc mil cent noranta euros amb vint-i-set cèntims.

8 CONCLUSIONS

Per a facilitar la connexió a internet del robot es substitueix l'Arduino Uno per un Raspberry Pi 0 W. S'ha dissenyat una placa adaptadora que permet connectar el Raspbery Pi seleccionat a l'Ardubot 4.1. Amb aquesta placa es pot actuar sobre tots els actuadors i es poden llegir tots els sensors del robot. Per comprovar el correcte funcionament de la placa i el Raspberry s'han creat diversos testos unitaris i un test de funcionament.

S'ha comprovat que l'autonomia del robot és suficient per a fer una pràctica estàndard de 3 hores. S'ha dissenyat una roda per imprimir amb tecnologia 3D que soluciona el problema del desgast dels encoders.

També s'han adaptat les pràctiques actuals pel nou controlador i s'han creat dues pràctiques noves per mostrar el potencial del Raspberry. Aquestes noves pràctiques mostren l'ús d'una càmera i utilitzen la tecnòloga MQTT per enviar dades a un servidor i comunicar-se entre els robots de l'aula.

Anna Planas Bahí
Graduada en Enginyeria Electrònica Industrial i Automàtica.

La Bisbal d'Empordà, 28 de maig del 2020.

9 RELACIÓ DE DOCUMENTS

El projecte del Robot i pràctiques per aprendre a programar sistemes encastats consta de cinc documents independents, aquests són la memòria, els plànols, el plec de condicions, l'estat d'amidaments i el pressupost.

10 BIBLIOGRÀFIA

ABELLAN, M., Detecta movimientos con una webcam y Motion en Raspberry Pi (https://www.programoergosum.es/tutoriales/webcam-con-motion-en-raspberry-pi/, 30 de maig de 2020).

BROADCOM CORPORATION. Full de característiques del BCM2835 ARM Peripherals. (https://www.raspberrypi.org/documentation/hardware/raspberrypi/bcm2835/BCM2835-ARM-Peripherals.pdf, 19 d'abril de 2020).

DESCUBREARDUINO. 5 maneras de escritorio remoto en Raspberry (Windows/Linux/Mac). (https://descubrearduino.com/escritorio-remoto-en-raspberry-pi/, 10 d'abril de 2020).

FIGUERAS, A., HESSE, R. Taller de robots de rescat Ardubot. Dossier de teoria. 2011.

FIGUERAS, A., Moodle de l'assignatura Aplicacions industrials dels microprocessadors (https://moodle2.udg.edu/course/view.php?id=23731, 10 de maig de 2020).

ICRAGGS, MQTT Client Library for C GitHub (https://github.com/eclipse/paho.mqtt.c, 28 de maig de 2020).

MOTION, Pàgina del programa Motion (https://motion-project.github.io/index.html, 30 de maig de 2020).

NXP B.V., Full de característiques del PCF8591. (https://www.nxp.com/docs/en/data-sheet/PCF8591.pdf, 20 de maig de 2020).

PÉREZ, M., Tutorial Raspberry Pi – GPIO y MQTT (https://geekytheory.com/tutorial-raspberry-pi-gpio-y-mqtt-parte-1, 28 de maig de 2020).

RASPBERRY, Raspberry Reference. (https://www.raspberrypi.org/, 5 d'abril de 2020).

TEXAS INSTRUMENTS INC, Full de característiques del LM2596 Simple Switcher. (http://www.ti.com/lit/ds/symlink/lm2596.pdf, 20 de maig de 2020).

UITHOVEN, P., AMERSFOORT, F. OpenSCAD Reference. (www.openscad.org/cheatsheet/, 18 de maig de 2020).

WIN32 DISK IMAGER, Win32 Disk Imager download. (https://win32diskimager.download/, 6 d'abril de 2020).

WIRINGPI, WiringPi GitHub. (https://github.com/WiringPi/WiringPi, 12 d'abril de 2020).

WIRINGPI, WiringPi library Reference. (http://wiringpi.com/reference/, 12 d'abril de 2020).

11 GLOSSARI

ADC: Analog to Digital Converter

APP: Application

ARLab: Agents Research Laboratory

ARM: Advanced RISC Machines

BT: Bluetooth

CNC: Control Numèric per Computadora

CPU: Central Processing Unit

CS: Chip Selector

CSI: Camera Serial Interface

DAC: Digital to Analog Converter

DNC: Do Not Connect

EEEA: Enginyeria Elèctrica, Electrònica i Automàtica

FTP: File Transfer Protocol

GCC: GNU Compiler Collection

GND: Ground

GPCLK: General Purpose Clock

GPIO: General Purpose Input/Output

HDMI: High Definition Multimedia Interface

I2C: Inter-Integrated Circuit

IDE: Integrated Devolopment Environment

IOT: Internet of Things

IP: Internet Protocol

ISM: Industrial, Scientific and Medical

IVA: Impuesto sobre el Valor Añadido

LCD: Liquid Crystal Display

LED: Leght-Emitting Diode

MISO: Màster Input Slave Output

MOSFET: Metal-Oxide Semiconductor Field-Effect Transistor

MOSI: Master Output Slave Input

MQTT: Message Queuing Telemetry Transport

PCB: Printed Circuit Board

PID: Proporcional-Integral-Derivative

PWM: Pulse Whith Modulation

PWR: Power

RTC: Real Time Clock

RX: Receive

SCL: Serial Clock

SCLK: Serial Clock

SD: Secure Digital

SDA: Serial Data

SMD: Surface-Mount Technology

SO: Sistema Operatiu

SPI: Serial Peripheral Interficie

SQW: Square Wave

SSH: Secure Shell

TPU: Termoplastic Polyurethane

TX: Transmit

UART: Universal Asynchronous Receiver Transmitter

UdG: Universitat de Girona

USB: Universal Serial Bus

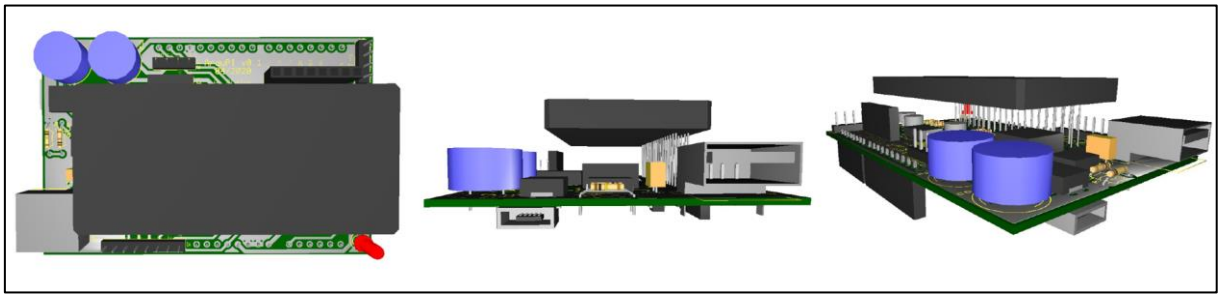
Vdc: Voltage Direct Current

VNC: Virtual Network Computing

WiFi: Wireless Fidelity

A MANUAL USUARI

La placada anomenada ArduPi és una placa adaptadora creada per a substituir l'Arduino Uno a la placa Ardunot 4.1. Tot i ser dissenyada per realitzar aquesta funció, es pot utilitzar com a un element independent. A la part superior de la placa hi ha els connectors per a connectar-hi un Raspberry Pi 0 W.



Il·lustració 1. Simulació 3D de la placa adaptadora.

Es pot alimentar entre 7 V i 40 V a través del pin anomenat Vin o bé amb 5 V a través del connector per potència. Per alimentar el Raspberry correctament es connecta un cable USB-miniUSB entre l'USB de la placa i del Raspberry. Hi ha un LED que indica la correcta connexió del Raspberry a la placa adaptadora.

El Raspberry disposa d'un connector miniHDM per a connectar-ho una pantalla i un miniUSB per a poder-hi connectar un teclat i un ratolí. D'aquesta manera o bé a través d'escriptori remot es pot accedir al Raspberry. Al seu interior hi ha els programes bàsics per a programar-lo i exemples per entendre el seu funcionament.

A part dels headers de la part inferior amb la mateixa forma que els de l'Arduino, hi ha diversos connectors a la part superior de la placa, que es poden utilitzar indiferentment. A tots els elements s'indica el nom, tant del número que té a la llibreria WiringPi com de les funcions extres que tenen.

B DOSSIER DE PRÀCTIQUES

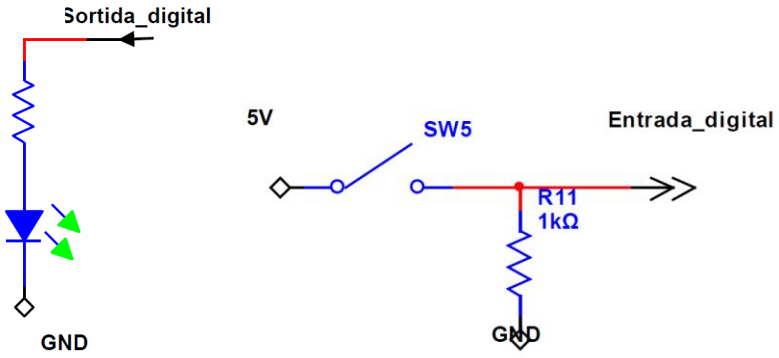
B.1 Pràctica 1: Diàleg home-màquina

B.1.1 Objectius

Els objectius de la pràctica de Diàleg home màquina són utilitzar les entrades i sortides digitals de robot, utilitzar la pantalla LDC del robot, fer programes amb llenguatge C i executar-los al Rasberry Pi 0 W.

B.1.2 Introducció

L'Ardubot disposa de tres LEDs, un polsador, dos para-xocs i una pantalla LCD per a poder rebre i enviar informació al Raspberry. Els tres LEDs són de color verd, taronja i vermell, com els d'un semàfor, i es poden activar a través d'una sortida digital. El polsador i els para-xocs actuen sobre una resistència de "pull-down" de forma que el polsador accionat equival a la lectura d'un "1" digital en el port i el polsador en repòs equival a la lectura d'un "0" digital. Es poden llegir a través d'una entrada digital.

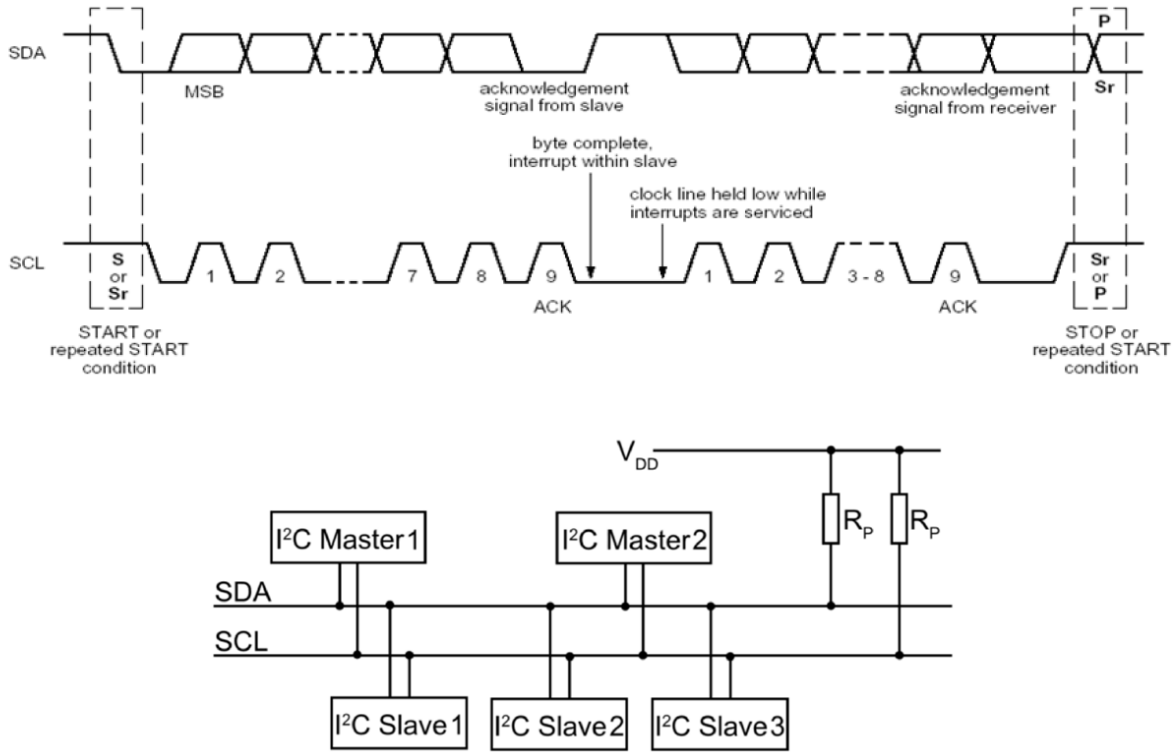


Il·lustració 1. Esquema d'un LED i un Polsador.

A la següent taula s'indica a quin pin físic es troba cada element i quin és el numero que té a la llibreria WiringPi.

Dispositiu	Pin físics / WiringPI del Raspberry
LED Verd	11 / 0
LED Taronja	13 / 2
LED Vermell	15 / 3
Polsador	22 / 6
Para-xocs Dret	16 / 4
Para-xocs Esquerra	18 / 5

La pantalla LCD es comunica amb el Raspberry a través de comunicació I2C. L'I2C (del anglès *Inter-Integrated Circuit*) és un bus de comunicació bidireccional sèrie de 2 cables, l'SCL que és el rellotge (que sincronitza les dades) i l'SDA que és la senyal de les dades. Aquestes dos senyals són binàries, per assegurar uns valor correctes de 1 i 0, s'ha de posar una resistència *pull up*.



Il·lustració 2. Comunicació i connexió de l'I2C.

En aquesta pràctica s'utilitzarà el Raspberry com a màster del bus per a fer anar diversos elements com a esclaus. La principal característica d'aquest bus de comunicació és que és munti-màster i multi-esclau. La diferència entre un màster (el Raspberry) i un esclau (el sensor o l'actuador) rau en que el màster és el que comença la comunicació. El bus és bidireccional i tots dos elements poden enviar i rebre dades però no en el mateix temps.

Quan el màster comença una comunicació envia d'adreça de l'element al que es vol adreçar. Tothom escolta i només l'element en qüestió respon "acknowledge". Quan això succeeix el màster envia la comanda. Seguidament l'esclau respon (si és necessari). Aquest procés es repeteix cada cop que es produeix una comunicació entre dos elements. Per aquest motiu és

imprescindible que cada element tingui una adreça única i irrepetible en tot el bus per a no tenir problemes de col·lisions.

La pantalla LCD és la que es troba a la part de davant del robot. Es tracta d'una pantalla de 2 files amb 16 caràcters cada una. El tipus de comunicació és I2C, per tant s'utilitza un bus de comunicació per enviar-li la informació. La pantalla és una interfície maquina-humà molt útil. Concretament és una pantalla de la marca MIDAS, del model MC21605C6W-SPTLYI-V2. S'alimenta a 5V i cada caràcter té 5x8 píxels. Segons el fabricant el pins s'han de connectar de la següent manera:

Pin Layout

PI	SYMBOL	DESCRIPTION	REMARKS
1	Vss	GND	
2	Vdd	Power supply for LCM	5.0V
3	V0	Operating Voltage for LCD	
4	NC	No Connection	
5	NC	No Connection	
6	NC	No Connection	
7	SA0	I2C interface, SA1 and SA0 slave	
8	SA1	address, connect to Vdd/Vss	
9	NC	No Connection	
10	NC	No Connection	
11	NC	No Connection	
12	CSB	Chip Select. Low=True High=False	
13	SDA	Serial Input Data	
14	SCL	Serial Clock Input	
15	A	Power Supply for BKL	
16	K	Power Supply for BKL	

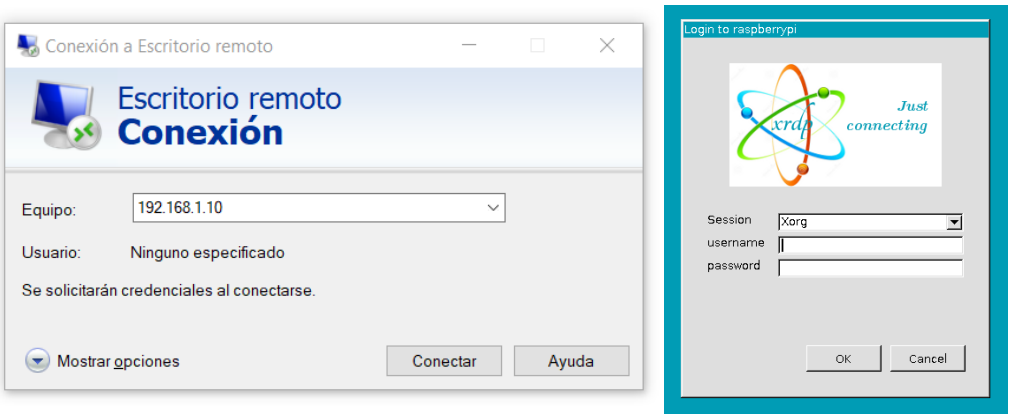
El pin 3 és el contrast necessari perquè es puguin distingir els caràcters de la pantalla, s'ajusta amb un potenciòmetre. El pin 7 i 8 són els que donen una adreça a la pantalla, en aquest cas els dos pins estan connectats a GND. L'adreça que s'obté fent aquesta connexió és 0x3C. El pin 12 també està a GND ja que així sempre es selecciona la pantalla. Per acabar els pins 13 i 14, es connecten al bus I2C del Raspberry. Per a utilitza la pantalla es farà servir una llibreria creada amb les funcions bàsiques.

Per a programat tots els dispositius s'utilitzarà la llibreria WiringPi, si es vol més informació es pot trobar a la seva pàgina web o al seu GitHub.

http://wiringpi.com/ o https://github.com/WiringPi/WiringPi

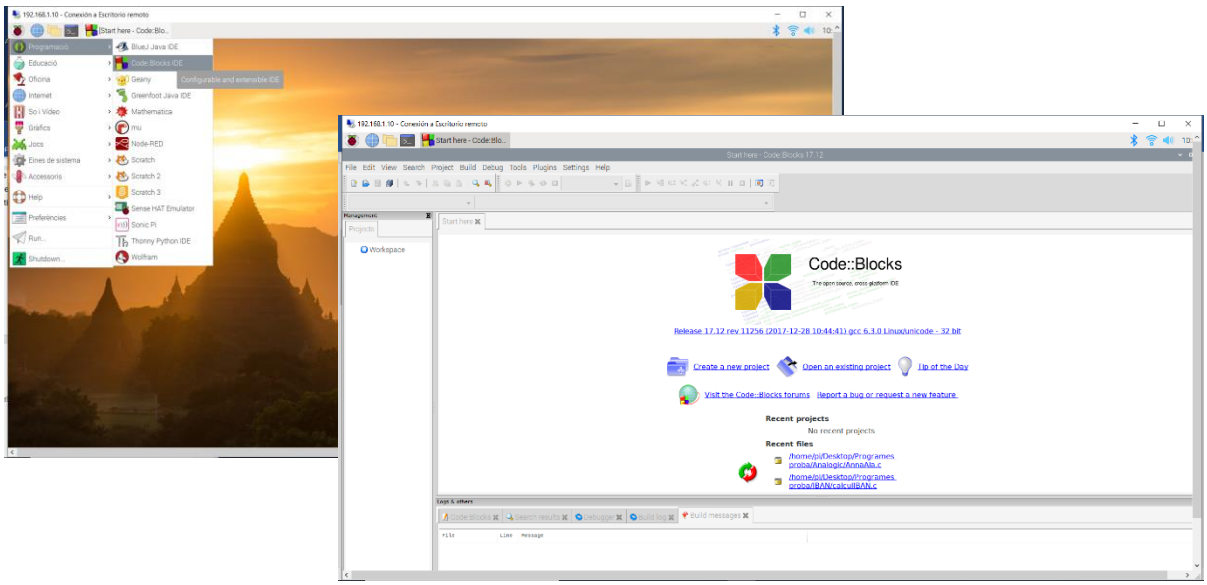
B.1.3 Interfície de programació

Per a programar el Raspberry primer s'ha d'establir una connexió amb escriptori remot. Per fer-ho s'ha d'obrir l'aplicació de Windows "Connexió a escriptori remot" i s'ha d'escriure quina és l'adreça del dispositiu. Per saber-la de manera fàcil s'ha donat una IP estàtica a cada Raspberry, des de la 192.168.1.11 a la 192.168.1.18. L'últim numero de l'adreça fa referència al numero de robot, que es troba enganxat a cada una de les plaques. Un cop es prem el botó de connectar apareix una finestra per iniciar sessió, l'usuari és **pi** i la contrasenya **AppluP**, d'Aplicacions industrials dels microprocessadors.



