Legobot: Electrònica de control del robot.

Esteban Centellas Vela
Grau d'Enginyeria Informàtica
Sistemes encastats

Jordi Bécares Ferrer
Pere Tuset Peiró

14/01/2018

FITXA DEL TREBALL FINAL

Títol del treball:	Legobot: Electrònica de control del robot
Nom de l'autor:	*Esteban Centellas Vela*
Nom del consultor/a:	*Jordi Bécares Ferrer*
Nom del PRA:	*Pere Tuset Peiró*
Data de lliurament (mm/aaaa):	*01/2018*
Titulació o programa:	*Grau d'Enginyeria Informàtica*
Àrea del Treball Final:	*Sistemes encastats*
Idioma del treball:	*Català*
Paraules clau	*Robot* *FreeRTOS* *MSP432P401R*

Resum del Treball:

El punt de partida d'aquest projecte ha estat un robot fet amb Lego, que disposa de dos rodes mogudes per motors individuals, un braç articulat amb 2 graus de llibertat i una mà que pot agafar objectes lleugers.

La finalitat del projecte ha estat desenvolupar una electrònica de control per tal de donar-li una mobilitat i percepció del entorn bàsica així com alguns moviments automàtics.

El projecte s'ha desenvolupat dins del context del projecte de final de grau per tal de desenvolupar coneixement adquirits en els estudis. Per aquest motiu, s'han fet servir els recursos que hi havia disponibles per tal de reduir al màxim les despeses i d'aquesta manera es factible reproduir el projecte.

El desenvolupament s'ha basat en la placa de Texas Instruments MSP-EXP432P401R [2] amb la extensió Educational BoosterPack MKII [4] ; i amb el hardware addicional necessari, implementat a mà, connectat als ports disponibles de la placa principal.

El resultat final ha estat un robot controlat per dispositius de interfície persona-computador disponibles a la Educational BoosterPack MKII i unes llibreries de control basic del mateix que permeten ampliar el projecte fàcilment.

Abstract:

The starting point of the project is a robot made with Lego, which has two wheels moved by two stepper motors, an arm with two movements and a hand able to

take light objects.

The finality of the project has been to develop the control electronics to give it mobility and environment perception and some automatic movement.

The project has been developed inside the end degree work to develop the acknowledge acquired during the studies. For this reason, the used resources have been what were available to reduce the expenses and to make possible to reproduce the project.

The development has been based on the Texas Instruments board MSP-EXP432P401R [2] with the extension Educational BoosterPack MKII [4] ; and with the needed additional handmade hardware connected to the available ports from the main board.

The final results have been a robot controlled by the human interfaces devices available in the Educational BoosterPack MKII and some basic libraries to manage the basic functionality that allows upgrading the project easily.

Índex

Llista de figures

1. Introducció

1.1 Context i justificació del treball

El punt de partida d'aquest projecte es un robot construït amb Lego, que disposa de dos rodes mogudes per dos motors pas a pas, un braç amb dos graus de llibertat, una mà que pot agafar objectes lleugers i tres sensors d'ultrasons. Aquest robot [6] es va construir fa uns anys i en el punt de partida d'aquest projecte estava controlat per una electrònica basada en el microcontrolador de Atmel ATMega64 que es movia a partir de les comandes rebudes des de un PC via Bluetooth a través d'una aplicació.