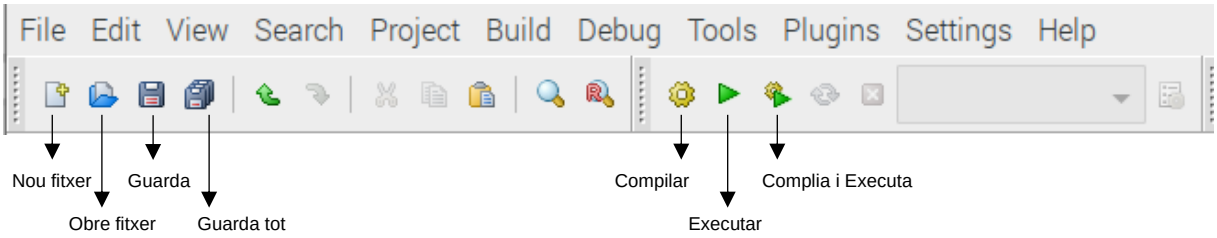
Il·lustració 3. Accés a l'escriptori remot del Rspberry.

Un cop s'ha accedit a l'escriptori del Raspberry, per obrir a la interfície de programació s'ha d'anar al menú del sistema, Programació i Code::Blocks.



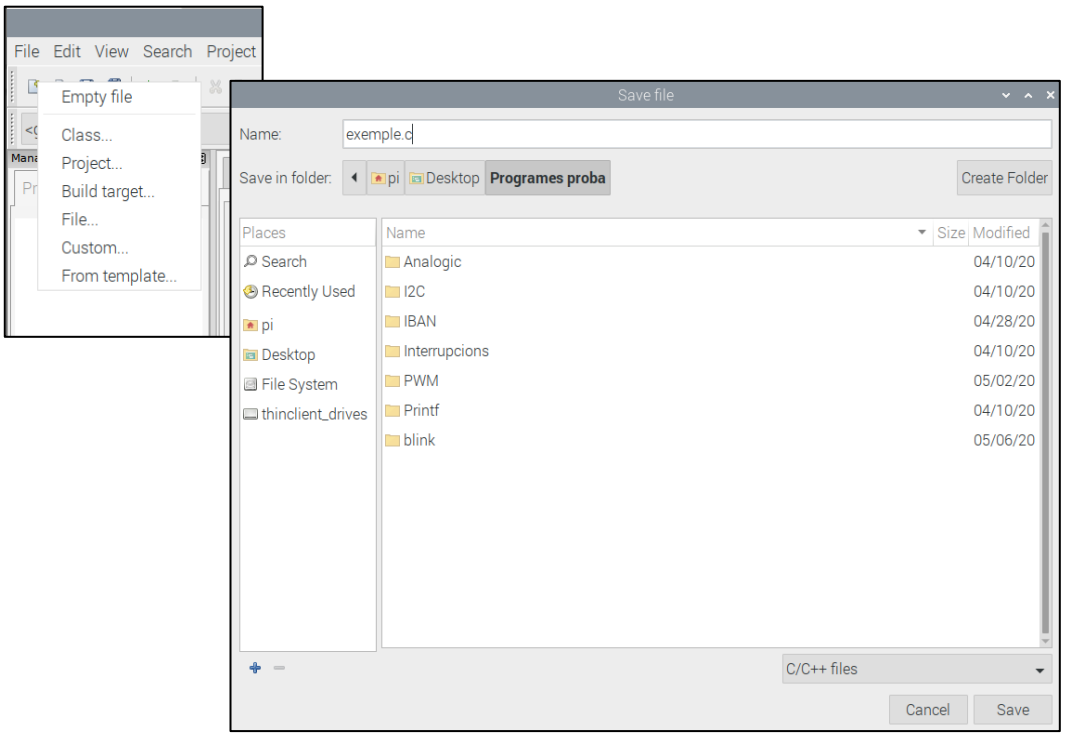
Il·lustració 4. Escriptori remot i nterficie de programació

Quan es té la interfície oberta, amb la barra de menú es pot crear un fitxer nou o bé importar-ne un d'existent. Si es té un fitxer a la zona de treball, canvia el color d'alguns elements de la barra, que indica que es poden fer servir. Seguidament s'indiquen quins són els principals elements de la barra de menú.



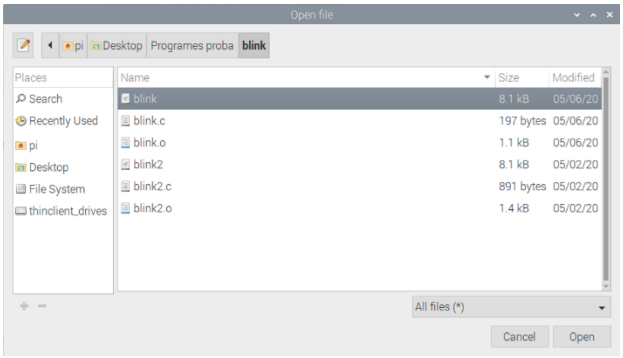
Il·lustració 5. Barra de menú del programa Code::Blocks

Quan es crea un fitxer nou es pot dir de quin tipus és, per fer un programa estàndard s'ha de fer un fitxer buit (Empty file). Un cop creat s'ha de guardar i és molt important que el nom del fitxer s'escrigui l'extensió correcta, és a dir ".c". Si es fa això, el programa reconeix el fitxer com un programa escrit amb llenguatge C per tant canvia el format del text i escull el compilador correcte.



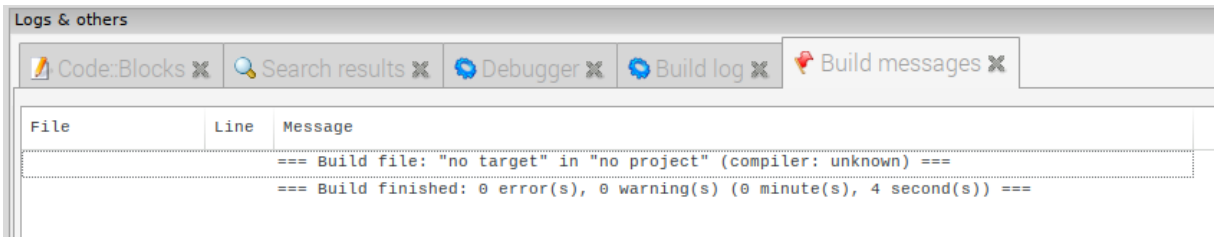
Il·lustració 6. Guardar un fitxer nou.

Per obrir un fitxer nou només s'ha de prémer la segona icona de la barra i a través de la finestra que apareix a la pantalla anar a la carpeta on es troba el fitxer. A l'escriptori hi ha una carpeta amb diversos programes d'exemple per a veure la programació i funcionament dels diferents elements de l'Ardubot. La carpeta es diu Programes proba i al seu interior s'hi troben els exemples.



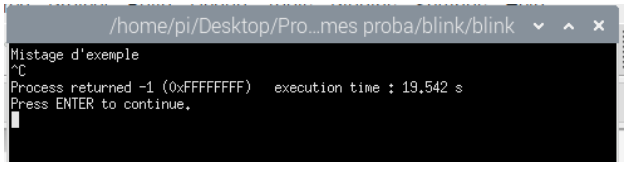
Il·lustració 7. Finestra per obrir un fitxer existent.

Quan es té un programa escrit es pot compilar i executar, però el primer pas sempre és compilar. Per fer aquesta acció s'ha de prémer la icona de l'engranatge i seguidament, a la finestra inferior de la pantalla principal es mostraran els missatges de la compilació. Si no hi ha cap error, es podrà executar el programa.



Il·lustració 8. Missatge de compilador sense cap error.

Quan s'està executant el programa s'obre un terminal on es poden veure missatges del programa. Si s'ha fet un programa que s'executa amb bucle es necessita prémer Control + C per finalitzar el programa.



Il·lustració 9. Terminal que s'obre al executar un programa

En cada apartat d'aquesta pràctica, mirant la taula, s'haurà d'identificar a quin pin del Raspberry correspon l'entrada o sortida corresponent. Es recomana obrir l'exemple anomenat "blink" per veure tots els elements que s'han d'incloure i com s'han de programar.

B.1.4 Experiència 1. LED controlat per polsador

Genera un programa mitjançant el qual es pugui controlar l'estat del LED verd de forma que mentre es premi el polsador s'encengui el LED. El primer pas serà configurar correctament els ports digitals del Raspberry, per controlar el LED necessitarem configurar un port de sortida i per llegir l'estat del polsador necessitarem un port d'entrada.

B.1.5 Experiència 2. LEDs controlats per para-xocs

Genera un programa mitjançant el qual es pugui controlar l'estat del LED groc i del LED vermell de forma que mentre es premi el costat esquerra del para-xocs s'encengui el LED groc i mentre es premi el costat dret del para-xocs s'encengui el LED vermell. El primer pas serà configurar correctament els ports digitals del Raspberry, per controlar els LEDs necessitarem configurar dos ports de sortida i per llegir l'estat del para-xocs necessitarem dos ports d'entrada.

B.1.6 Experiència 3. Semàfor

Genera un programa que simuli el funcionament d'un senzill semàfor mitjançant els tres LEDs de colors. L'encesa i apagada dels LEDs haurà de seguir una seqüència determinada de forma continuada. En la propera taula es mostra quina hauria de ser la seqüència i la duració de cada estat, un cop executat l'últim pas el programa haurà de tornar al primer pas de nou.

Pas	LEDs	Tamps
1	Vermell (L3)	6 segons
2	Vermell + taronja (L2)	1 segon
3	Verd (L1)	4 segons

Per tal de controlar els temps de cada LED pots fer servir la funció delay(ms) que genera una pausa en el programa del temps en milisegons determinat pel paràmetre d'entrada ms.

B.1.7 Experiència 4. Semàfor millorat

El programa generat en l'exercici anterior té un greu inconvenient. Imagina que vols "forçar" que el semàfor es posi en verd en el moment que es prem el polsador i vols que la resposta sigui immediata. Si utilitzes la funció delay el programa queda "parat" dintre aquesta funció de

forma que és impossible detectar si algú ha polsat o no el polsador. Com a norma general sempre és preferible que el bucle principal (la funció while(1)) s'executi de forma continuada sense quedar atrapat en cap funció.

Millora el programa del semàfor de forma que no necessitis fer servir la funció delay. Per controlar el temps es proposa fer servir la funció millis() com a alternativa. Aquesta funció retorna el nombre de milisegons que han transcorregut des que s'ha iniciat l'execució del programa.

B.1.8 Experiència 5. Hello world!

En aquest exercici es veurà el funcionament de la llibreria per la pantalla LCD. Obre el programa d'exemple per LCD de l'entorn de programació:

Open File -> Desktop -> Exemples -> LCD

Llegeix el codi d'aquest exemple i determina quina funció realitza cada línia del codi. En aquest exemple també es mostra com es pot imprimí informació per la línia de comandes que apareix a l'executar el programa.

B.2 Pràctica 2: Mesura analògica

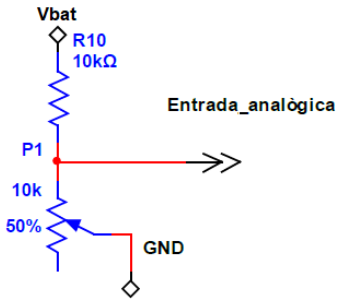
B.2.1 Objectius

Els objectius de la pràctica de Mesures alògiques són treballar amb la lectura analògica i fer operacions matemàtiques.

B.2.2 Introducció

Per tal de monitoritzar el nivell de la bateria s'ha connectat la tensió de la bateria a una entrada analògica del mòdul analògic de la placa adaptadora mitjançant un divisor de tensió. Cal realitzar aquesta connexió amb un divisor de tensió ja que el valor màxim per una entrada analògica és de 5 Vdc i la bateria a plena càrrega pot superar els 9 Vdc.

Donat que els valors de les resistències estan sempre sotmesos a una certa tolerància una de les resistències del divisor és un potenciòmetre per poder fer el calibratge.



Il·lustració 10. Esquema de la connexió per a la lectura de bateria.

Si apliquem la llei d'ohm sobre l'anterior circuit trobem que:

$$V_{bat} = I \cdot (R_{10} + P) \rightarrow I = \frac{V_{bat}}{R + P}$$

A partir d'aquí podem obtenir la tensió que es llegirà en l'entrada analògica i aïllar la tensió de la bateria:

$$V_a = \frac{V_{bat}}{R + P} \cdot P \rightarrow V_{bat} = \frac{V_a \cdot (R + P)}{P}$$

El Raspberry no té de cap entrada analògica i per tant s'ha instal·lat un mòdul amb comunicació I2C per a fer aquesta feina. No obstant cal tenir en compte que la lectura que ens dona el mòdul analògic no són volts sinó un valor enter proporcional. Donat que el convertidor ADC del mòdul analògic és de 8 bits el valor es trobarà entre 0 i 255.

El valor de R el sabem i és fixa de 10 kΩ, en canvi el valor del potenciòmetre no el sabem ja que és variable. Una possible solució seria fixar el valor del potenciòmetre en un valor teòric (per exemple 8kΩ per realitzar els càlculs. Llavors podem fer un programa que realitzi la lectura analògica, en calculi el nivell de bateria corresponent i mostri el resultat per la pantalla. Per realitzar el calibratge necessitarem un multímetre per mesurar el nivell real de la bateria i un tornavís amb el qual ajustarem el potenciòmetre fins que la pantalla mostri el mateix resultat que el multímetre.

Per a programar el mòdul analògic s'ha d'incloure, a part de la llibreria WiringPi, la de l'I2C i la del mòdul analògic. Quan s'inicia el mòdul, se li ha de dir l'adreça d'I2C que té i la base que s'ha escollit per a nombrar les sortides. Aquest mòdul pot tenir diverses adreces i per tant poden existir diferents mòduls i forces números d'entrada i sortides. Per a facilitar la programació s'escull un número base per a cada mòdul i es suma al numero de sortida.

```
#include <stdio.h>
#include <wiringPi.h>
#include <pcf8591.h>
#include <wiringPiI2C.h>

#define PCF 64 // número base del mòdul

int main(void){
    int value0 ;
    int value1 ;
    int value2 ;
    wiringPiSetup () ; //inici de la llibreria wiringPi
    // Setup pcf8591 on base pin 64, and address 0x48
    pcf8591Setup (PCF, 0x48) ;
    while(1) // loop forever
    {
        value0 = analogRead (PCF + 0) ; // entrada 0
        value1 = analogRead (PCF + 1) ; // entrada 1
        value2 = analogRead (PCF + 2) ; // entrada 2
        printf("Pote %i, LDR %i, NTC %i\n", value0, value1, value2);
        analogWrite (PCF + 0, value1) ;
        delay (500) ;
    }
}
```

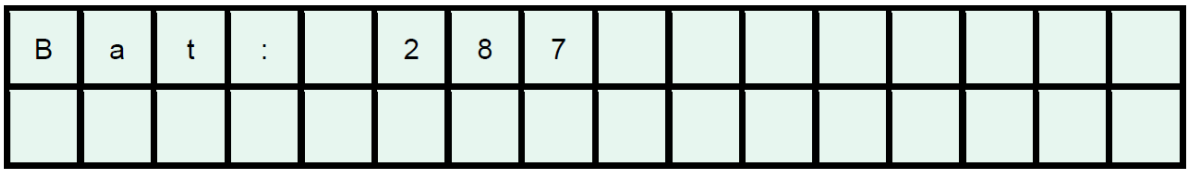
Il·lustració 11. Programa d'exemple del mòdul analògic

Trobareu un exemple a l'escriptori del Raspberry Pi, dins la carpeta dels Programes de proba, a Analògic. Per a informació addicional es pot anat a la pàgina web de la llibreria.

http://wiringpi.com/extensions/i2c-pcf8591/

B.2.3 Experiència 1. Lectura analògica

Realitza la lectura analògica de la tensió de bateria i mostra el resultat per pantalla. Comprova que el valor varia entre 0 i 255 variant el potenciòmetre de calibratge (segons la càrrega de la bateria pot ser que no arribi mai a 255). A la pantalla caldrà visualitzar el resultat d'aquesta forma:

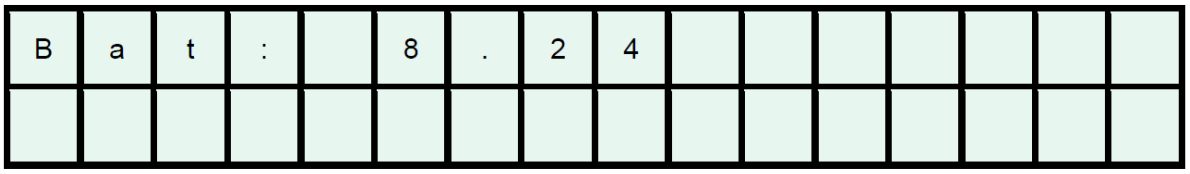


Consell pràctic:

Tingues en compte que segons el valor llegit el resultat correspon a un nombre d'una, dues o tres xifres. Haureu de tenir això en compte quan escriviu per pantalla, del contrari podria passar que es mostri una barreja del valor actual (per exemple de 2 xifres) i del valor anterior (que per exemple podria haver tingut 3 xifres).

B.2.4 Experiència 2. Nivell de bateria

Agafant com a base el codi de l'exercici anterior realitza la lectura del nivell de bateria i fes els càlculs pertinents per convertir la lectura a volts. Mostra el resultat per pantalla i realitza la calibració del potenciòmetre tal com s'explica en l'apartat de teoria. En la pantalla caldrà visualitzar el resultat d'aquesta forma:



Consell pràctic:

Guarda el resultat en una variable del tipus float per mantenir la precisió. Donat que la lectura analògica es retorna en format enter cal convertir la lectura a float abans d'operar amb ella per forçar que les operacions es realitzin en format de coma flotant. Per convertir un enter a float cal fer un "cast":

```
float(valor_enter)
```

B.2.5 Experiència 3. Funció “llegir_bateria”

A partir del codi generat en l’apartat anterior crea una funció que realitzi les operacions de lectura, conversió i escriptura per pantalla del nivell de bateria. La funció ha de tenir dos paràmetres d’entrada que correspondran a la fila i la columna on es vol representar el resultat en la pantalla.

void llegir_bateria(int fila, int columna)

Després fes diferents proves cridant la funció des del bucle principal amb diferents paràmetres i comprova el correcte funcionament.