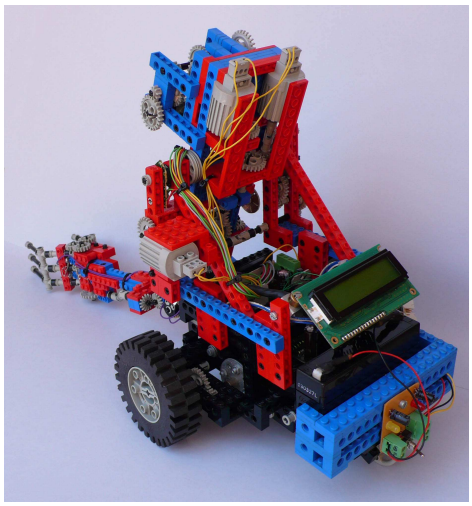


Figura 1. Robot Legobot

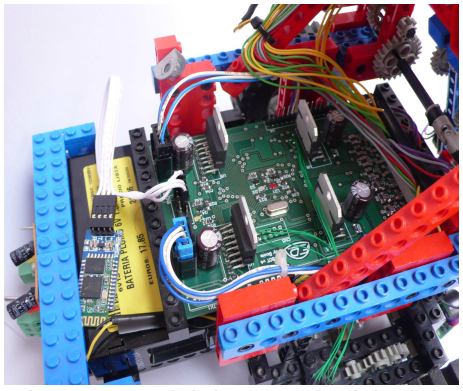


Figura 2. Modul de control original de Legobot

El sistema inicial està implementat sobre un microcontrolador de 8 bits de Atmel amb recursos limitats i programat amb un scheduler. Es per aquest motiu es vol modernitzar el sistema fent servir una servir una plataforma amb un microcontrolador MSP432P401R [2], basat amb un core ARM 32-Bit Cortex, amb molts més recursos i més velocitat.

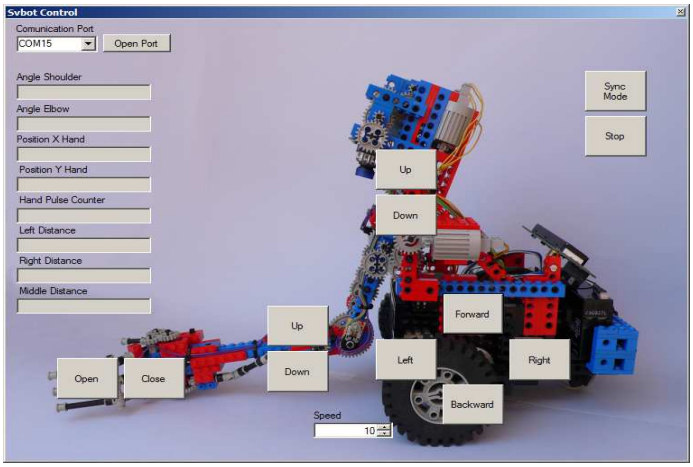


Figura 3. Aplicació que controlava el robot inicialment

El resultat final es una plataforma més ràpida, més modulable i escalable que permet seguir ampliant les prestacions del sistema. Això s'aconsegueix, primer per la integració d'una plataforma més rapida, amb més recursos i més pins disponibles; i segon, per la integració que es farà d'un sistema operatiu encastat per gestionar el producte final.

1.2 Descripció del treball

Les característiques del robot que tenim com a punt de partida son les següents:

- Moviment de desplaçament mitjançant un motor pas a pas a cada roda permetent controlar amb precisió aquest moviment.
- Moviment del braç amb dos graus de llibertat i dos potenciòmetres acoblats que permeten tindre una lectura analògica de la seva posició en tot moment.
- Moviment de la mà amb lectura de fi de carrera quan s'obre i de polsos mitjançant un encoder per mesura l'apertura d'aquesta
- Tres sensors d'ultrasons en la part davantera per detectar objectes. Aquests mesuren la distancia als objectes que estan davant seu.

En el projecte s'ha desenvolupat un sistema de control que permet gestionar els moviments d'aquest robot. Les característiques d'aquest control son:

- Es controla a través dels botons i joystick que hi ha en Educational BoosterPack MKII [4].
- El sistema te diferents modes de funcionament que es pot canviar mitjançant el polsador del joystick.
- El display informa del mode actual i mostra una ajuda explicant com funciona aquest mode.
- El primer mode es el de translació on podem desplaçar el robot mitjançant el joystick. Podem graduar la velocitat de desplaçament segons com prenem el joystick.