B.2.6 Experiència 4. Sortida Analògica (Voluntària)

Connecta un LED o un brunzidor (amb el condicionament necessari) a la sortida analògica de la placa adaptadora. Realitza un programa que modifiqui el valor d’aquesta sortida de tal manera que es pugui beure una progressió al llarg de 5 segons. Tingues en compte que la sortida analògica se li pot donar un valor màxima de 255.

B.3 Pràctica 3: Control motors PWM

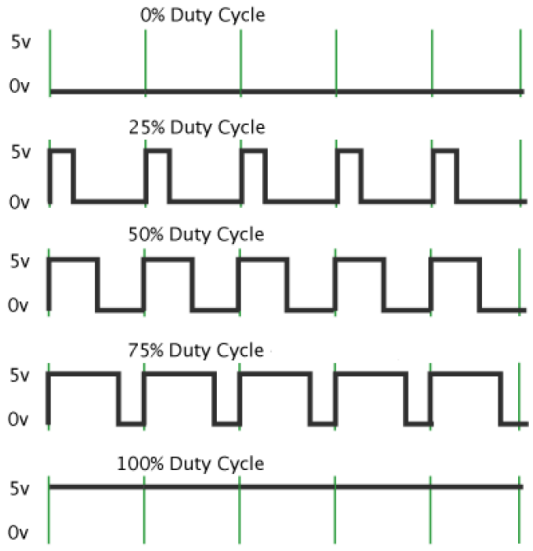
B.3.1 Objectius

Els objectius de la pràctica de Control de motors amb PWM1 són utilitzar una sortida de PWM per hardware, utilitzar una sortida de PWM per software i utilitzar els encoders per mesurar la velocitat.

B.3.2 Introducció

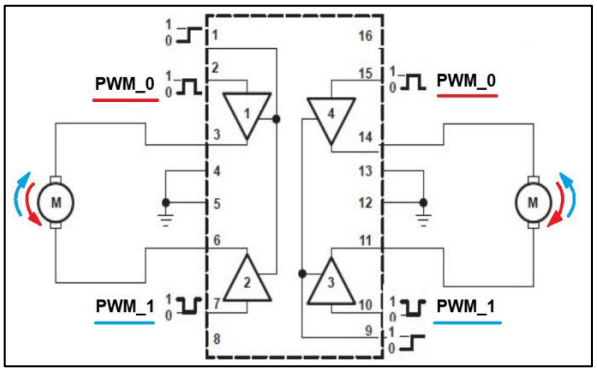
PWM són les sigles en anglès per *Pulse Width Modulation*, és a dir, modulació per ample de pols. Un senyal de PWM és un senyal periòdic (normalment quadrat) del qual podem modificar el cicle de treball (*duty cicle*).

El cicle de treball d'un senyal periòdic és la porció de temps que aquest senyal es troba en estat alt en funció del període. En la figura es poden observar diferents cicles de treball per un senyal PWM d'un determinat període.



Il·lustració 12. Cicles de treball d'un PWM

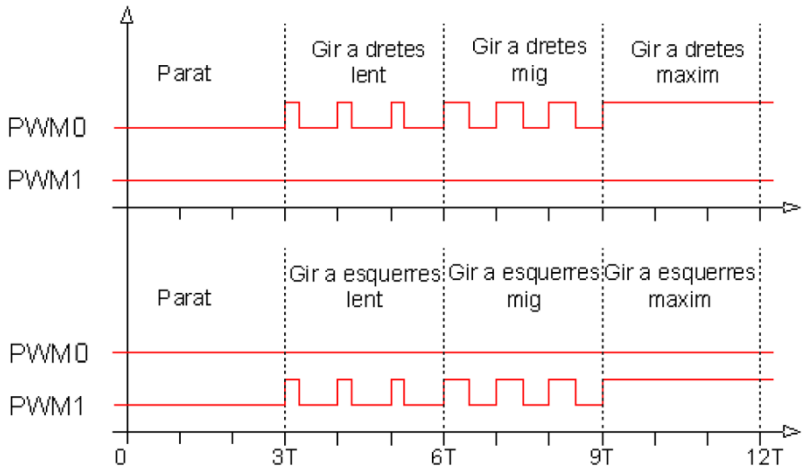
Si alimentem els motors amb un voltatge variable a partir d'un senyal de PWM podrem controlar-ne la velocitat, com més elevat sigui el cicle de treball més elevada serà la velocitat. Si a part de la velocitat es vol controlar el sentit de gir dels motors es necessiten dos senyals de PWM. Segons quina de les sortides es manté a nivell baix i quina porta el senyal de PWM el motor gira en un sentit o altre.



Il·lustració 13. Motor controlat per PWM

Donat que les sortides d'un microcontrolador no poden alimentar directament un motor (recordem que el GPIO del Raspberry pot subministrar un màxim de 16 mA) fa falta un circuit driver per subministrar la potència i tensió necessària als motors.

En la propera figura es pot observar de forma resumida les diferents combinacions de sortides i de cicles de treball que donen com a resultat diferents velocitats i sentits de gir del motor.

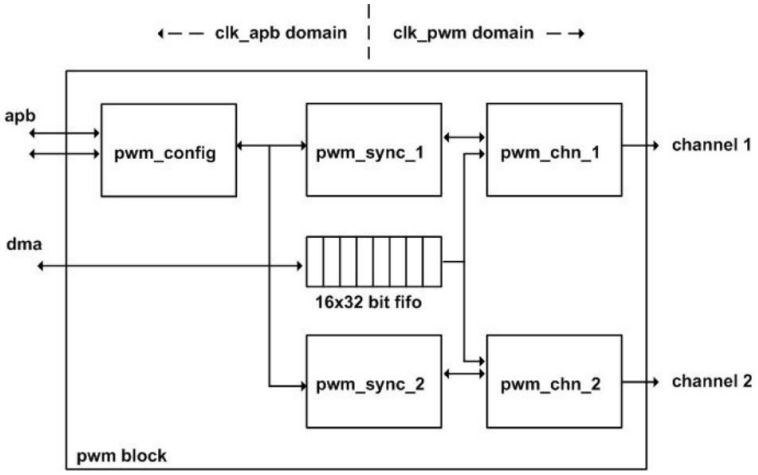


Il·lustració 14. Diferents configuracions de PWM per controlar motors

En la propera taula es detallen els diferents ports del Raspberry que es fan servir per tal de controlar les velocitats i sentits de gir dels dos motors.

Dispositiu	Pin físics / WiringPI del Raspberry
Motor Dret (horari)	33 / 23
Motor Dret (antihorari)	32 / 26
Motor Esquerra (horari)	35 / 24
Motor Esquerra (antihorari)	12 / 1

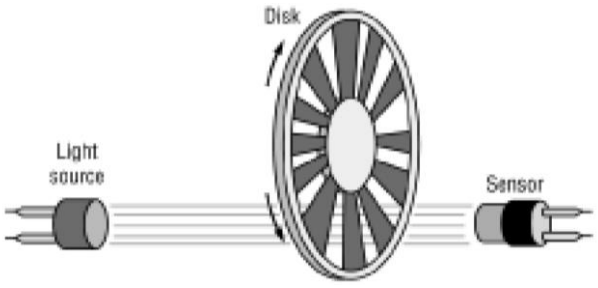
Per a modificar aquesta velocitat es pot fer de dos maneres, a través dels PWM per hardware o software. El Raspberry té dos PWM per hardware, que estan connectats als dos motors, com es mostra a la Il·lustració 12. Aquest té una velocitat de 100 MHz i funciona a través d'un mòdul independent, que no ocupa temps del processador.



Il·lustració 15. Mòdul del PWM per Hardware.

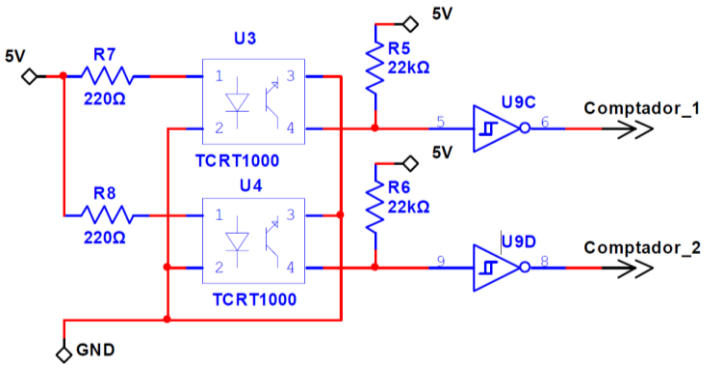
Per altra banda, el PWM per software, és un llibreria creada per a crear un PWM a qualsevol sortida GPIO, la freqüència és més baixa, de 100 Hz. Aquest mètode funciona a través d'interrupcions síncrones de manera que la CPU del Raspberry s'atura en el moment precís, per a crear la sortida correctament. Això ocupa part del temps de processament de la CPU i per això no es pot fer una freqüència més rapida.

Per moltes aplicacions en robòtica interessa saber el desplaçament que ha realitzat i la velocitat a la qual es desplaça. Un encoder òptic ens permet determinar aquestes variables. Un encoder es basa en un sensor òptic de barrera o de reflexió que detecta les marques o forats sobre un disc rotatori. La sortida del sensor un cop digitalitzada correspon a un tren de polsos on cada pols correspon a una marca o un forat del disc.



Il·lustració 16. Principi de funcionament d'un encoder

En el cas de l'Ardubot els encoders estan formats per un sensor òptic amb el mateix principi de funcionament que els sensors del seguidor i una roda foradada, amb franges opaques i transparents. De la mateixa forma que amb els sensors de llum pel seguidor el senyal de l'encoder també cal digitalitzar-lo mitjançant una porta inversora schmitt trigger.



Il·lustració 17. Circuit de l'escoder

Si mitjançant el nostre microcontrolador comptem aquests polsos de l'encoder podem calcular la velocitat de cada roda i el desplaçament que ha fet cada roda.Donat que segons la velocitat del robot els polsos poden ser d'una durada molt curta és convenient realitzar el comptatge mitjançant una interrupció per temps per assegurar-nos que no ens en deixem cap per comptar.

La propera taula detalla les entrades de l'Rasberry que corresponen a cada encoder:

Dispositiu	Pin físics / WiringPI del Raspberry
Encoder Esquerra	29 / 21
Encoder Dret	31 / 22

Moltes aplicacions de microcontroladors requereixen realitzar determinades accions o càlculs amb una base de temps fixa i precisa. En el cas de l'Ardubot un d'aquests casos és, per exemple, el càlcul de la velocitat de desplaçament.

Sabem que la velocitat ve determinada per l'espai recorregut per un determinat temps. L'espai recorregut el podem determinar pels polsos d'encoder, llavors falta saber en quin temps s'ha fet aquest desplaçament. S'hauran de fer servir la funció millis(), tinguent en copte que al cap de 49 dies es satura la memòria.

B.3.3 Experiència 1. Moviment del robot a través de PWM per hardware

Realitza un programa que faci moure el robot a través del PWM per hardware. Per a fer-ho obre l'exemple anomenat "PWM_hard" i utilitza les mateixes funcions. Programa un sol PWM, què passa? Programa l'altre PWM posant el primer a 0, què passa? Ara programa els dos PWM, què passa?

Un cop hagueu guardat el programa s'ha de compilar per el terminal des de la carpeta on es troba el programa. Per moure-us a través de les carpetes per el terminal heu de utilitzar la funció cd i per a visualitzar que hi ha a la carpeta la comanda ls. Si per exemple teniu el programa guardat a la carpeta de l'escriptori anomenada "Practica3" s'hauria d'escriure cd Desktop/Practica3. Per visualitzat què hi ha dins la carpeta només s'hauria d'executar ls. Per tornat al directori anterior s'ha d'escriure cd .. i prémer enter.

Un cop a la carpeta desitjada s'ha d'executar:

```
gcc -o nomprograma nomprograma.c
./nomprograma
```

B.3.4 Experiència 2. Moviment del robot a través de PWM per software

Realitzar un programa que faci moure el robot endavant a una velocitat constant. S'ha d'utilitzat la funció de la llibreria WiringPi, softPwmWrite(), trobaràs l'exemple a la carpeta "PWM_soft". Per a més informació obre http://wiringpi.com/reference/software-pwm-library/. Després crea una nova funció amb dos paràmetres d'entrada de la forma:

```
consigna(int Esquerra, int Dreta)
```

Els paràmetres Esquerra i dreta representen la velocitat de cada una de les rodes expressada com un valor en percentatge de forma que per una consigna de +100% la roda corresponent giri a màxima velocitat endavant i per una consigna de -100% la roda giri a màxima velocitat endarrere.

B.3.5 Experiència 3. Mesurar de la velocitat per polsos

A partir del codi de l'exercici anterior configura les interrupcions externes dels encoders per poder comptar els polsos de cada un d'ells, trobareu un exemple a la carpeta "Interrupcions". Mostra el valor dels comptadors per pantalla segons l'exemple:

B	a	t	:		8	.	2	4						
E	:	4	7		D	:	8	9		p	u	l	s	.

B.3.6 Experiència 4. Càlcul de la velocitat

Configura una interrupció per temps per calcular la velocitat de cada una de les rodes del robot. Es recomana realitzar el càlcul aproximadament cada 0.2 segons per obtenir un resultat prou precís. Realitza els càlculs per obtenir el resultat expressat en cm/s i mostra les velocitats per pantalla tal com es mostra.

B	a	t	:		8	.	2	4						
E	:	1	5	.	0	3		D	:	1	2	.	0	0

Aplica diferents consignes de PWM i comprova com varien les velocitats calculades. Per millorar aquesta aplicació pots afegir un signe a les lectures segons el signe de les consignes aplicades.

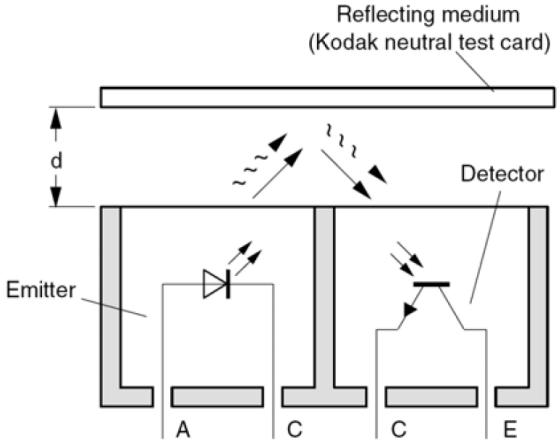
B.4 Pràctica 4: Seguidor d'una línia amb sensors òptics

B.4.1 Objectius

Els objectius de la pràctica del Seguidor d'una línia amb sensors òptics són utilitzar els sensors òptics analògica de manera discretitzada i seguir una trajectòria en forma de línia blanca.

B.4.2 Introducció

L'Ardubot està equipat amb quatre sensors òptics. Els sensors òptics permeten determinar la presència d'objectes reflectants, a l'interior de cada un d'aquests sensors hi ha un díode emissor de llum infraroja i un fototransistor.



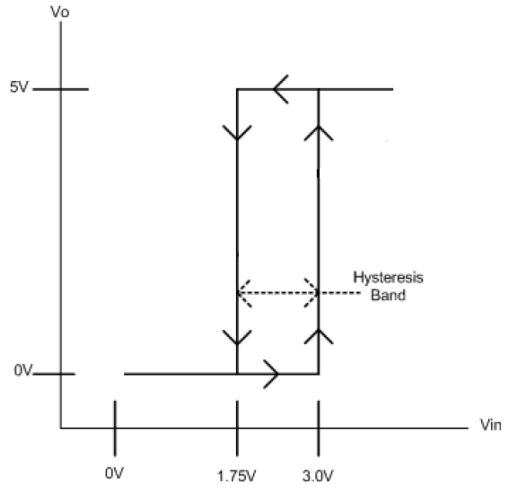
Il·lustració 18. Sensor infraroig CNY70

Si s'alimenta el díode emet llum infraroja, si aquesta llum topa amb un obstacle aquesta pot ser reflectida o no segons el color. Si es tracta d'un objecte de color clar aquest reflectirà la llum de forma que incideix sobre el fototransistor a l'interior del sensor i aquest entra en conducció (equivalent a un "1" digital). Si per contra es tracta d'un objecte fosc la llum serà absorbida per l'objecte de forma que el fototransistor no conduirà corrent (equivalent a un "0" digital).

Donat que en el mon real els objectes no son ni de color blanc ni negre perfecte la lectura d'aquests sensors en realitat no es pot interpretar de forma directa com un "1" o un "0" digital ja que un objecte gris donaria una lectura incerta i erràtica i per tant s'han de digitalitzar. Aquesta acció es realitza mitjançant una porta inversora Schmitt Trigger.

Les portes Smitt Trigger eliminen aquesta zona de incertesa ja que canvien d'estat baix a estat alt quan l'entrada supera un cert llindar màxim (3V en l'exemple de la figura) però no tornen a canviar la sortida a estat baix fins que l'entrada no assoleix un cert llindar mínim (1.75V en l'exemple de la figura). D'aquest efecte se'n diu efecte histèresi i s'anomena zona d'histèresi a aquella zona d'incertesa on la sortida pot valer "1" o "0" depenent dels valors anteriors en el temps de l'entrada.

Les portes Smitt Trigger del robot addicionalment són portes inversores de forma que l'estat de la sortida és invers al de la entrada, d'aquesta forma si llegim un "0" en el corresponent port digital del Raspberry significarà que s'està detectant un objecte de color clar i per contra un objecte fosc donarà una lectura d'un "1" digital.



Il·lustració 19. Efecte histèresi

Per tal de poder observar de forma directa l'estat dels quatre sensors s'han ubicat quatre LEDs que ens informen sobre com és l'objecte detectat per cada sensors. Si el LED està encès (un "1" digital) voldrà dir que l'objecte és de color fosc i pel contrari el LED apagat significarà que l'objecte és de color fosc. En la propera taula es resumeix la relació dels quatre sensors, els quatre LEDs i els ports d'entrada digital del Raspberry. Per tal de identificar de forma visual l'estat de cada un dels sensors s'ha connectat un LED a cada canal que n'indica l'estat.

Sensor	LED	Pin físics / WiringPI del Raspberry
S1 (Lateral esquerra)	L4	37 / 25
S2 (Centre esquerra)	L3	40 / 29
S3 (Centre dret)	L2	38 / 28
S4 (Lateral dret)	L1	36 / 27

B.4.3 Experiència 1. Sniffer

Programa les corresponents consignes de velocitat per tal de seguir una línia marcada sobre el terra. Es recomana programar la solució de tal forma que modificant un paràmetre o una variable a l'inici del codi es pugui escollir si el robot ha de buscar una línia blanca sobre un fons negre o una línia negra sobre un fons blanc.

B.4.4 Experiència 2. Millores al sniffer

A partir del codi de l'apartat anterior incorpora les següents millores.

La primera és, que en cas que el robot perdi la línia de vista fes-lo anar en línia corba fins que la torni a trobar.

La segona és que si el robot ha perdut la línia i xoca amb un obstacle el robot ha de recular un pam, girar cap a un costat aleatori 180º i tornar a anar recte per buscar la línia.

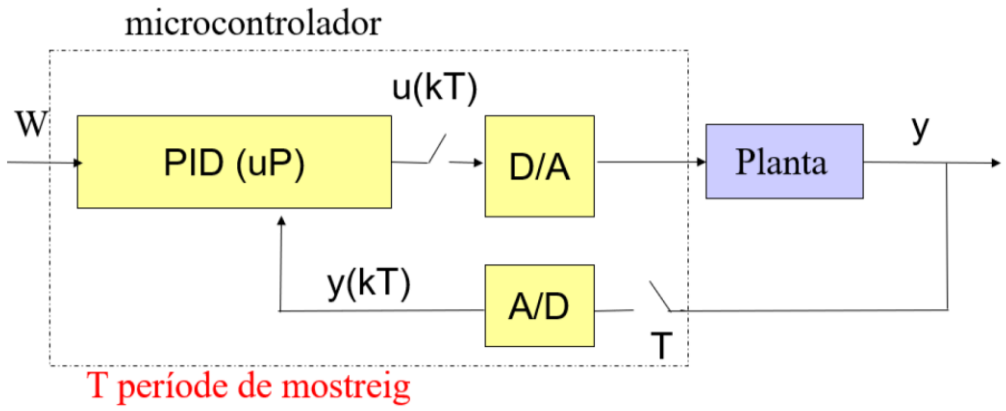
B.5 Pràctica 5: Control amb PID

B.5.1 Objectius

Els objectius de la pràctica Control amb PID són programar un algoritme de control digital directe, un PID digital; obtenir un model aproximat del procés a controlar a partir de la seva resposta a un escaló de tensió i determinar les contents del PID digital a partir de l'estimació del model de procés

B.5.2 Introducció

El control que utilitzarem serà un control discret ja que l'implementarem amb un microcontrolador. La nostra planta serà el conjunt motor-robot i el microcontrolador el Raspberry Pi. L'equació genèrica del control incremental PID (després de discretitzar, aplicant les transformades z corresponents) és la que es mostra a continuació.



Il·lustració 20. Diagrama de blocs d'un control PID discretitzat.

$$m(n) = m(n-1) + K_p[e(n) - e(n-1)] + K_i T_s e(n) + K_d/T_s[e(n) - 2e(n-1) + e(n-2)]$$

On: m() és la senyal de control que s'envia a la planta

e() és l'error entre la dada real i la consigna d'entrada

n és la mostra actual

K_p és la constant de proporcionalitat

K_i és la constant d'integració

K_d és la constant de derivació

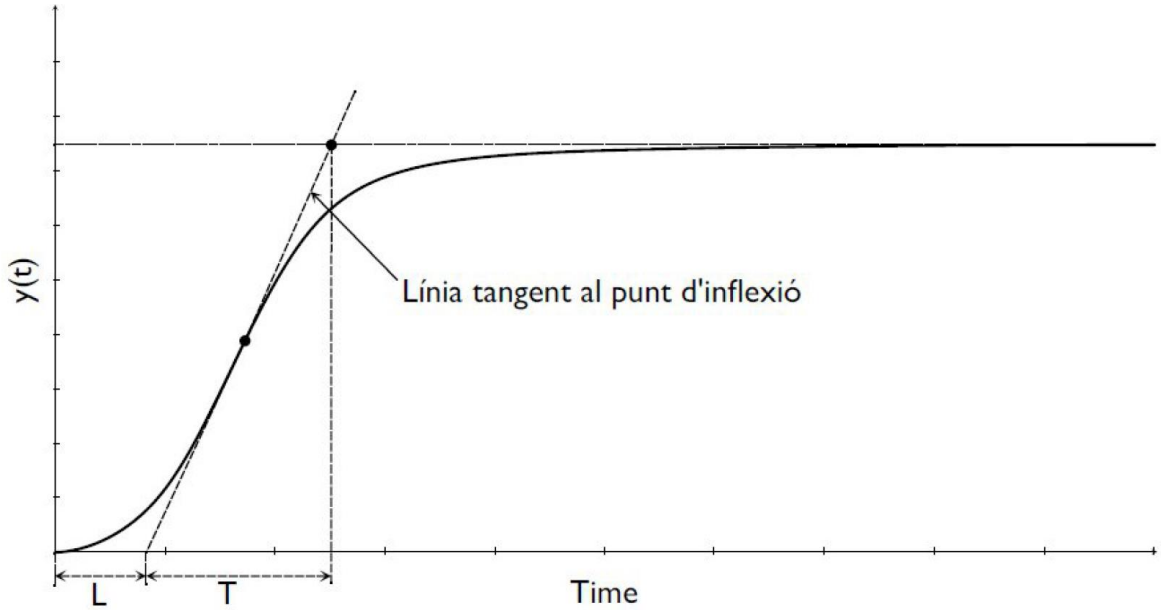
T_s és el temps de mostreig

El control que realitzarem seria unicam PI perquè la derivada de l'error dona problemes. Així doncs l'equació que haurem d'implementar amb el Raspberry serà:

$$m(n) = m(n-1) + K_p[e(n) - e(n-1)] + K_i T_s e(n)$$

Un cop tenim l'equació falta sintonitzar el PID, haurem de triar els valors de Kp, Ki i Ts perquè el control funcioni segons les especificacions de control. Com que no podem conèixer l'equació de la planta (motor-robot) haurem de fer-ho de forma empírica.

Tenim una resposta de primer ordre, utilitzarem el mètode de Ziegler-Nichols utilitzant la resposta de la planta a un graó. Es tracta d'aplicar un graó al sistema i estudiar la seva resposta.

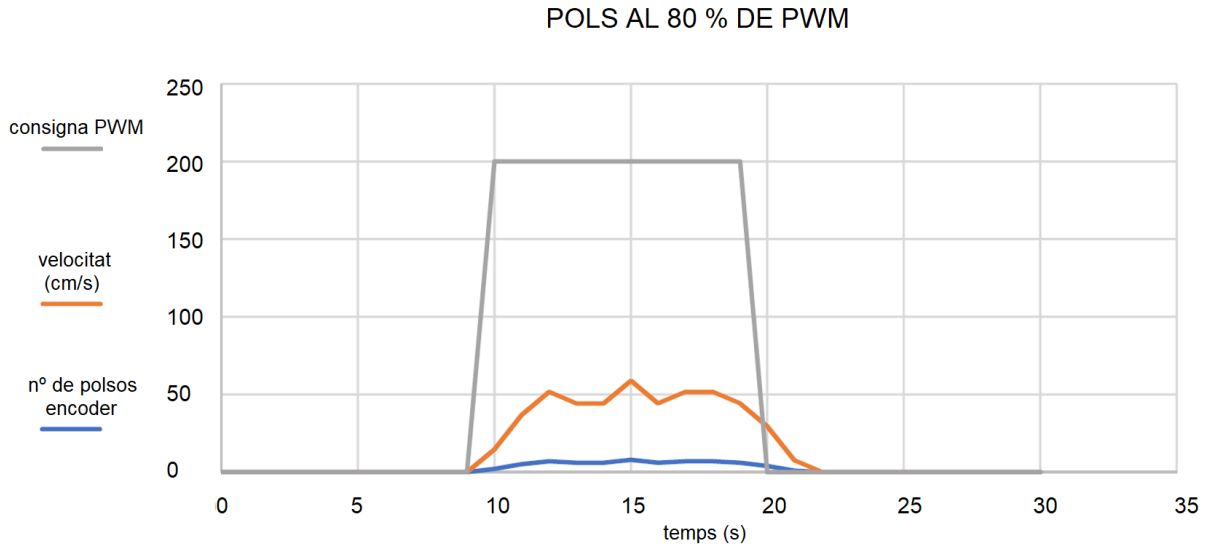


Il·lustració 21. Corba del sistema al aplicar-li un graó per a utilitzar el mètode de Ziegler-Nichols.

Un cop tenim la resposta utilitzant la taula següent obtenim els valors de K_p, τ_i i τ_d.

Controlador	**K_p**	**τ_i**	**τ_d**
P	T/L	Inf	0
PI	0,9T/L	L/0,3	0
PID	1,2T/L	2L	0,5L

Els motors que utilitzem se'ls hi envia una valor entre 0 i 250. Perquè el PID sigui més correcte farem l'estudi Ziegler-Nichols amb la velocitat que utilitzarem a l'hora de fer el control. Aquesta velocitat serà d'uns 200 al PWM que equivalen més o menys a 50cm/s, a la meitat alta del rang de velocitats del robot. Les dades estan agafades cada 100 ms i el pols dura un total de 10 segons.



Il·lustració 22. Gràfica de comportament d'un motor al donar-li una consigna de PWM de 200

Amb la il·lustració anterior obtenim que el valor de L és de 50 ms i que els valor de T és de 300 ms.

$$\boldsymbol{K_p} = 0.9 * \left(\frac{300}{50}\right) = \mathbf{5.4}$$

$$\tau_i = \frac{0.05}{0.3} = 0.16 \rightarrow \boldsymbol{K_i} = \frac{1}{\tau_i} = \frac{1}{0.16} = \mathbf{6.25}$$

El valor del temps de mostreig també és molt important. La teoria diu que hauria de ser entre 5 i 10 vegades més petit que el temps de resposta del sistema. En el nostre cas això no és possible ja que l'encoder òptic que utilitzem per calcular la velocitat no ens pot donar suficients dades. Per aquest motiu hem fet proves i el millor resultat és a Ts=100ms.

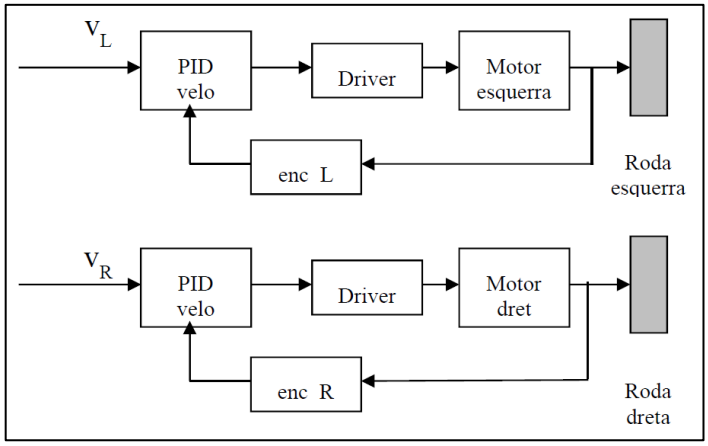
Amb això i el coneixement adquirit a les pràctiques anteriors ja tenim tot el necessari per realitzar el PID.

B.5.3 Experiència 1. Crear PID

Amb les dades donades a la introducció heu de realitzar un control PID, implementat amb el Raspberry, utilitzant la següent equació:

$$m(n) = m(n-1) + K_p[e(n) - e(n-1)] + K_i T_s e(n)$$

Heu d'utilitzar dos controls PID, un per cada roda, però amb els mateixos valors de K_P, K_i i T_s. La velocitat de consigna serà de 40cm/s.



Il·lustració 23. Diagrama de Blocs del PID

B.5.4 Experiència 2. Ajustar el PID

Canvia els paràmetres de K_P, K_i i T_s per a obtenir un PID més ajustat al teu robot, i potser a cada una de les rodes. Llegeix la velocitat a la pantalla LCD i comprova assoleix correctament la consigna amb diferents condicions (sense carrega, pla, pujada, baixada).

El robot va més recte que si li donéssim la mateixa velocitat a les dos rodes sense el PID?

Modifiqueu el T_s, què passa si el fiques més gran? I si el fiques més petit? Perquè creus que passa?

B.6 Pràctica 6: Comunicació Bluetooth

B.6.1 Objectius

Els objectius de la pràctica de Comunicació Bluetooth són conèixer els mòduls sense fils per a comunicació sèrie Bluretooth, conèixer com crear aplicacions Android amb App Inventor 2 i finalment controlar la placa Raspberry Pi 0 W a distància amb el mòbil.

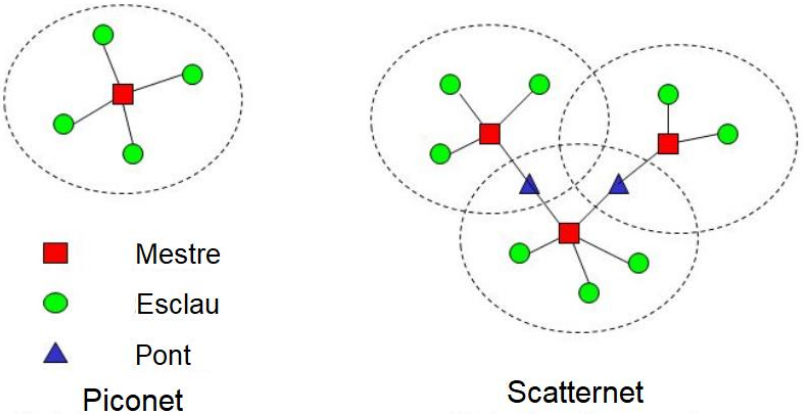
B.6.2 Introducció

La tecnologia Bluetooth és una tecnologia molt popular i estesa entre els dispositius de nova generació d'avui dia. Entre les seves principals característiques, està la seva robustesa, baixa complexitat, baix consum i baix cost. La funció d'aquest mòdul dins el robot, és donar la capacitat de transmetre i rebre informació amb altres dispositius sense la necessitat de cables, és a dir, per un mitjà sense fil.

Més que una tecnologia, Bluetooth és un estàndard per a les comunicacions sense fils de curt abast o Xarxes Sense Fils d'Àrea Personal (WPAN), que possibilita la transmissió de veu i dades entre diferents dispositius mitjançant un enllaç per radiofreqüència en la banda ISM dels 2,4 GHz.

Els principals objectius que es pretenen aconseguir amb aquest estàndard són: facilitar les comunicacions entre equips mòbils; eliminar els cables i connectors entre aquests; oferir la possibilitat de crear petites xarxes sense fils i facilitar la sincronització de dades entre equips personals; obtenir una tecnologia de baix cost i potència que possibilitin dispositius barats.

La topologia de les xarxes Bluetooth pot ser punt a punt o multipunt. Aquesta, es basa en la manera d'operació mestre / esclau on un dispositiu mestre es pot connectar simultàniament amb fins a 7 dispositius esclaus actius amb un abast radial de fins a 10 m o fins i tot més, si s'utilitzen els amplificadors adequats. A la xarxa que forma un dispositiu i els dispositius que es troben dins del seu rang, se li denomina piconet. A més, es pot estendre la xarxa mitjançant la formació de scatternets. Una scatternet és la xarxa produïda quan dos dispositius pertanyents a dos piconets diferents, es connecten. Poden coexistir fins a un màxim de 10 piconets dins d'una sola àrea de cobertura.



Il·lustració 24. Tipus de xarxa Bluetooth.

L'objectiu del mòdul Bluetooth dins del treball és poder establir una connexió amb un Smartphone. Un cop establerta aquesta connexió i amb ajuda d'aplicacions programari dels Smartphone, es pot desplaçar al robot com si d'un radiocontrol es tractés. I també, dóna l'opció de monitoritzar en temps real variables importants del sistema de control.

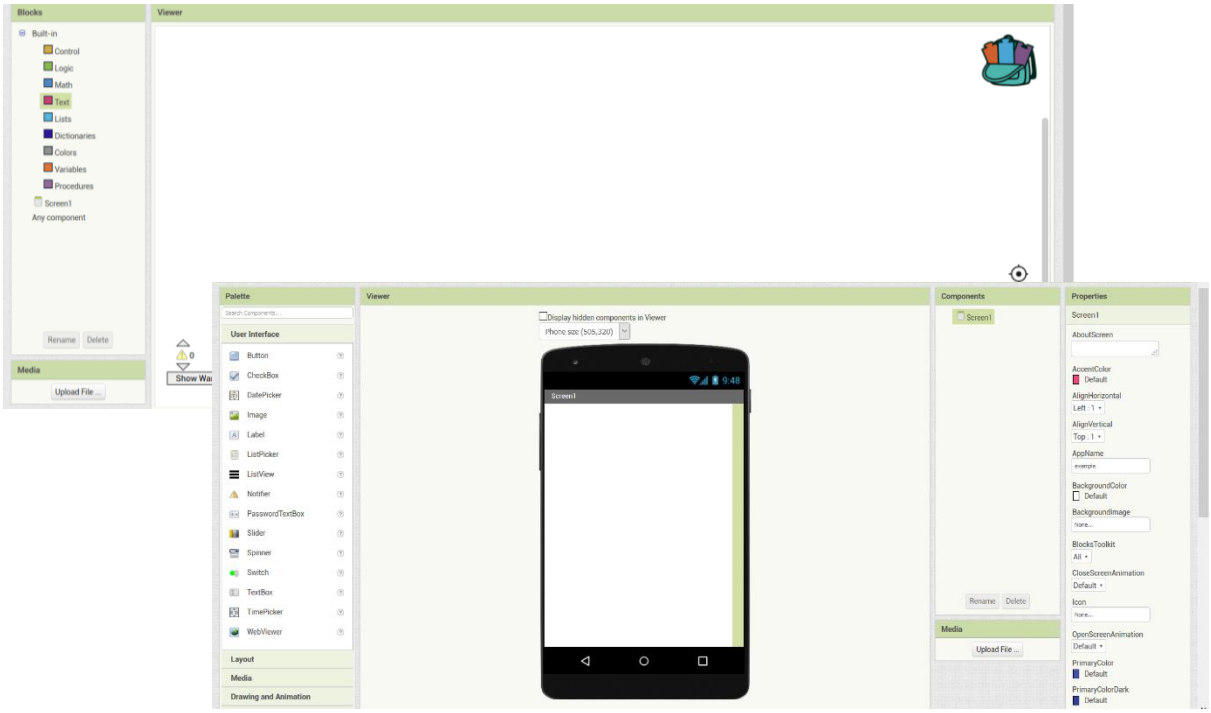
Hi ha molts mòduls Bluetooth per a ser usats en projectes d'electrònica, però els més utilitzats són els mòduls de JI-MCU, a causa que són molt econòmics i fàcils de trobar en el mercat. Són mòduls petits i amb un consum molt baix que permeten afegir funcionalitats Bluetooth amb facilitat a la placa diveses plaques. De totes maneres, el Raspberry Pi 0 W té un Bluetooth incorporat, per tant no s'haurà de ficar cap element extern a la placa per a utilitzar-lo.

Per crear Apps en el mòbil, es pot utilitzar un software dissenyat pel MIT anomenat App Inventor 2. Aquest consta de dues parts o pantalles, una on es fa el disseny gràfic del que serà l'aplicació i en l'altre pantalla és on s'agrupen els blocs per tal de configurar el programa. Així, doncs, en la pantalla de disseny s'incorporen els elements com els botons, quadres de text, imatges i altres components amb els que es pot interactuar i es configura la distribució, la mida i el contingut d'aquests. En l'altre pantalla, la dels blocs, s'agrupen blocs per a tal d'aconseguir que l'aplicació faci el que es vol. Hi ha diferents tipus de blocs.

Els blocs de control es representen de color groc i és on es poden trobar estructures de programació com l'if, el while o obrir i tancar finestres de l'aplicació. Els blocs lògics es representen de color verd i s'hi pot trobar els valors cert i fals. Els blocs matemàtics, de color blau fosc, són aquells destinats a treballar amb valors - numèrics.

Els blocs de color rosa fan referència a text. S'hi pot escriure text, ajuntar-ne dos, comparar textos o separar-ne. Els blocs de color blau cel fan referència a llistes. Es poden crear llistes, afegir o treure un ítem d'una llista i tot tipus d'operacions. Els blocs de color gris són colors, es poden utilitzar per definir un color per un element en concret. Els blocs de les variables es representen de color taronja fosc i permeten crear variables locals o globals i definir el seu valor.

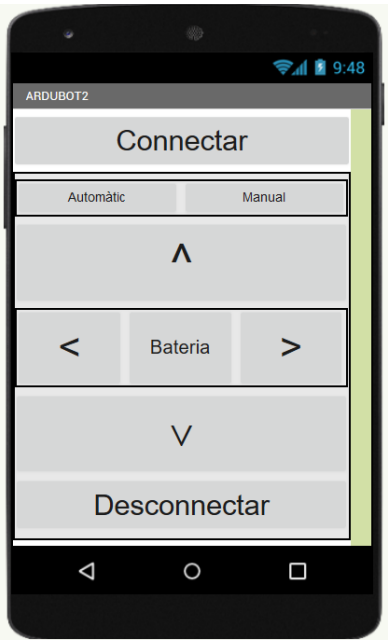
Els blocs de color lila són anomenats blocs de procediments i serveixen per crear funcions i cridar-les. A més de tots aquests, cada element té uns blocs associats. Per exemple, un botó tindrà un bloc que permet canviar el color del text o del fons, un altre que defineix què passa quan es pitja el botó, quan es solta o quan es deixa pitjat.