- En el segon mode es controla el moviment del braç. Es pot moure el braç a munt i a baix amb el eix vertical del joystick, però aquest es fa sincronitzant els motors de ombro i colze mantenint el avant braç paral·lel al terra. Moure el braç d'aquesta manera fa que sigui més fàcil controlar-lo ja que només necessitem un eix per moure tot el braç. Amb el eix horitzontal es controla el moviment del colze per tal d'acabar d'ajustar la posició. Per últim es pot obrir i tancar la mà amb el botons u i dos.
- El últim mode està pensat per programar funcions automàtiques. Finalment només s'ha programat una funcionalitat que avança fins que troba un objecte i l'agafa amb la mà.

1.3 Objectius del TFC

Els objectius d'aquest treball es dotar al robot de moviment per mitjà d'un joystick amb la possibilitat de fer alguns moviments autònoms.

Els objectius bàsics del projecte son:

- Dotar al robot amb moviment de desplaçament horitzontal.
- Dotar al robot moviment en el braç. Aquest te dos graus de llibertat que son el colze i el ombro. També te el moviment de la mà.
- Implementar la capacitat de conèixer la posició del braç en tot moment.
- Moure al robot amb el joystick del Educational BoosterPack MKII [4].
- Afegir moviment del braç sincronitzant la velocitat dels motors per tal de mantenir el avantbraç paral·lel al terra.
- Tindre capacitat de coneixement del entorn a través de sensors d'ultrasons.

Els objectius secundaris son

- Presentar informació en el display del estat del sistema.
- Implementació de modes autònoms per realitzar accions.
- Implementació de modes low power.
- Afegir gràfics en la informació mostrada en el display

1.4 Enfocament i mètode seguit

Per desenvolupar un treball d'aquestes característiques cal primer fer un estudi de les característiques de hardware que necessitem. Aquestes característiques han de ser tant a nivell de entrades i sortides del sistema com de prestacions d'aquest. Aquest primer pas es comú a les diferents estratègies i un cop fet em d'escollir quina estratègia farem servir per continuar. En aquest cas tenim dos possibilitat

- Desenvolupar el codi des de cero. Aquesta possibilitat ens permet fer un codi més optimitzat i controlar els recursos del sistema emprats. Per tant es podria basar el projecte en un microcontrolador amb menys recursos, sobretot de memòria i velocitat.

- Fer servir un el sistema operatiu FreeRTOS com a punt de partida i aprofitar les llibreries donades per el fabricant del microcontrolador per gestionar els perifèrics. Amb aquesta opció escurcem els terminis de desenvolupament i reduïm les possibilitat d'errors, degut a que es fa servir software estàndard i per tant més provat. En contra tenim que necessitem una plataforma amb més recursos, per el fet de fer servir un sistema operatiu i per fer servir software que no està optimitzat per el nostra sistema.

La primera opció reduiria els costos de producció del producte final per el fet de fer servir una plataforma més barata. La segona redueix els costos de desenvolupament per el fet de fer servir software existent i per no haver de fer tant testos. Per tant, la primera opció seria més adequada per productes amb una producció molt gran, on contrarestaríem els costos de desenvolupament amb els de producció, i la segona es més adequada per produccions petites. Com que estem en un projecte que no s'ha de portar a producció i disposem d'una plataforma que te molts més recursos dels necessaris, la millor opció es la segona.

En el desenvolupament del codi definirem un petit procés. Les regles que seguirem son:

- Tot el codi es farà comentat amb estil Doxygen [7]. D'aquesta manera es podrà generar documentació automàticament on es veuran les relacions entre components i les seves APIs.
- Es farà servir una base de dades local de Subversión [9]. Així es tindrà un històric de tot els desenvolupament i es podran recuperar versions antigues dels components.
- Totes les maquines d'estat que puguin aparèixer durant el desenvolupament es faran amb l'aplicació FSM (Finite State Machine Editor and C-code generator) [8]

1.5 Planificació del treball

Inicialment es va fer una planificació amb les següents tasques:

- Muntar el prototip (T1): En primer lloc cal muntar el prototip incloent el muntatge del hardware en el robot. Les subtasques a realitzar son:
  - Disseny de hardware (T1.1): Fer els esquemàtics del hardware addicional que s'ha de fer i especificar les connexions amb la resta de mòduls i el robot.
  - Fabricació del hardware addicional (T1.2): Cal fer un prototip en placa de forats dels hardware addicional.
  - Muntatge del sistema (T1.3): Es muntarà els diferents mòduls electrònics que s'acoblaran al robot.