Il·lustració 25. Les dues pantalles de l'AppInventor.

S'ha escollit App Inventor per molts motius entre ells són la flexibilitat, la facilitat i el cost. Com ja es va esmentar s'executa sobre un navegador, de manera que no cal instal·lar programes addicionals. Per crear una App en Android només cal dissenyar-la, més que programar-la, ja que aquest framework posseeix blocs amb funcions específiques les quals només és necessari assignar-les als components de l'Aplicació. A més a més, App Inventor és una eina gratuïta.

Exemple de pantalla del joystick del mòbil que s'utilitzarà per a comandar l'Ardubot en aquesta pràctica.



Il·lustració 26. Pantalla que es mostra a la pàgina d'AppInventor amb el programa per a moure el robot.

B.6.3 Experiència 1. Connectar-se la Bluetooth

Aparella el teu telèfon amb el Bluetooth i connectat-hi utilitzat la comanda al terminal:

```
sudo rfcomm whatch hci0
```

Obre l'exemple i executa'l, què passa quan es premen els diferents botons de l'aplicació?

B.6.4 Experiència 2. Robot teleoperat, enviar dades cap al robot

En aquesta experiència cada grup haurà d'utilitzar el seu mòbil com a joystick i teleoperar l'Ardubot que disposarà del mòdul bluetooth. En el robot hi haurà carregat un programa per fer moure'l, endavant, endarrere, girar a l'esquerra o girar a la dreta, segons les ordres que se li enviïn. Els codis que fan funcionar aquests robots són: 'A' aturat; 'B' gir a l'esquerra; 'C' gir a la dreta; 'D' recte endavant; 'E' recte enrere.

En el programa del mòbil realitzat amb l'aplicació APPInvetor 2, s'enviaran les comandes descrites anteriorment.

B.6.5 Experiència 3. Monitoritzar dades, rebre dades des del robot

En aquesta experiència sobre l'aplicació d'APPInventor 2 anterior s'haurà d'incloure el rebre dades del nivell de bateria del robot, rebent les dades cada un cert temps utilitzant un rellotge.

Amb aquests dues experiències s'aprendrà a utilitzar el mòbil per rebre i enviar dades a través de Bluetooth.

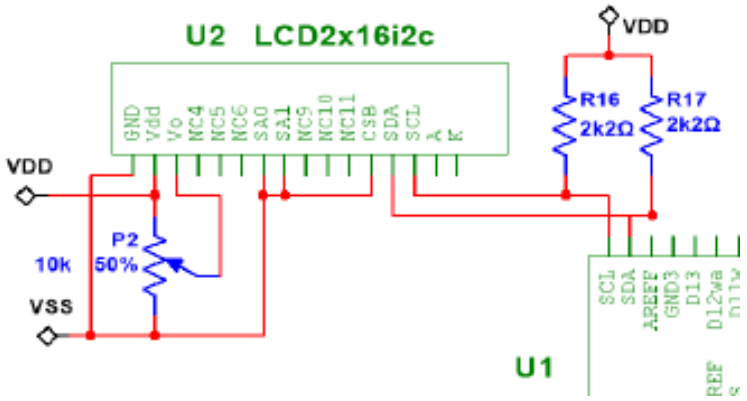
B.7 Pràctica 7: Comunicació I2C

B.7.1 Objectius

Els objectius de la pràctica de Comunicació I2C són utilitzar l'I2C per a comunicar-se amb altres elements utilitzant el Raspberry com a màster, utilitzar element sense llibreria i aprendre a entendre i a modificar codi adquirit.

B.7.2 Introducció

Tant la pantalla LCD com el mòdul analògic funcionen amb aquesta tecnologia, però fins ara s'han fet servir amb una llibreria creada anteriorment. La pantalla, però, s'ha utilitzat amb un útil molt senzill i fàcilment modificable.



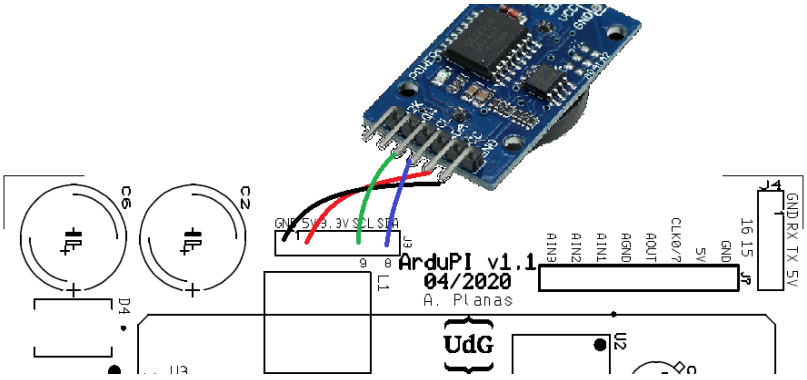
Il·lustració 27. Connexió de la pantalla LCD

Per altra banda, el DS3231 és un rellotge de temps real (RTC) I2C de baix cost, extremadament precís, amb un oscil·lador de cristall (TCXO) amb compensació per temperatura integrada. El dispositiu incorpora una entrada de bateria i manté un comptatge precís quan s'interromp l'alimentació principal del dispositiu.

L'RTC té informació de segons, minuts, hores, dia, mes i any. La data al final del mes s'ajusta automàticament per mesos amb menys de 31 dies, incloses les correccions per a l'any de traspàs. El rellotge funciona en format de 24 hores o de 12 hores amb indicador AM / PM, es pot programar el dia de la setaman. Es tenen dues alarmes programables i una sortida d'ona quadrada programable. L'adreça i les dades es transfereixen en sèrie a través d'un bus bidireccional I2C.

Un circuit de comparació de tensió i un comparador de tensió de precisió monitoritza l'estat de VCC per detectar fallades d'alimentació, proporcionar una sortida de restabliment i canviar automàticament a l'alimentació de còpia de seguretat quan sigui necessari.

Per a connectar aquest element a l'Ardubot s'han d'utilitzar els pins lliures de la part dreta superior, com es mostra a l'esquema següent.



Il·lustració 28. Connexió de l'RTC a la placa adaptadora.

Per a realitzar la segona part de la pràctica no s'utilitzarà cap llibreria, per aquest motiu és molt important descarregar-se el full de característiques de l'RTC DS3231.

Element	Adreça
Rellotge RTC	0x68

B.7.3 Experiència 1. Adreça I2C

Obre un terminal i executa la comanda "i2cdetect -y 1", quines adreces veus? Ara connecta l'RTC i torna a executar la comada, què veus?

B.7.4 Experiència 2. RTC

S'ha creat un codi amb les funcions bàsiques per a poder enviar dades al rellotge, rebre dades del rellotge i llegir-les correctament. A més a més es mostrar l'hora per pantalla. Aquest programa s'anomena RTC i es troba a la carpeta d'exemples, I2C.

Primer de tot s'ha d'entendre el codi adquirit, per fer-ho llegeix atentament el programa i comprova el funcionament que té. Seguidament posa el rellotge a l'hora que toca. Finalment visualitza i programa també la data

Un dels elements interessants de l'RTC és l'alarma, que es pot activar en diferents ocasions. Cada dia a la mateixa hora, cada hora, cada minut, cada segon... Una de les sortides del rellotge és la sortida de l'alarma, que es posa a 1 lògic quan l'alarma s'ha activat.

Per activar l'alarma s'han de modificar diversos registres en un ordre molt concret. Primer es defineixen les adreces de l'alarma 1.

```
#define sega1 0x07
#define mina1 0x08
#define hora1 0x09
#define diaa1 0x0A
```

Seguidament, al *setup(),* s'ha de fer tota la inicialització i programació de l'alarma. Els passos que s'han de fer son: inicialitzar el rellotge, netejar el registre de l'alarma, programar l'alarma perquè s'activi cada segon 5 de cada minut i finalment activar els bits de control de l'alarma1.

Per programar l'alarma s'han de modificar quatre registres. A la pàgina 12 del full de característiques explica com s'han de modificar per a activar-la de cada una de les maneres. Seguidament hi ha un exemple de la inicialització perquè s'alarma s'activi cada segon 5 de cada minut.

```
//inicialització del DS3231
wiringPiI2CWriteReg8(fd, con, 0x04); //activo el INTCN
//activació Alarma
//PRIMER ES NETEJA EL FLAG L'ALARMA
// estatus és una variable que s'ha declarat prèviament //byte estatus = 0;
estatus = wiringPiI2CReadReg8(fd, stat); //s'adquireix valor de tots els registres
estatus = estatus & B11111110 ; //es fa una màscara per modificar únicament l'últim
bit que fa referència a l'estatus del l'alarma 1, i que es posa a 0
wiringPiI2CWriteReg8(fd, stat, estatus); //s'envia el registre
//SEGUIDAMENT ES PROGRAMA L'ALARMA
//a la pàgina 12 del full de característiques ens mostra les màscares de les alarmes
// A1M1 (seconds)   (0 to enable, 1 to disable)
// A1M2 (minutes)   (0 to enable, 1 to disable)
// A1M3 (hour)      (0 to enable, 1 to disable)
// A1M4 (day)       (0 to enable, 1 to disable)
//  posem els registres perquè l'alarma s'activi al segon 5 de cada minut
wiringPiI2CWriteReg8(fd, sega1, 0x05); //B00000101
```

```
wiringPiI2CWriteReg8(fd, mina1, 0x80); //B10000000
wiringPiI2CWriteReg8(fd, hora1, 0x80); //B10000000
wiringPiI2CWriteReg8(fd, diaa1, 0x80); //B10000000
//activem INTC i l'alarma 1 als bits de control
wiringPiI2CWriteReg8(fd, con, 0x5);    //B00000101
```

Per acabar, cada cop que l'alarma s'activa s'ha de tornar a netejar la bandera de l'alarma, perquè es pugui tornar a activar. Per a saber quan s'ha activat l'alarma es pot mirar el registre intern o es pot utilitzar la sortida SQW.

Programar l'alarma com es proposa a l'exemple i mirar el registre de l'alarma per a saber si s'ha activat. (S'ha de mirar l'últim bit del registre 0x0F.)

Mirar el funcionament de l'alarma amb la sortida SQW. Aquest sortida té lògica negada i pot necessitar una resistència pull up.

Modificar la programació de l'alarma per a que s'activi cada 5 regons. Finalment utilitzar aquesta alarma per a fer una lectura analògica.

B.7.5 Experiència 3. LCD

Afegeix una funció a l'útil de la llibreria per escriure un missatge predeterminat. Copia l'útil que es troba a /home/pi/Desktop/Exemples/LCD a la mateixa carpeta i canvia-li el nom per a poder-lo incloure correctament al programa realitzat.

B.8 Pràctica 8: Comunicació amb un servidor local MQTT

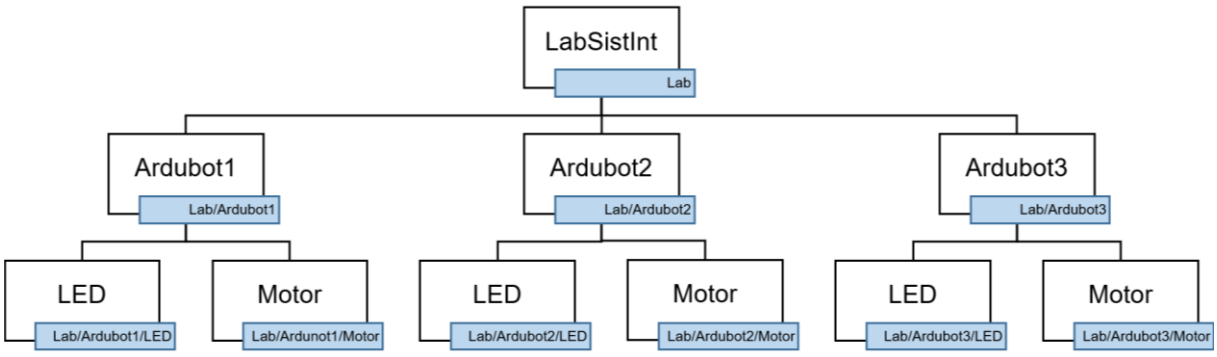
B.8.1 Objectius

Els objectius de la pràctica de Comunicació amb un servidor local MQTT són entendre el funcionament de l'MQTT, publicar a un tòpic de l'MQTT, subscriure's d'un tòpic de l'MQTT i controlar l'Ardubot a través de tòpics.

B.8.2 Introducció

En els últims anys l'internat de les coses ha passat d'un àmbit acadèmic a un desplegament real de milers de dispositius arreu del mot, tant amb caràcter domèstic com en industrial. Un dels grans elements que ha permès l'auge de l'IOT és el protocol MQTT, extremadament lleuger, versàtil i multi-format. Aquest protocol permet enviar tot tipus de missatges amb diversos formats, els més habituals són l'XML, el JSOM i el RAW. També es poden transferir fitxers i imatge. Un dels grans avantatge és la flexibilitat que aporten els tòpics, que són dinàmics i es poden crear per qualsevol moment a mesura que es necessiten.

L'MQTT és un protocol de missatgeria publicador-subscriptor basat en el protocol TCP/IP de codi obert. Es requereix un brokert, que és el que gestiona els missatges. Els clients poden estar subscrits a un tòpic o bé publicar-lo. Els tòpics són rutes estructurades amb forma d'arbre, de forma que si estàs subscrit a un tòpic, de més jerarquia rebràs tots els missatges dels tòpics que es troben per sota.

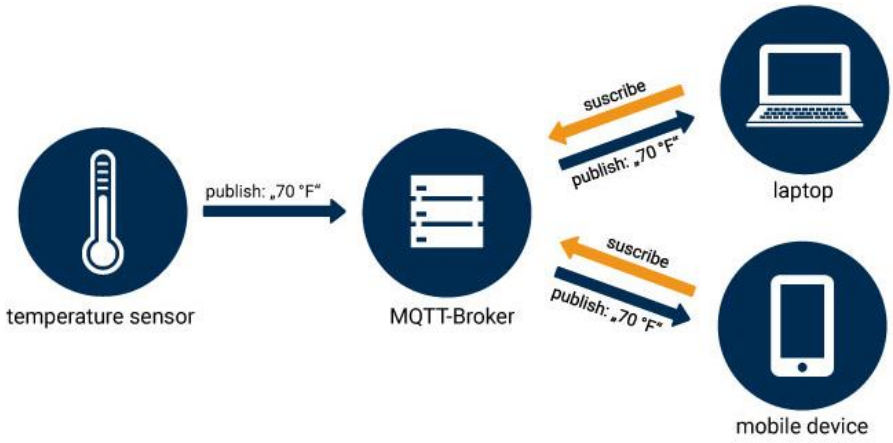


Il·lustració 29. Jerarquia dels tòpics a la pràctica

Per a poder subscriure's a diversos tòpics es pot fer de diferents maneres, per fer-ho existeixen dos operadors, el "+" i el "#". El més s'utilitza per a dir que es volen rebre tots els elements d'una mateixa jerarquia i el coixinet que es volen rebre tots els missatges fins al final

d'una branca. Per exemplificar-ho millor, si es volguessin rebre totes les etiquetes dels LEDs dels diferents robots, s'hauria d'estar subscrit a "Lab/+/LED". Per contra, si volguessis rebre tots els missatges de l'Ardubot3, s'hauria d'estar subscrit a "Lab/Ardubot3/#".

Tots els clients poden estar subscrits i poden publicar a qualsevol tòpic sempre que tinguin connexió al broker. Els ordinadors del laboratori tenen el programa MQTT Explorer instal·lat, de forma que es podrà observar tot el transit de dades de la xarxa i publicar als diferents tòpics des d'ell.



Il·lustració 30. Estructura de la xarxa MQTT

B.8.3 Experiència 1. MQTT a través de terminal

Primer el descarrega el mosquito, que és l'entorn que permet aixecar el servei de Broker MQTT i dels clients. La primera comanda és per instal·lar el necessari pel broker, els clients només necessita la segona comanda. Obre un terminal i executa la segona comanda.

```
sudo apt-get install mosquitto
sudo apt-get install mosquitto-clints
```

Seguidament subscriu el sistema a un tòpic utilitzant:

```
mosquitto_sub -d -h 192.168.1.50 -t "LAB"
```

El -d és el depurador, et dóna la informació de la comunicació (si està connectat, de quin tòpic prové el missatge, etc)

El “-h adreça IP” és necessari per indicar l’adreça del broker, en el laboratori és la IP acabada amb 50.

El -t “tòpic” indica el tòpic on em subscric

Després, s’obre un altre terminal i publica el nom del teu Ardubot al mateix tòpic amb al comanda:

```
mosquitto_pub -h 192.168.1.50 -t “LAB” -m “ArdubotX”
```

El “-h indica de adreça IP” és necessari per indicar l’adreça del broker.

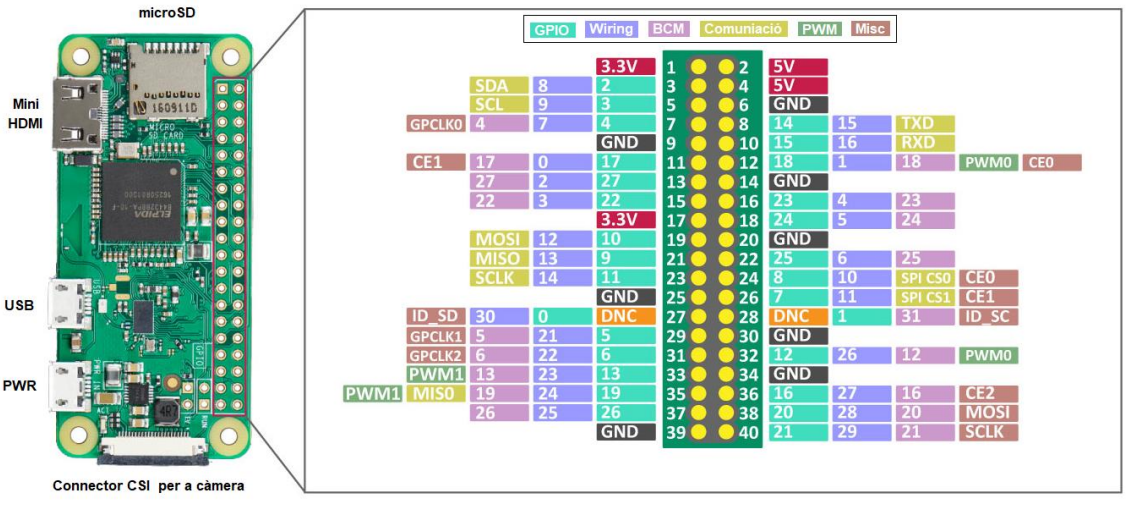
El -t “tòpic” indica el tòpic on publico

El -m “Missatge” indica el missatge

Comprova que el primer terminal ha rebut el missatge. Si és així ha publicat i estàs subscrit correctament al tòpic.

B.8.4 Experiència 2. Comandar un LED a través de l’MQTT per terminal i el programa MQTT Explorer

Primer de tot s’ha d’activar la sortida GPIO del LED per terminal.



Il·lustració 31. Sortides i entrades del Raspberry

Els Diferents LEDs estan connectats als pins físics 11, 13 i 15. Per utilitzar-los a través del terminal, és necessària fer servir la nomenclatura del GPIO, que són 17, 27 i 22. En aquesta pràctica s'activarà el LED vermell que es troba al GPIO 22.

Obre un terminal nou, per activar aquesta sortida cal escriure la comanda:

```
cd /sys/class/gpio/
```

Escriu "ls" en aquesta direcció i t'apareixerà la següent informació:

No hi ha cap instància per poder activar o desactivar cap sortida. Per fer-ho cal escriure la comanda:

```
echo 22 > export
```

Si es torna a executar a comanda ls ha d'haver aparegut el gpio22. Si s'entra amb la comanda cd en aquesta direcció i es torna a executar la comanda ls ens informa de l'estat i les possibles comandes del pin. Així comprovem que està activat correctament. Seguidament s'ha d'assignat la direcció del GPIO, en aquest cas, és una sortida, per tant s'ha d'executar la comanda:

```
echo out > direction
```

Per comprovar si el LED s'encén i s'apaga a través del terminal s'ha d'executar la comanda:

```
echo 1 > value      o      echo 0 > value
```

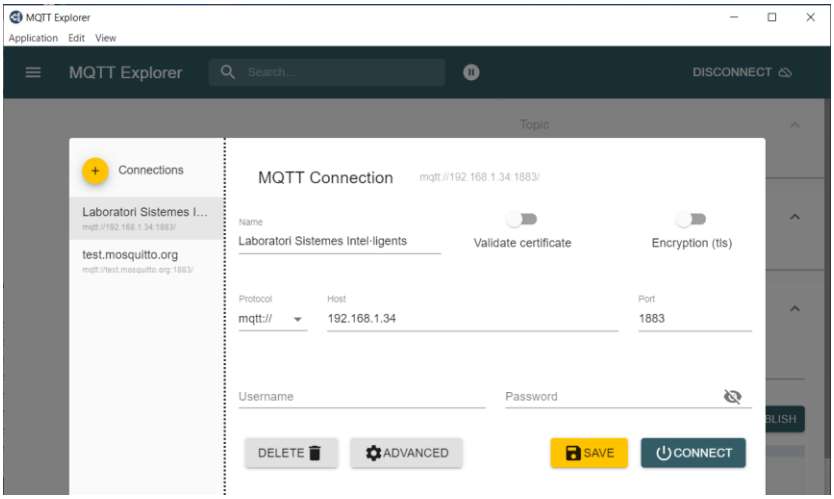
A la il·lustració següent es mostren les diferents comandes que s'han d'utilitzar.

```
pi@raspberrypi:/sys/class/gpio $ echo 22 > export
pi@raspberrypi:/sys/class/gpio $ ls
export  gpio22  gpiochip0  unexport
pi@raspberrypi:/sys/class/gpio $ cd gpio22
pi@raspberrypi:/sys/class/gpio/gpio22 $ ls
active_low  device  direction  edge  power  subsystem  uevent  value
pi@raspberrypi:/sys/class/gpio/gpio22 $ echo out > direction
pi@raspberrypi:/sys/class/gpio/gpio22 $ echo 1 > value
pi@raspberrypi:/sys/class/gpio/gpio22 $ echo 0 > value
pi@raspberrypi:/sys/class/gpio/gpio22 $
```

En aquest punt obriu un altre terminal i subscriviu-vos al tòpic “LAB/ArdubotX”, substituint la X per el numero de teu Ardubot, i enviant el valor del tòpic al GPIO del LED amb la comanda:

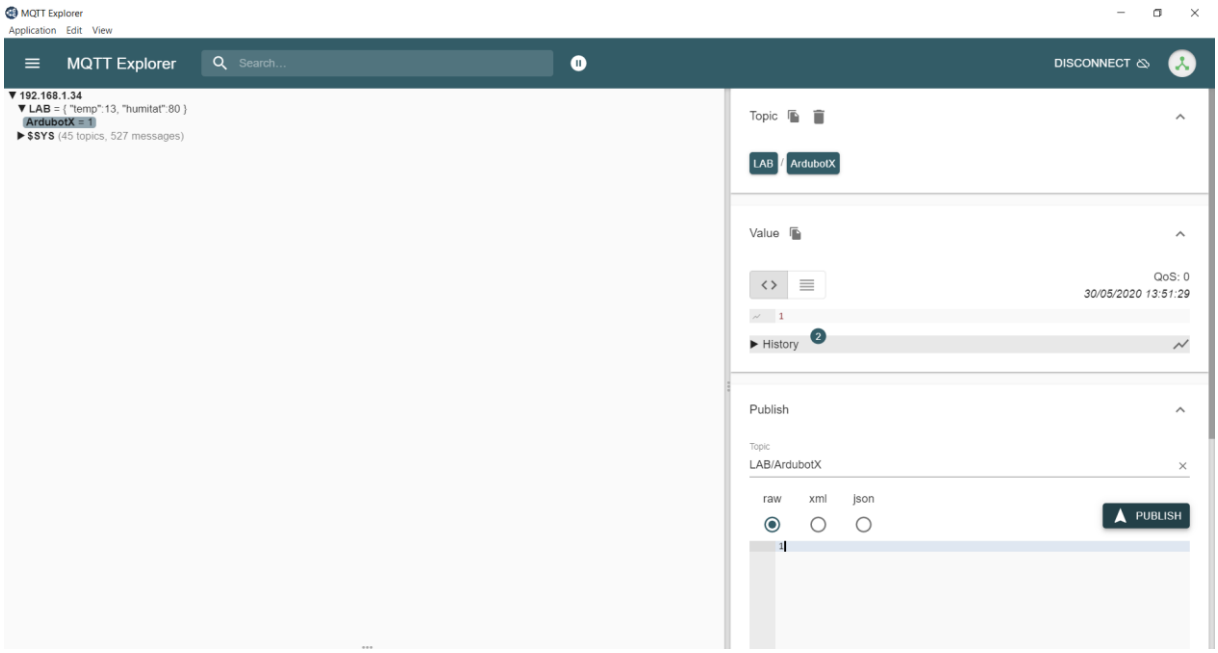
```
mosquitto_sub -d -h 192.168.1.50 -t “LAB/ArdubotX” > /sys/class/gpio/gpio22/value
```

En aquest punt toca publicar al tòpic “LAB/ArdubotX” un 0 o un 1 per encendre o apagar el LED vermell de l’Ardubot. Aquest cop, però, es publicarà a través d’un ordinador del Laboratori. Obriu l’Aplicació MQTT Explorer a l’ordinador del laboratori i obriu la connexió anomenada Laboratori Sistemes Intel·ligents que està configurada de la manera següent.



Il·lustració 32. Configuració de la connexió del programa MQTT Explorer

Al menú de la dreta, s’ha d’escriure el tòpic LAB/ArdubotX, s’ha de seleccionar el format de dades “raw” i a sota ja es pot enviar la informació que es vulgui, en aquest cas, un 1 o un 0. Seguidament només s’ha de prémer Publush per publicar et tòpic i observa si s’encén el LED. Aquest software també permet veure tot els transit de dades del broker a la part esquerra de la pantalla.



Il·lustració 33. Pantalla principal del programa MQTT Explorer

B.8.5 Experiència 3. Jerarquia dels tòpics.

Amb la mateixa estructura que hi ha a l'exemple, a on s'hauria d'estar subscrit si es volgués rebre el missatge del motor de l'Ardubot2? I si es volguessin rebre tots els missatges dels Ardubots. Ara s'afegeix l'estat de la bateria al teu Ardubot. A quin tòpic s'hauria de publicar aquest valor?

B.8.6 Experiència 4. Publicar l'estat del polsador

Obre l'exemple que es troba a la carpeta de l'escriptori Exemples/MQTT/Publicar. Guarda'l amb un altre nom i modifica el programa per a publicar l'estat del polsador al tòpic que has creat anteriorment. Un cop hagis guardat el programa l'has de compilar i executar pel terminal anant al directori on està guardat el programa i executant la comanda:

```
gcc -o nomprograma nomprograma.c -l paho-mptt3c -l wiringPi
./nomprograma
```

B.8.7 Experiència 5. Subscriure's a l'estat del polsador d'un altre robot

Obre l'exemple que es troba a la carpeta de l'escriptori Exemples/MQTT/Subscriure. Guarda'l amb un altre nom i modifica el programa per a subscriure's a l'estat del polsador d'un altre robot. Si no hi ha cap company publicant l'estat de polsador, fes una prova amb el mateix mètode que l'exercici 2. Seguidament, fes que s'encengui el LED verd en funció el tòpic.

Un cop hagis guardat el programa l'has de compilar i executar pel terminal anant al directori on està guardat el programa i executant la comanda:

```
gcc -o nomprograma nomprograma.c -l paho-mptt3c -l wiringPi
./nomprograma
```

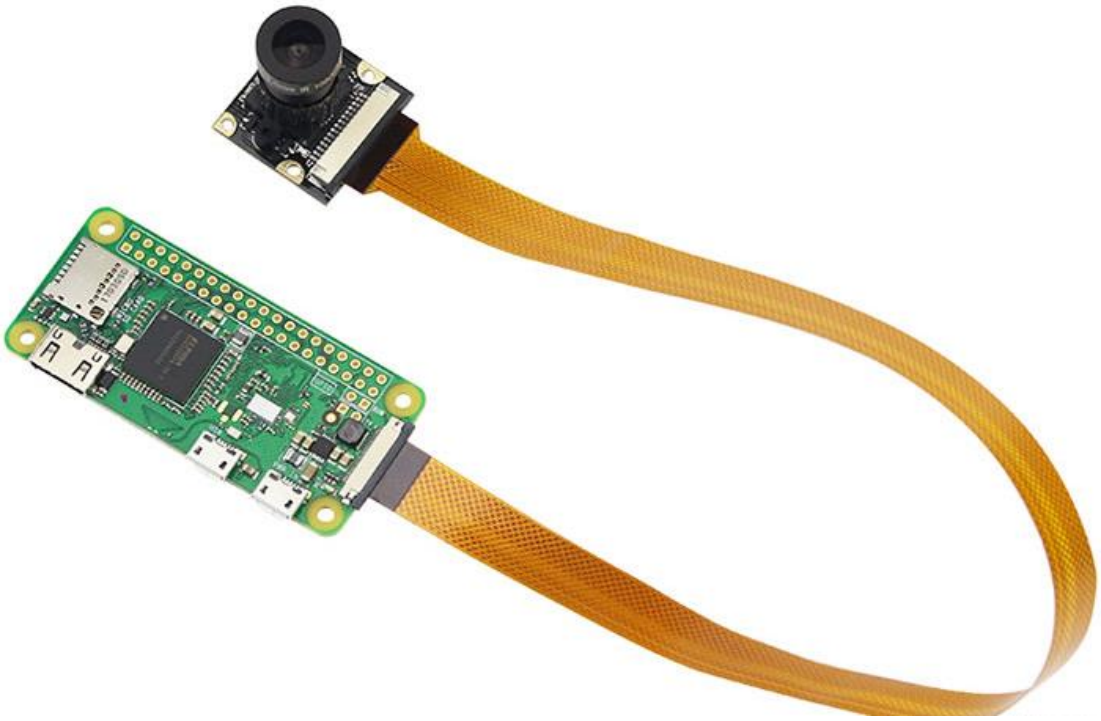
B.9 Pràctica 9: Sistema de videovigilància

B.9.1 Objectius

Els objectius de la pràctica de Sistema de videovigilància són visualitzar la càmera a través del Raspberry Pi, fer una fotografia, fer un vídeo i utilitzar el programa Motion per crear un sistema de videovigilància.

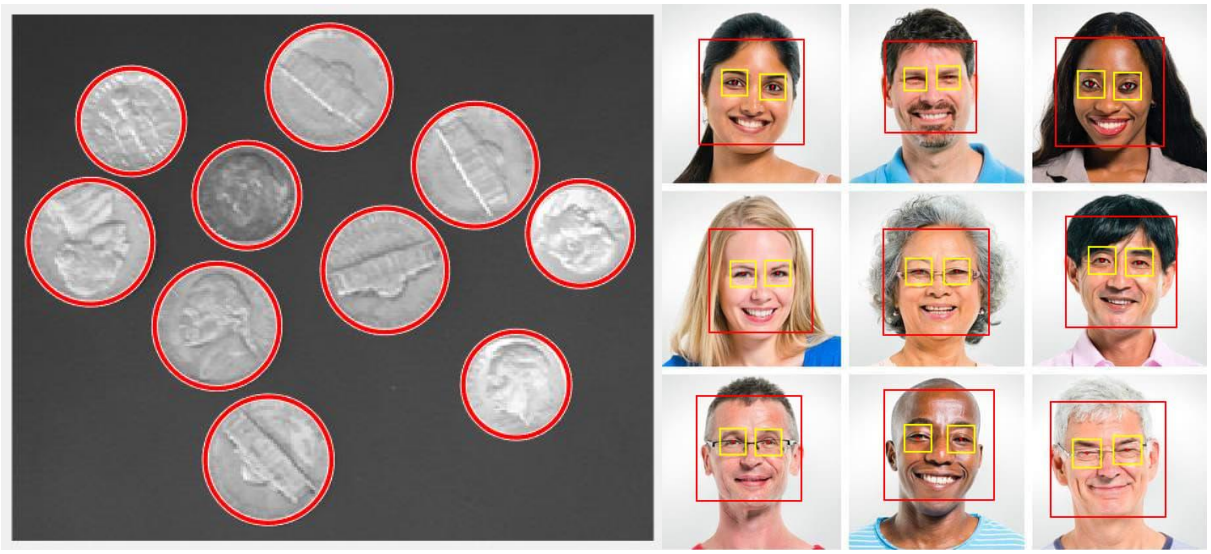
B.9.2 Introducció

El Raspberry Pi 0 W té un connector mini CSI, per a instal·lar-hi una càmera. Es disposa d'una càmera de 640x480 píxels i un màxim de 30 mostres per segon. Es connecta al Raspberry Pi estirant la llengüeta negra del connector. Seguidament s'insereix el cable a l'escletxa que apareix, amb la part metàl·lica mirant cap a la placa. Un cop muntada queda com la il·lustració següent.



Il·lustració 34. Càmera connectada al Raspberry Pi 0 W.

Com que el Raspberry és un ordinador es poden arribar a fer petits programes de tractament d'imatge i visió artificial utilitzant llibreries i programes creats per això. La gran majoria de programes i accessoris utilitzen el llenguatge Python, ja que és molt més senzill per a fer prototipatge. Una de les moltes accions que permet fer és detectar objectes sobre un fons llis o fins i tot, detectar cares i persones en una imatge.



Il·lustració 35. Exemples de visió artificial per a detectar cercles o cares

En aquesta pràctica, però, no es farà, ja que no s'ha introduït el llenguatge Python i requeriria força temps per aprendre'n el suficient. Tot i això una de les llibreries recomanades és la d'OpenCV, de codi lliure i gratuïta. Per contra, es crearà un servidor per a una càmera de videovigilància que emmagatzema el vídeo quan detecta moviment. Per a fer-ho s'utilitzarà el programa Motion.

Il·lustració 36. Logotip del programa utilitzat per a crear la càmera IP amb detecció de moviment.

Motion és un programa de codi lliure totalment gratuït, dissenyat per a poder utilitzat tot tipus de càmera i per a crear un petit servidor de càmera IP. A més a més permet emmagatzemar les imatges que es graven quan es detecta moviment, perfecte per a crear una càmera de videovigilància o per observar nius d'ocell. Té un munt de configuracions que el fan totalment personalitzable. Aquestes configuracions es poden canviar a través del fitxer de configuracions i posteriorment a través d'un cercador i de l'adreça IP i el port assignat anteriorment.

En aquesta pràctica també s'aprendrà a fer una foto i a crear un vídeo des del terminal. A més s'utilitzarà el programa Luvcview per a visualitzar la càmera i les seves característiques principals.

B.9.3 Experiència 1. Fer una foto a través del terminal

Obre el terminal i executa la següent comanda:

```
raspistill -o Desktop/imatge.jpg
```

Què passa? On creus que es desa la imatge? Visualitza-la des de la carpeta on es trobi fent doble clic amb el ratolí.

B.9.4 Experiència 2. Fer una video a través del terminal

Obre el terminal i executa la següent comanda:

```
raspivid -o Desktop/video.h264
```

Què passa? Visualitza el vídeo des de la carpeta on es trobi fent doble clic amb el ratolí i comprova quant de temps a gravat.

Per a modificar el temps predeterminat s'ha d'afegir -t juntament amb el temps. Ves a la pàgina web de Raspberry Pi (https://www.raspberrypi.org/documentation/usage/camera/raspicam/), comprova com s'ha de ser aquest paràmetre i modifica'l per crear un vídeo de 30 segons. Quins altres paràmetres es poden modificar?

B.9.5 Experiència 3. Visualitzar la càmera i les característiques

Obre el terminal i executa la següent comanda:

```
luvcview
```

Què passa? Observa les característiques de la càmera i guarda-les per a poder modificar la configuració del programa Motion. Si la imatge quedat borrosa, pots moure l'enfocament de la càmera perquè es vegi més enfocat.

B.9.6 Experiència 4. Crear un sistema de videovigilància

En aquesta experiència es modificaran els paràmetres de configuració del programa per a poder visualitzar la imatge de la càmera des de qualsevol navegador d'un orinador que es trobi a la mateixa xarxa i que es guardin les imatges quan es detecta moviment.

Obre un terminal i obre la configuració per programa Motion amb la següent comanda:

```
sudo nano /etc/motion/motion.conf
```

Al fer-ho s'obrirà la configuració amb el visualitzador de documents nano. Mou-te pel document amb les fletxes del teclat i modifica els següents paràmetres:

Width → pel valor llegit amb el programa Luvcview

Hight → pel valor llegit amb el programa Luvcview

Output_pictures → best

Locate_motion_mode → on

Stream_localhost → off

Webcontrol_port → 8080

Webcontrol_localhost → off

Webcontrol_parms → 2

Observa la descripció d'aquests paràmetres per a descobrir què estàs canviant. Busca també la capeta on es guardaran les imatges creades (paràmetres target_dir) i el port de la projecció en directe (tream_port).

Un cop modificat prem control X, guarda'l amb el mateix nom i executa el programa amb la comanda:

```
sudo motion start
```

Obre un navegador del teu ordinador i escriu la IP del teu robot juntament amb el port de l'emissió de les imatges en directe, per exemple el 192.168.1.10:5052. Comprova que visualitzes la imatge correctament. Si no hi ha moviment, mou la mà per davant de la imatge per crear moviment.

Vés a la carpeta on es guarden les imatges i busca si ha detectat el moviment. Que ha aparegut en aquesta carpeta? Perquè creus que no es guarden vídeos sinó imatges?

Obre un altre navegador i de la mateixa manera entra les configuracions del sistema. Canvia busca el paràmetre que enquadra el moviment per a convertir-lo amb una creu.

Tenca el programa prement control C al terminal del Raspberry pi. Per a més informació sobre aquest programa pots anar a la pàgina Motion (https://motion-project.github.io/index.html).