- Configuració del sistema operatiu (T2): Cal configurar el sistema operatiu i afegit les tasques que necessitarem testejant que es cridin correctament.

- Fer el codi necessari per fer que el robot es desplaci horitzontalment (T3): Per poder fer que es mogui tenim les següents subtasques:
  - Implementació de drivers dels ports de baix nivell (T3.1): Cal fer drivers per poder gestionar els ports que farem servir per moure els motors de les rodes. Es a dir configurar els ports com a sortides i fer les funcions per poder controlar-los.
  - Implementació de drivers d'alt nivell (T3.2): Afegirem una segona capa d'abstracció on s'implementaran funcions del tipus moure el robot endavant, o cap a darrera o girar.

- Fer el codi per moure el braç (T4): Hem de desenvolupar els divers necessaris per moure el braç. Les subtasques son:
  - Desenvolupament de driver PWM (T4.1): En aquests motors cal controlar la velocitat per tant funcionaran amb un driver PWM.
  - Fer els divers d'alt nivell per moure els motors (T4.2): Caldrà afegir les funcions del driver d'alt nivell per moure els motors del braç. Cal controlar cap a on es mouen els motors i amb quina velocitat ho fan.

- Fer el codi per llegir la posició del braç (T5): Aquí hem de poder llegir la posició del braç en tot moment. Les subtasques a desenvolupar son les següents:
  - Implementació de divers de baix nivell dels ADC (T5.1): Necessitem llegir els convertidor analògic digital i crear la tasca periòdica necessària per fer-ho.
  - Fer el driver d'alt nivell per llegir el sensor de final de carrera (T5.2): En aquest cas com que ja tenim els drivers de baix nivell fets en etapas anterior, desenvoluparem directament aquest driver que ha d'indicar el estat del sensor de final de carrera.
  - Desenvolupar el driver d'alt nivell de la posició de la mà (T5.3): Aquest driver ha de llegir els polsos que indiquen la posició de la mà. Cal assegurar que no es perdi cal pols i s'haurà d'avaluar si es farà amb tasca periòdica o per interrupció.

- Fer el codi per fer les lectures del sensor d'ultrasons (T6): Per fer aquestes lectures cal desenvolupar les següents tasques:
  - Implementació de drivers I2C de baix nivell (T6.1): Cal fer la comunicació bàsica a nivell de trames amb els sensors.
  - Implementació de drivers d'alt nivell dels sensors (T6.2): Aquí desenvoluparem el codi per descodificar la informació dels sensors per separat i llegir constantment la distancia al objecte més proper.

- Fer el codi del Educational BoosterPack MKII [4] (T7): Per fer anar el Educational BoosterPack MKII tenim les següents subtasques:

- Integrar el codi de control del Educational BoosterPack MKII (T7.1): El Educational BoosterPack MKII ja te uns divers disponibles a la web que es poden integrar en el sistema.
- Implementar les funcions per mostrar per pantalla les dades (T7.2): Començaríem per mostrar un informació molt simple en mode text però podríem acabar afegint dibuixos

- Moure el robot amb les ordes del Joystick (T8): En aquest pas farem moure el robot connectant els drivers del joystick i tecles als que mouen el robot.
- Implementació de modes autònoms (T9): Aquí tenim per implementar funcions autònomes cop pot ser agafar un objecte automàticament. Aquestes serien funcions d'aplicació

- Implementació de modes low power (T10): Per últim faríem que el hardware anés a mode low power quan estigui inactiu.

En la següent figura es pot veure el diagrama de Gantt de la planificació inicial.