C PROGRAMARI

C.1 Test Unitari Entrada Digital

```
//PROGRAMA: Entrada Digital
//es mostra per pantalla si s'activen les diferents entrades digitals
#include <wiringPi.h>
#include <stdio.h>
//Numero dels pins amb WitingPi
#define Polsador 0
#define ParaXocsD 4
#define ParaXocsE 5
int main (void){
    wiringPiSetup(); //Inicial llibreria Wiring Pi
    pinMode(Polsador, INPUT);  //Declarar que son entrades
    pinMode(ParaXocsD, INPUT);
    pinMode(ParaXocsE, INPUT);
    while(1){
             if (digitalRead(Polsador)){ //Llegir el polsador
                   printf("Polsador \n");
                   delay (500);
             }
             if (digitalRead(ParaXocsD)){
                   printf("ParaXocsD \n");
                   delay (500);
             }
             if (digitalRead(ParaXocsE)){
                   printf("ParaXocsE \n");
                   delay (500);
             }
    }
    return 0;
}
```

C.2 Test Unitari Sortides Digitals

```
//PROGRAMA: Sortides Digitals
//S'encenen els 3 LEDs mig segon, un darrera l'altre
#include <wiringPi.h>
#define LED1 0
#define LED2 2
#define LED3 3
int main (void){
    wiringPiSetup();
    pinMode(LED1, OUTPUT);
    pinMode(LED2, OUTPUT);
```

```
    pinMode(LED3, OUTPUT);
    while(1){
        digitalWrite(LED1, HIGH);
        digitalWrite(LED2, LOW);
        digitalWrite(LED3, LOW);
        delay (500);
        digitalWrite(LED1, LOW);
        digitalWrite(LED2, HIGH);
        digitalWrite(LED3, LOW);
        delay (500);
        digitalWrite(LED1, LOW);
        digitalWrite(LED2, LOW);
        digitalWrite(LED3, HIGH);
        delay (500);
      }
      return 0;
}
```

C.3 Exemple Blink

```
//EXEMPLE: Blink
//S'encén i s'apaga el LED verd de l'Ardubot cada mig segon
#include <wiringPi.h>
#include <stdio.h>

//Definir els pins de WiringPi dels elements
#define LED1 0
void  main (){
      wiringPiSetup(); //inicialitzar la llibreria WirnigPi
      pinMode(LED1,OUTPUT); //si fons una entrada INPUT
      delay (50);
      while (1){
            digitalWrite(LED1, HIGH);
            printf("LED ON");
            delay (500);
            digitalWrite(LED1, LOW);
            delay (500);
      }
}
```

C.4 Test Unitari Lectura de la Bateria

```
//PROGRAMA: Lectura de la Bateria
//Es mostra per pantalla el valor d'ADC de la bateria
#include <stdio.h>
```

```
#include <wiringPi.h>
#include <pcf8591.h>
#include <wiringPiI2C.h>

#define Bateria 0 // pin de la bateria
#define PCF 64 // número base del mòdul

int main(void){
     int bat;
     wiringPiSetup () ; //inici de la llibreria wiringPi
     // Setup pcf8591 on base pin 64, and address 0x48
     pcf8591Setup (PCF, 0x48) ;
     while(1) // loop forever
     {
        bat = analogRead (PCF + 0) ; // entrada 0
        printf("ADC de la Bateria és: %i", bat);
        delay (500) ;
     }
}
```

C.5 Exemple Lectura i Escriptura Analògica

```
//EXEMPLE ENTRADA ANALÒGICA
//En aquest exemple es llegeixen les entrades 0, 1 i 2 del modul i s'escriu el valor
de 0 a al sortida analogica
#include <stdio.h>
#include <wiringPi.h>
#include <pcf8591.h>
#include <wiringPiI2C.h>

#define PCF 64 // número base del mòdul

int main(void){
     int value0 ;
     int value1 ;
     int value2 ;
     wiringPiSetup () ; //inici de la llibreria wiringPi
     // Setup pcf8591 on base pin 64, and address 0x48
     pcf8591Setup (PCF, 0x48) ;
     while(1) // loop forever
     {
         value0 = analogRead (PCF + 0) ; // entrada 0
         value1 = analogRead (PCF + 1) ; // entrada 1
         value2 = analogRead (PCF + 2) ; // entrada 2
```

```
         printf("Pote %i, LDR %i, NTC %i\n", value0, value1, value2);
         analogWrite (PCF + 0, value0) ;
         delay (500) ;
     }
}
```

C.6 Test Unitari Moviment de Motors

```
//PROGRAMA MOTORS
//en aquest programa mou els motors 1 segons cap a cada sentit amb un PWM per software
#include <wiringPi.h>
#include <softPwm.h>

#define rodaDdav 23
#define rodaDenr 26
#define rodaEdav 24
#define rodaEenr 1

int main (void){
       int llum ;
    wiringPiSetup();
       softPwmCreate(rodaDdav, 0, 255);
       softPwmCreate(rodaDenr, 0, 255);
       softPwmCreate(rodaEdav, 0, 255);
       softPwmCreate(rodaEenr, 0, 255);

    while(1){
        softPwmWrite(rodaDdav, 200);
             softPwmWrite(rodaEdav, 200);
             softPwmWrite(rodaDenr, 0);
             softPwmWrite(rodaEenr, 0);
             delay(1000);
             softPwmWrite(rodaDdav, 0);
             softPwmWrite(rodaEdav, 0);
             softPwmWrite(rodaDenr, 200);
             softPwmWrite(rodaEenr, 200);
       }
       return 0;
}
```

C.7 Exemple PWM per Hardware

```
//EXEMPLE:   PWM PER HARDWARE
//Aquest programa activa un dels motors per hardware
//Un cop guardat s'ha de compilar i executar el programaper terminal utilitzant les
següents comandes al directori corresponent:
```

```
//gcc -o nomprograma nomprograma.c
//./nomprograma

#include <wiringPi.h>
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>

#define MotorD1 23

int main (void){
  int vel ;
  printf ("Progrma d'exemple de PWM per hardware\n") ;
  wiringPiSetup ();
  pinMode(MotorD1, PWM_OUTPUT); // configuració del pin

  while (1){
    for (vel = 0 ; vel < 1024 ; ++vel){
         pwmWrite (MotorD1, vel) ;
        delay (1) ;
    }
    for (vel = 1023 ; vel >= 0 ; --vel){
         pwmWrite (MotorD1, vel) ;
        delay (1) ;
    }
  }
  return 0 ;
}
```

C.8 Exemple PWM per Software

```
//EXEMPLE: PWM per Software
//en aquest programa es crea un PWM per software per modificar la lluminositat d'un
LED
#include <wiringPi.h>
#include <softPwm.h>

#define LED1 0
int main (void){
    int llum ;
    wiringPiSetup();
    softPwmCreate(LED1, 0,100); // (Sortida, valor inicial, valor màxim)

    while(1){
```

```
        for (llum = 0 ; llum < 100 ; ++llum){
               softPwmWrite (LED1, llum) ;
               delay (1) ;
              }

              for (llum = 100 ; llum >= 0 ; --llum){
               pwmWrite (LED1, llum) ;
               delay (1) ;
              }
       }
       return 0;
}
```

C.9 Test Unitari Encoders

```
//PROGRAMA: encoders
//Es llegeixen els canvis de flangs dels encoders i es mostra el comptador per pantalla
#include <stdio.h>
#include <string.h>
#include <errno.h>
#include <stdlib.h>
#include <wiringPi.h>

// WiringPi dels encoders
#define encoderEs 21
#define encoderDr 22

int ComptEs = 0 ;
int ComptDr = 0 ;

//Interrupcionc
void EnDre (void){
  ComptEs++ ;
}
void EnEsq (void){
  ComptDr++ ;
}

// main()
int main (void)
{

  wiringPiSetup ();
```

```
  //inicialitzar la interrupció amb: (nom del PIN, INT_EDGE_FALLING, INT_EDGE_RISING,
INT_EDGE_BOTH, nom interrupcio)
  wiringPiISR (encoderDr, INT_EDGE_BOTH, &ComptDr);
  wiringPiISR (encoderEs, INT_EDGE_BOTH, &ComptEs);

  while (1){
    printf ("Encoder Dret: %d  Encoder Esquerra: %d\n", ComptDr, ComptEs) ;
  }
  return 0 ;
}
```

C.10 Exemple Interrupcions Asíncrones

```
//EXEMPLE: INTERRUPCIONS ASÍNCRONES
//Si es prem el polsador es crea un interrupció que suma 1 al comptador i es mostra
per pantalla la suma total
#include <stdio.h>
#include <string.h>
#include <errno.h>
#include <stdlib.h>
#include <wiringPi.h>

// WiringPi del polsador
#define      Polsador     6

int globalCounter = 0 ;

//Interrupció
void myInterrupt (void){
  ++globalCounter ;
}

// main()
int main (void)
{
  int myCounter = 0 ;
  wiringPiSetup ();

  //inicialitzar la interrupció amb: (nom del PIN, INT_EDGE_FALLING, INT_EDGE_RISING,
INT_EDGE_BOTH, nom interrupcio)
  wiringPiISR (Polsador, INT_EDGE_BOTH, &myInterrupt);

  while (1){
    printf ("Waiting ... ") ;
```

```
    fflush (stdout) ; //obre de flux de dades del terminal a l'espera d'un caràcter

    while (myCounter == globalCounter){
    delay (100) ;
       }
    printf (" Counter: %5d\n", globalCounter) ;
    myCounter = globalCounter ;
  }
  return 0 ;
}
```

C.11 Exemple Bluetooth

```
//EXEMPLE: US DEL BT COM A RECEPTOR
//amb el programa creat amb l'APP inventor mostra per pantalla les comandes dels diferents polsadors
//primer s'ha de connectar a través del treminal amb la comanda: sudo rfcomm watch hci0.
//Despreś es pot fer serir normalment.

#include <stdio.h>
#include <string.h>
#include <errno.h>
#include <wiringPi.h>
#include <wiringSerial.h>

int main(void){
    int fd;
    int i = 0;
    int estatSerial;
    int Baud = 115200;
    char charin;

    if ((fd = serialOpen ("/dev/rfcomm0", Baud)) < 0)
      {
        fprintf (stderr, "Unable to open serial device: %s\n", strerror (errno)) ;
        return 1 ;
      }
      printf(" -> %i ",fd);
    while (1){
        while (serialDataAvail (fd))
        {
            charin = serialGetchar (fd);
```

```
            printf (" -> %i ", charin) ;
            delay(10);
            if (charin == 'D') printf("Davant \r\n");
            if (charin == 'C') printf("Dreta \r\n");
            if (charin == 'B') printf("Esquerra \r\n");
            if (charin == 'E') printf("Enrrera \r\n");
            if (charin == 'K') { // per tencar s'ha de tencar la comunicació
                serialClose(fd);
                delay(500);
                fd = serialOpen ("/dev/rfcomm0", Baud);
                while (fd == -1){ //a l'espera de nova connexió
                    fd = serialOpen ("/dev/rfcomm0", Baud);
                    printf(" -> %i ",fd);
                    delay(100);
                    fflush (stdout) ;
                }
            }
            //if (charin == 'F') serialClose(fd);
            fflush (stdout) ;
        }
    }
}
```

C.12 Prova unitària Pantalla LCD

```
//PROGRAMA: LCD amb l'útil
//Es crea una vonversa per comporvar que l'útil creat funciona correctamet.

#include <stdio.h>
#include <stdlib.h>
#include "Lcd_I2C_Ardubot.h"
#include <wiringPi.h>

//Veriables globals
int SW = 0;
int contador = 0;

int main(void){
    //variables locals
    int fd;
    char fila[8];
    //inicialització programa
    fd=LCD_InitI2C();
       sendLcdInit(fd);
```

```
	wiringPiSetup();

	sleep(2); //espera de 2 segons per poder lleguir el missatge de comprovació de
la inicialització
	sendLcdCursorLocate(fd,0,1); //mosar el cursor a la segona fila
	sendLcdString(fd,"Hola a tothom!");
	sleep(2);
	sendLcdClear(fd, 0); //neteja la primera fila
	sendLcdCursorLocate(fd,0,0);
	sendLcdString(fd,"Com va?");
	sleep(2);
	sendLcdClear(fd, 1); //neteja la segona fila
	sendLcdCursorLocate(fd,0,1);
	sendLcdString(fd,"Molt be, ADEU!");
	sleep(2);
	sendLcdClear(fd, 2); //neteja tota la pantalla
}
```

C.13 Exemple Pantalla LCD

```
//Exemple: Us de la pantalla LCD amb l'útil
//En aquest programa s'utilitza la pantalla LCD per a mostrar l'estat del polsador i
un compotador intern
//També es mostra l'us de la funcio ptintf() per mostrar informació per pantalla

#include <stdio.h>
#include <stdlib.h>
#include "Lcd_I2C_Ardubot.h"
#include <wiringPi.h>

#define pulsador 6

//Veriables globals
int SW = 0;
int contador = 0;

int main(void){
    //variables locals
    int fd;
    char fila[8];
    //inicialització programa
    fd=LCD_InitI2C();
    sendLcdInit(fd);
    wiringPiSetup();
```

```
    pinMode(pulsador, INPUT);

    sleep(2); //espera de 2 segons per poder lleguir el missatge de comprovació de la
inicialització
    sendLcdClear(fd, 2); //neteja de tota la pantalla
    sendLcdCursorLocate(fd,0,0);
    sendLcdString(fd,"Pulsador: | cont:");

    //bucle iteratiu
    while(1){
        SW = digitalRead(pulsador);
        printf("L'estat del pulsador és: %u\n", SW);
        if(SW){
            sendLcdCursorLocate(fd,0,1);
            sendLcdString(fd,"Apretat");
        }
        else{
            sendLcdCursorLocate(fd,0,1);
            sendLcdString(fd,"Relaxat");
        }

        sprintf(fila, "Seg:%u", contador);
        printf(" El sistema porta encès (%s) \n", fila);
        sendLcdCursorLocate(fd,8,1);
        sendLcdString(fd,fila);
        contador++;

        putchar('\n'); //Per fer un 2n salt de linia
        sleep(1);
    }
    sendLcdClear(fd, 2);
    LCD_CloseI2C(fd);
}
```

C.14 Útil per a la pantalla LCD I2C

```
/*
  Name: Lcd_I2C_Ardubot. Utseful to linux's i2c lcd control sample
  EPS-UDG Laboratory of Sistemes Inteligents
  TFG Robot i pràctiques per apendre a progremar sistemes encastats
  Coded by Anna Planas Bahí, La Bisbal d'Empordà,  Fab 2020
  ver:2.1
*/
```

```
#include <stdio.h>
#include <stdlib.h>
#include <wiringPi.h>
#include <wiringPiI2C.h>
#include <errno.h>

#define I2C_SLAVEADDR_LCD 0x3c

int LCD_InitI2C(void){
//Funció per inicialitzar el I2C de la pantalla
    //veriables locals
    int fd;
    //detecció del dipositiu i si el troba retorna el fitxe del linux assignat per
gestionar-lo
    fd = wiringPiI2CSetup(I2C_SLAVEADDR_LCD);
    if(fd < 0){
        printf("Esclau i2c no identificat\n");
        exit(1);
    }
    return fd;
}

void sendLcdInit(int fd){
//funció per inicialitzar la pantalla
    //Arrays de configuració de la pantalla
    unsigned char init_1[] = { 0x3e,0x39,0x14,0x75,0x5e,0x6c };
    unsigned char init_2[] = { 0x0c,0x01,0x06 };
    unsigned char buff[2];
    int i;
    int result;

    //crida de Unix per iniciar la comunicació amb un dispositiu identificat(fd)
    //Fd ref del fitxer a modificar
    //0x0703 Solicitud per accedir el driver o dipositiu
    //I2CAddr tipu de dades que que s'ha d'emegatzemar, en aquest cas int.
    ioctl(fd,0x0703,I2C_SLAVEADDR_LCD);

    //començo a editar el fitxer amb els arrays de configuració
    buff[0] = 0;
    for(i=0;i<sizeof(init_1);i++){
        buff[1] = init_1[i];
        result = write(fd,buff,2);
        usleep(30);
    }
```

```
    //espera 20ms perque es configuru la pantalla LCD
    usleep(200000);

    //2na configuració de la pantalla
    for(i=0;i<sizeof(init_2);i++){
        buff[1] = init_2[i];
        result = write(fd,buff,2);
        usleep(30);
    }

    //espera perque la pugui configurar
    usleep(1000);

    //Netejo tota la pantalla un cop iniciada
    sendLcdClear(fd, 2);

    //Es comprava la comunicació i s'escriu a la pantalla
    sendLcdString(fd," LCD I2C ArduPI");
}

void sendLcdString(int fd,char* str){
//Funció per printar un string a pantalla
    //Veriables locals
    int i;
    unsigned char buff[2];
    int result;

    //Accedir ens els fitxers per iniciar la comunciació amb dipositius
    ioctl(fd,0x0703,I2C_SLAVEADDR_LCD);

    //Budiar el array d'entrada que ens ha entrat en foramt punter
    buff[0] = 0x40;
    while(*str){
        buff[1] = *str;
        result = write(fd,buff,2);
        str++;
    }
}

void sendLcdCursorLocate(int fd,int row,int col){
//Funció per el posicionament del cursor
    //veriables locals
    unsigned char buff[2];
```

```
    int result;
    //accés al fitxe de comunicació amb drivers de Unix
    ioctl(fd,0x0703,I2C_SLAVEADDR_LCD);

    //Mascares pel moviment de les columnes
    buff[0] = 0x00;
    buff[1] = 0x80;

    if(col != 0){
        buff[1] += 0x40;
    }
    buff[1] += row;

    result = write(fd,buff,2);
}

void sendLcdClear(int fd, int row){
//Funció per netejar la pantalla
    //veriables locals
    unsigned char buff[2];
    int result;
    int i;

    //Accés als fitxers de Unix per comendar dipositius
    ioctl(fd,0x0703,I2C_SLAVEADDR_LCD);

    //mascara
    buff[0] = 0x40;

    //Desplaço per tots els cacters i s'escriu un espai
    if (row != 2){
        sendLcdCursorLocate(fd,0,row);
        for (i = 0 ; i<=16 ; i++){
            buff[1] = ' ';
            result = write(fd,buff,2);
        }
    }
    else {
        sendLcdCursorLocate(fd,0,0);
        for (i = 0 ; i<=16 ; i++){
            buff[1] = ' ';
            result = write(fd,buff,2);
        }
        sendLcdCursorLocate(fd,0,1);
```

```
        for (i = 0 ; i<=16 ; i++){
            buff[1] = ' ';
            result = write(fd,buff,2);
        }
    }
  sendLcdCursorLocate(fd,0,0);
}

void LCD_CloseI2C(int fd){
    close(fd);
}
```

C.15 Exemple RTC

```
//Exemple: RTC
//Programa per connectar-le a l'RTC DS3231 i llegir els segons
#include <wiringPi.h>
#include <stdio.h>
#include <wiringPiI2C.h>
#include <errno.h>

//Adreces del rellotge
#define seg 0x00
#define mii 0x01
#define hor 0x02
#define dset 0x03
#define dia 0x04
#define mes 0x05
#define any 0x06
#define con 0x0E
#define stat 0x0F
#define offset 0x10
#define temp1 0x11
#define temp2 0x12

#define adr 0x68

int main (void){
    wiringPiSetup();
    int fd = 0;
    int C = 0;
    int ini = 0;
    int numerror;
    int con = 0x0e;
```

```
    int adr = 0x68;

    int bcd_dec(int bcd);

    fd = wiringPiI2CSetup(adr); //inicial l'I2C
    printf("iniciat I2C %i\r\n", fd);

    ini = wiringPiI2CWriteReg8(fd, con, 0x04); //Iniciar l'RTC
    numerror = errno; //detecció d'errors
    printf("inicialitzat %i %i\r\n", ini, numerror);
    perror("error ini");
    delay(100);

    while(1) {
      C=wiringPiI2CReadReg8(fd, seg); //es llegeix al registre "seg" del model amb in
identficador "fd"
      C=bcd_dec(C); //es passa de bcd a decimal
      errno;
      printf("segon %i %i\r\n", C,errno); //s'imprimerix
      delay(1000);
    }
}

int bcd_dec(int bcd){
  int dec = 0;
  int temp = 0;
  temp = bcd % 16;
  bcd=bcd>>4;
  dec=bcd*10;
  dec=dec+temp;
  return dec;
}
```