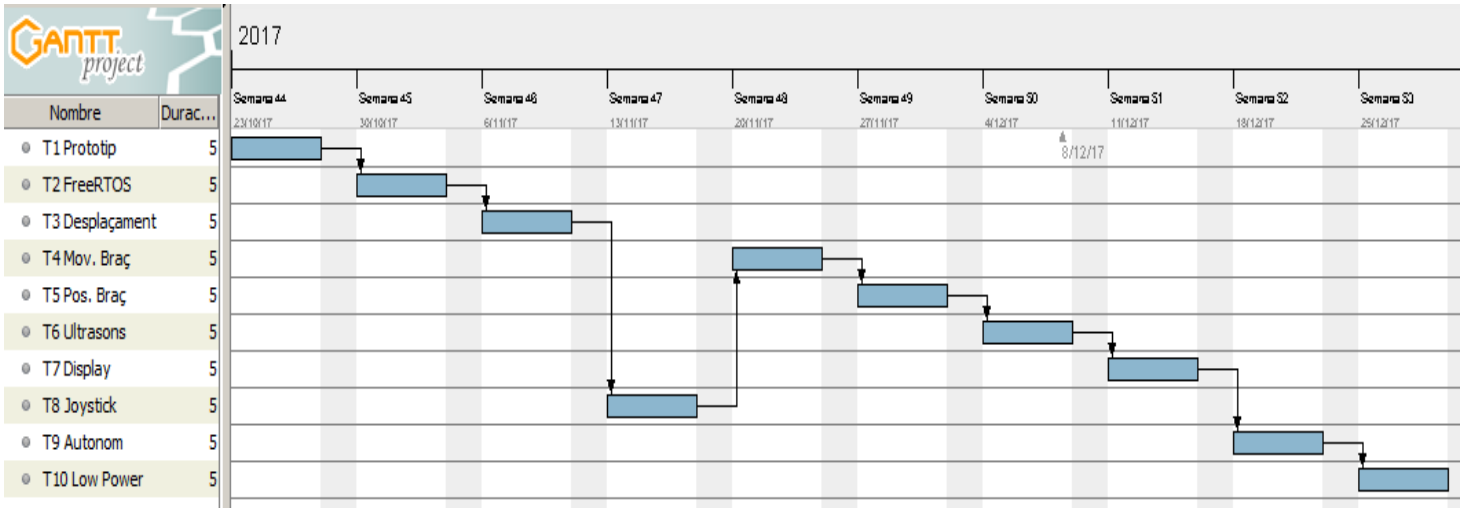


Figura 4. Diagrama de Gantt inicial

Així, les tasques previstes per la PAC 2 eren

- Muntar el prototip (T1)
- Configuració del sistema operatiu (T2)
- Fer que el robot es desplaci horitzontalment (T3)
- Moure el robot amb les ordes del Joystick (T8)

Per la PAC 3 les tasques previstes eren:

- Fer el codi per moure el braç (T4).

- Fer el codi per llegir la posició del braç (T5).
- Fer el codi per fer les lectures del sensor d'ultra sons (T6).
- Fer el codi del Educational BoosterPack MKII (T7).

Durant el desenvolupament del projecte es va afegir un altre tasca que es la següent:

- Implementació del moviment sincronitzant del braç (T11). Aquesta tasca consisteix en fer servir les lectures de la posició del braç; i controlant la velocitat dels motors, fer moure el braç mantenint el avantbraç paral·lel al terra.

El diagrama de Gantt de la execució final del projecte ha estat com es veu en la figura següent.