C.16 Exemple MQTT Publicador

```
/*******************************************************************************
 * Copyright (c) 2012, 2013 IBM Corp.
 *
 * All rights reserved. This program and the accompanying materials
 * are made available under the terms of the Eclipse Public License v1.0
 * and Eclipse Distribution License v1.0 which accompany this distribution.
 *
 * The Eclipse Public License is available at
 *    http://www.eclipse.org/legal/epl-v10.html
```

```
 * and the Eclipse Distribution License is available at
 *   http://www.eclipse.org/org/documents/edl-v10.php.
 *
 * Contributors:
 *    Ian Craggs - initial contribution
 *******************************************************************************/

#include "stdio.h"
#include "stdlib.h"
#include "string.h"
#include "time.h"
//#include "ncurses.h"
#include "MQTTClient.h"
#include "wiringPi.h"

#define ADDRESS     "192.168.1.50:1883"
#define CLIENTID    "ArdubotX"
#define TOPIC1      "LAB/ArdubotX/seg"
#define TOPIC2      "LAB/ArdubotX/SW"
//#define PAYLOAD     "Hello World!"
#define QOS         0
#define TIMEOUT     10000L

#define LED3 3
#define pulsador 6

int var = 0;
char str[20];
int SW = 0;

volatile MQTTClient_deliveryToken deliveredtoken;

void delivered(void *context, MQTTClient_deliveryToken dt){
        printf("Message with token value %d delivery confirmed\n", dt);
        deliveredtoken = dt;
}

int msgarrvd(void *context, char *topicName, int topicLen, MQTTClient_message
*message){
        int i;
        char* payloadptr;

        printf("Message arrived\n");
        printf("     topic: %s\n", topicName);
```

```
        printf("   message: ");

        payloadptr = message->payload;
        for(i=0; i<message->payloadlen; i++){
                putchar(*payloadptr++);
        }
        putchar('\n');
        MQTTClient_freeMessage(&message);
        MQTTClient_free(topicName);
        return 1;
}

void connlost(void *context, char *cause){
        printf("\nConnection lost\n");
        printf("     cause: %s\n", cause);
}

//implematicó timeout

int main(int argc, char* argv[]){
//verialbes locals
int ch;
    //iniclaització amb el broker mqtt i connecció
       MQTTClient client;
       MQTTClient_connectOptions conn_opts = MQTTClient_connectOptions_initializer;
       MQTTClient_message pubmsg = MQTTClient_message_initializer;
       MQTTClient_deliveryToken token;
       int rc;
       //realitzar connexió amb el Broker MQTT
    MQTTClient_create(&client, ADDRESS, CLIENTID,
              MQTTCLIENT_PERSISTENCE_NONE, NULL);
    conn_opts.keepAliveInterval = 20;
    conn_opts.cleansession = 1;
    MQTTClient_setCallbacks(client, NULL, connlost, msgarrvd, delivered);
    if ((rc = MQTTClient_connect(client, &conn_opts)) != MQTTCLIENT_SUCCESS){
         printf("Failed to connect, return code %d\n", rc);
         exit(-1);
    }

    //inicialitzacióñ del codi
    wiringPiSetup();
    pinMode(LED3, OUTPUT);
    pinMode(pulsador, INPUT);
       while (1){
```

```
    //rutin adquisició
    SW=digitalRead(pulsador);
    digitalWrite(LED3, SW);

    //rutina comunicació mqtt
        sprintf(str, "%u", var);
        pubmsg.payload = str;
        pubmsg.payloadlen = strlen(str);
        pubmsg.qos = QOS;
        pubmsg.retained = 0;
        deliveredtoken = 0;
        MQTTClient_publishMessage(client, TOPIC1, &pubmsg, &token);
        printf("Waiting for publication of %s\n"
                    "on topic %s for client with ClientID: %s\n",
                    str, TOPIC1, CLIENTID);

        sprintf(str, "%u",SW);
        pubmsg.payload = str;
        pubmsg.payloadlen = strlen(str);
        pubmsg.qos = QOS;
        pubmsg.retained = 0;
        deliveredtoken = 0;
        MQTTClient_publishMessage(client, TOPIC2, &pubmsg, &token);
        printf("Waiting for publication of %s\n"
                    "on topic %s for client with ClientID: %s\n",
                    str, TOPIC2, CLIENTID);
        //while(deliveredtoken != token);

         var++;
         delay(1000);

       }
       //finalitzar connexió MQTT
       MQTTClient_disconnect(client, 10000);
     MQTTClient_destroy(&client);
       return rc;
}
```

C.17 Exemple MQTT Subscriptor

```
/*******************************************************************************
 * Copyright (c) 2012, 2013 IBM Corp.
 *
 * All rights reserved. This program and the accompanying materials
```

```
 * are made available under the terms of the Eclipse Public License v1.0
 * and Eclipse Distribution License v1.0 which accompany this distribution.
 *
 * The Eclipse Public License is available at
 *   http://www.eclipse.org/legal/epl-v10.html
 * and the Eclipse Distribution License is available at
 *   http://www.eclipse.org/org/documents/edl-v10.php.
 *
 * Contributors:
 *    Ian Craggs - initial contribution
 *******************************************************************************/

#include "stdio.h"
#include "stdlib.h"
#include "string.h"
#include "MQTTClient.h"
#include "wiringPi.h"

//definicions MQTT
#define ADDRESS     "192.168.1.38:1883"
#define CLIENTID    "ArdubotX"
#define TOPIC       "LAB/ArdubotX/#"
#define PAYLOAD     "Hello World!"
#define QOS         0
#define TIMEOUT     10000L

//definicions robot
#define LED1 0
#define LED2 2
#define LED3 3

volatile MQTTClient_deliveryToken deliveredtoken;

void delivered(void *context, MQTTClient_deliveryToken dt){
       printf("Message with token value %d delivery confirmed\n", dt);
       deliveredtoken = dt;
}

int msgarrvd(void *context, char *topicName, int topicLen, MQTTClient_message
*message){
    // veriables MQTT
      int i;
      char* payloadptr;
      char temp;
```

```
        //verialbes robot
        char numero[5];
        int num = 0;

        //processament MQTT
    printf("Message arrived\n");
        printf("     topic: %s\n", topicName);
        printf("   message: ");
    payloadptr = message->payload;
    for(i=0; i<message->payloadlen; i++){
         temp=*payloadptr++;
         putchar(temp);
         numero[i]=temp;
    }
        putchar('\n');
        MQTTClient_freeMessage(&message);
        MQTTClient_free(topicName);

         //processament tòpics i valors d'entrada
         num = atoi(numero);
         if (strstr(topicName,"LED1")){
         printf("LED1: %u\n", num);
         digitalWrite(LED1, num);
         }
         else if (strstr(topicName,"LED2")){
         printf("LED2: %u\n", num);
         digitalWrite(LED2, num);
         }
         else if (strstr(topicName,"LED3")){
         printf("LED3: %u\n", num);
         digitalWrite(LED3, num);
         }
         else{
         printf("topic no reconegut\n");
         }

         return 1;
}

void connlost(void *context, char *cause){
        printf("\nConnection lost\n");
        printf("     cause: %s\n", cause);
}
```

```
int main(int argc, char* argv[]){
    //inicialització i connexió amb el broker MQTT
       MQTTClient client;
       MQTTClient_connectOptions conn_opts = MQTTClient_connectOptions_initializer;
       int rc;
       int ch;

       MQTTClient_create(&client, ADDRESS, CLIENTID,
               MQTTCLIENT_PERSISTENCE_NONE, NULL);
       conn_opts.keepAliveInterval = 20;
       conn_opts.cleansession = 1;

    MQTTClient_setCallbacks(client, NULL, connlost, msgarrvd, delivered);
       if ((rc = MQTTClient_connect(client, &conn_opts)) != MQTTCLIENT_SUCCESS){
        printf("Failed to connect, return code %d\n", rc);
        exit(-1);
       }
       printf("Subscribing to topic %s\nfor client %s using QoS%d\n\n"
                      "Press Q<Enter> to quit\n\n", TOPIC, CLIENTID, QOS);
       MQTTClient_subscribe(client, TOPIC, QOS);

    //inicialització robot
    wiringPiSetup();
    pinMode(LED1, OUTPUT);
    pinMode(LED2, OUTPUT);
    pinMode(LED3, OUTPUT);

    //programa iteratiu (while(1))
        do{
               ch = getchar();
        } while(ch!='Q' && ch != 'q');

    //desconneció del MQTT
        MQTTClient_disconnect(client, 10000);
        MQTTClient_destroy(&client);
        return rc;
}
```

C.18 Test Funcional

```
//TEST FUNCIONAL DE L'ARDUBOT4.1
//Aquest programa activa tites les sirtodes i utilitza totes les entrades per a
comprovar el seu funcionament
```

```
#include <stdio.h>
#include <stdlib.h>
#include <wiringPi.h>
#include <wiringPiI2C.h>
#include <errno.h>
#include <softPwm.h>
#include <pcf8591.h>
#include "Lcd_I2C_Ardubot.h"

//constants del nom de cada pin a la llibreria WirinPi
#define LEDR 3
#define LEDY 2
#define LEDG 0
#define paraxocsE 5
#define paraxocsD 4
#define SDD 27
#define SCD 28
#define SCE 29
#define SEE 25
#define boto 6
#define bateria 0
#define rodaDdav 23
#define rodaDenr 26
#define rodaEdav 24
#define rodaEenr 1
#define encoderEs 21
#define encoderDr 22

#define I2C_Analog 0x48 //Adeça I2C  del mòdul analògic
#define PCF 64 // número base del mòdul

//variables del programa
int contEs = 0;
int contDr = 0;
int sniferEE = 0;
int sniferCE = 0;
int sniferCD = 0;
int sniferDD = 0;
int pxE = 0;
int pxD = 0;
float estatbat = 0;
char variablesLCD[16];

int main() {
```

```
      int fdLCD;

      wiringPiSetup () ; //inici de la llibreria wiringPi

    fdLCD = LCD_InitI2C();//inicia la comunicació i2c per la pantalla
    sendLcdInit(fdLCD);//Inicial pantalla

      pcf8591Setup (PCF, I2C_Analog) ; //inicial el mòdul analògic

      pinMode(LEDR, OUTPUT);
      pinMode(LEDY, OUTPUT);
      pinMode(LEDG, OUTPUT);
      pinMode(paraxocsD, INPUT);
      pinMode(paraxocsE, INPUT);
      pinMode(SDD, INPUT);
      pinMode(SCD, INPUT);
      pinMode(SCE, INPUT);
      pinMode(SEE, INPUT);
      pinMode(boto, INPUT);

      //inicialització dels PWM per sofware
      softPwmCreate(rodaDdav, 0, 255);
      softPwmCreate(rodaDenr, 0, 255);
      softPwmCreate(rodaEdav, 0, 255);
      softPwmCreate(rodaEenr, 0, 255);

      //definició de les interrupcions
      wiringPiISR (encoderEs, INT_EDGE_BOTH, &intEs);
      wiringPiISR (encoderDr, INT_EDGE_BOTH, &intDr);

      while(1) {
            sniferEE = digitalRead(SEE); //lactura deles sensors de contrast pel
segiodor
            sniferCE = digitalRead(SCE);
            sniferCD = digitalRead(SCD);
            sniferDD = digitalRead(SDD);

            //realització de les accions a fer per a seguir la línia
            if (sniferEE == LOW and sniferCE == HIGH and sniferCD == HIGH and sniferDD
== LOW)
            {
              softPwmWrite(rodaDdav, 200);
              softPwmWrite(rodaEdav, 200);
              softPwmWrite(rodaDenr, 0);
```

```
              softPwmWrite(rodaEenr, 0);
             }
             else if (sniferEE == LOW and sniferCE == HIGH and sniferCD == LOW and
sniferDD == LOW)
             {
               softPwmWrite(rodaDdav, 120);
               softPwmWrite(rodaEdav, 200);
               softPwmWrite(rodaDenr, 0);
               softPwmWrite(rodaEenr, 0);
             }
             else if (sniferEE == LOW and sniferCE == LOW and sniferCD == HIGH and
sniferDD == LOW)
             {
               softPwmWrite(rodaDdav, 200);
               softPwmWrite(rodaEdav, 120);
               softPwmWrite(rodaDenr, 0);
               softPwmWrite(rodaEenr, 0);
             }
             else if (sniferEE == HIGH and sniferCE == HIGH and sniferCD == LOW and
sniferDD == LOW)
             {
               softPwmWrite(rodaDdav, 0);
               softPwmWrite(rodaEdav, 200);
               softPwmWrite(rodaDenr, 0);
               softPwmWrite(rodaEenr, 0);
             }
             else if (sniferEE == LOW and sniferCE == LOW and sniferCD == HIGH and
sniferDD == HIGH)
             {
               softPwmWrite(rodaDdav, 200);
               softPwmWrite(rodaEdav, 0);
               softPwmWrite(rodaDenr, 0);
               softPwmWrite(rodaEenr, 0);
             }
             else
             {
               softPwmWrite(rodaDdav, 0);
               softPwmWrite(rodaEdav, 0);
               softPwmWrite(rodaDenr, 0);
               softPwmWrite(rodaEenr, 0);
             }

             //lectura del parxocs
             pxE = digitalRead(paraxocsE);
```

```
            pxD = digitalRead(paraxocsD);

            //quan es toca el paraxocs esquerra s'encen el LED verd
            //quan es toca el paraxocs dret s'encen el LED vermell
            //quan es toqun els dos paraxocs alhora s'encen el LED groc
            //quan no s'en toca cap no s'encen cap LED
            if (pxE==HIGH and pxD==LOW)
            {
              digitalWrite(LEDR, LOW);
              digitalWrite(LEDG, HIGH);
              digitalWrite(LEDY, LOW);
            }
            else if(pxE == LOW and pxD==HIGH)
            {
              digitalWrite(LEDR, HIGH);
              digitalWrite(LEDG, LOW);
              digitalWrite(LEDY, LOW);
            }
            else if (pxD==HIGH and pxE==HIGH)
            {
              digitalWrite(LEDR, LOW);
              digitalWrite(LEDG, LOW);
              digitalWrite(LEDY, HIGH);
            }
            else
            {
              digitalWrite(LEDR, LOW);
              digitalWrite(LEDG, LOW);
              digitalWrite(LEDY, LOW);
            }

            //Quan es prem el polsador es mosta a la pantalla el nombre de franges
que ha comptat cada encoder
            //Sino es mostra Ardubot 4.1 i l'estat de la bateria
            if (digitalRead(boto) ==0)
            {
                  sendLcdCursorLocate(fdLCD,0,1);
                  sendLcdString(fdLCD,"Dr ");
                  sprintf (variablesLCD, contDr);//per imprimir una variable s'ha
de pasar a string
                  sendLcdString(fdLCD,variablesLCD);
                  sendLcdString(fdLCD," Es ");
                  sprintf (variablesLCD, contEs);
                  sendLcdString(fdLCD,variablesLCD);
```

```
            }
            else
            {
                  estatbat = (analogRead (PCF + bateria)/2.5) ;
                  sendLcdCursorLocate(fdLCD,0,1);
                  sendLcdString(fdLCD," Bateria ");
                  sprintf (variablesLCD, estatbat);//per imprimir una variable s'ha
de pasar a string
                  sendLcdString(fdLCD,variablesLCD);
                  sendLcdString(fdLCD," V");
            }

      }
      LCD_CloseI2C(fd);
}

//interrupció de l'ecoder esquerra
    void intEs() //quan hi ha l'interrupció que detecte el color blanc de la roda
suma 1 a contEs
    {
      contEs = contEs+1;
    }

//interrupció de l'encoder dret
    void intDr()
    {
      contDr = contDr+1;
    }
```

D DISSENY DE LA RODA AMB ENCODER

```
//Roda amb encoder incorporat per a l'Ardubot 4.1

resolution = 200; //resolució, 180 és la maxia de la impresora
$fn = resolution;

//Paràmetres manipulables
Hroda = 9; // gruix de la roda
Nforats = 12; //nº de forats que hi hauran a l'encoder (negre)
Hpneumatic = 2; //gruix del pneumàtic
Motor_nou = false; //si és el motor nou

//Paràmetres NO manipulables
Roda_original = 52;
Din = Roda_original - 3; // diametre interior del pneumàrica (no pot ser més petit
que això si es vol utilitzar l'encoder.
Dout = (Hpneumatic*2) + Din; //diametre exterior de la roda

module cilindre(){
    color("cyan")
    cylinder(d = Dout, h = Hroda, center = true);
}

Angle = 360/(Nforats*2);
module triangle(){
    Llargada = (Dout/2);
    //Angle = 360/(Nforats*2);
    Yt = cos(Angle)*Llargada;
    Xt = sin(Angle)*Llargada;
    linear_extrude(height = Hroda, center = true)
    polygon([[0,0],[0,Llargada],[Xt,Yt]]);
}

module forats(){
    for (a = [0: Nforats]){
        rotate(a*Angle*2){
            triangle();
        }
    }
}

module encoder(){
    difference(){
        cilindre();
```

```
        forats();
    }
}

module pneumatic(){
    color("green")
    difference(){
        cylinder(d = Dout, h = Hroda, center = true);
        cylinder(d = Din, h = Hroda, center = true);
    }
}

Hcentre = 3; //mm que s'eleva el centre de la roda
module centre(){

    Dcentre = 10; // diametre del rentre
    Drefor = 25; // diametre del reforç

    translate([0,0,-(Hroda/2)])
    union(){
        color("red")
        cylinder(d = Dcentre, h = (Hroda+Hcentre));
        color("cyan")
        cylinder(d = Drefor, h = Hroda);
    }
}

module foratC(){
    eix = 3; //diàmetre de l'eix
    Hforat = Hroda+Hcentre+10; //llargada del forat
    if (Motor_nou){
        color("orange")
        difference(){
            cylinder(d = eix, h = (Hforat),center = true);
            dis = (((Hforat)/2)+(2.45-eix/2)); //distancia que es desplaça del centre
            translate([dis,0,0])
            cube([Hforat, Hforat,Hforat],true);
        }
    }
    else{
        color("orange")
        cylinder(d = eix, h = (Hforat), center = true);
    }
}
```

```
ajust = 0.94;
module roda(){
    difference(){
        union(){
            encoder();
            centre();
            pneumatic();
        }
     scale([ajust, ajust, ajust])
     foratC();
    }
}

module logo(){
    translate([-0.7,-308.5,-(Hroda/2)-1])
    linear_extrude(height = 3)
    import ("LogoEu2.dxf");
}

module logo_text(){
    font = "Liberation Sans";
    Lletre = "EPS";
    translate([0, 7, ((-Hroda/2)+2)])
    rotate([180, 0, 0])
    linear_extrude(height = 3) {
    text(Lletre, size = (5), font = font, halign = "center", valign = "center", $fn
= 200);
  }
}

module Forma_relleu(){
    font = "Liberation Sans";
    Lletre = "S";
    translate([0, -(Dout/2), 0])
    rotate([90, 0, 0])
    linear_extrude(height = 0.4) {
    text(Lletre, size = (Hroda-3), font = font, halign = "center", valign = "center",
$fn = 200);
  }
}

module relleu(){
    NumeroRelleu = 28;
```

```
    AngleRelleu = 360/NumeroRelleu;
     for (a = [0:NumeroRelleu]){
        rotate(a*AngleRelleu){
            Forma_relleu();
        }
    }
}

module fi_logo(){
    difference(){
        roda();
        logo();
    }
}

module fi_logoText(){
    difference(){
        roda();
        logo_text();
    }
}

module fi_relleu(){
    union(){
        roda();
        relleu();
    }
}
//cilindre();
//triangle();
//forats();
//encoder();
//centre();
//foratC();
//logo();
//roda();
//fi_logo();
//relleu();
//Forma_relleu();
//logo_text();
fi_relleu();
//fi_logoText();
```