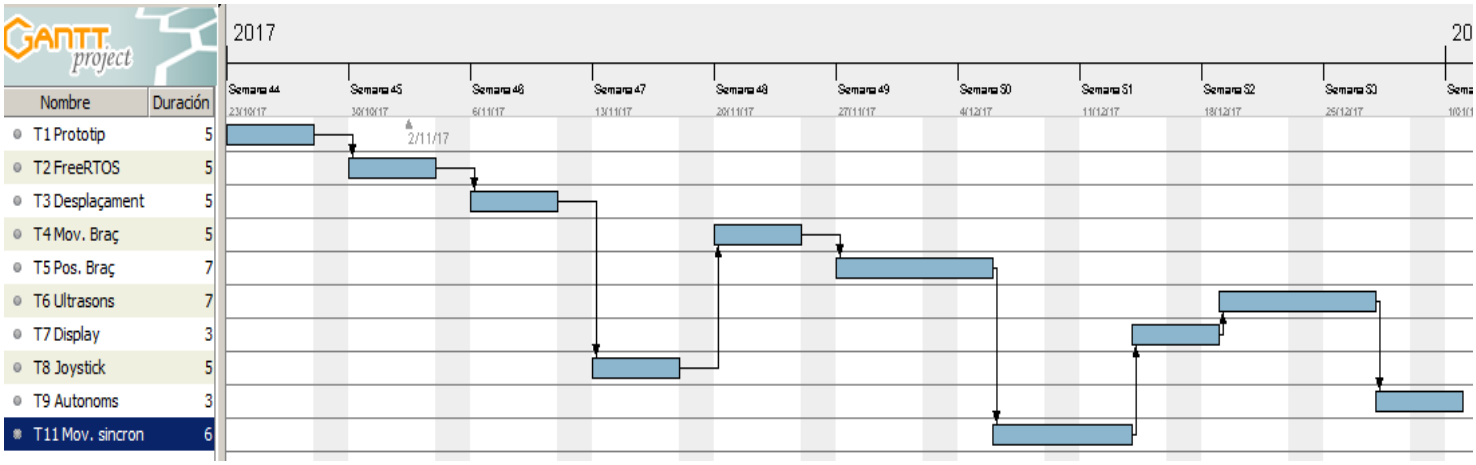


Figura 5. Diagrama de Gantt final

Durant el desenvolupament de la PAC2 es va veure que el muntatge del prototip era una tasca més llarga del previst, per el que es va optar per muntar només les parts necessàries; per tant en la entrega d'aquesta PAC només estava muntada la part de la electrònica de control dels motors que mouen les rodes. En la PAC3 es va acabar de muntar la electrònica que faltava però es va enrederir la implementació del sensor d'ultrasons i no es van poder acabar a temps. En la última entrega de codi, degut als problemes en la implementació d'aquest sensors, es va decidir deixar-los per al final. També es va introduir la tasca nova per implementar el moviment sincronitzat i es va eliminar la implementació dels low power modes per falta de temps.

1.6 Recursos emprats

Els recursos utilitzats en el projecte han estat:

- Code Composer Studio 7.3.0 amb compilador de C i recursos de software de Texas instruments
- Placa MSP-EXP432P401R [2]
- Placa Educational BoosterPack MKII [4].
- Ordinador HP Pavilion dv6 Notebook PC i7 a 1.6GHz i 4GB

- Font de PC de 400W
- Legobot
- Placa de prototips
- Soldador JBC 30S
- Oscil·loscopi Tektronic TDS 1002 de dos canals a 60MHz
- Tester Meterman 15XL
- Analitzador lògic Salae de 8 canals a 24 Ms/s amb aplicació Salae Logic 1.1.34 Beta
- Aplicació FSM Diagram Editor 0.5r10. Es una aplicació que genera codi en C a partir de diagrames d'estats.

1.7 Productes obtinguts

Finalment s'ha obtingut un robot que es capaç de moure's controlat per un joytick i que pot fer alguns moviments autònoms programats. També està equipat amb sensors que poden detectar objectes propers.

També hem obtingut una plataforma de software capaç de fer el control bàsic del sistema i que permet fer ampliacions abstraient-se del robot i del maquinari en si,

1.8 Breu descripció dels altres capítols de la memòria.

En els pròxims capítols començarem per una breu descripció de les tecnologies utilitzades en projecte, així com alguna alternativa i de productes similars. Després en el capítol 3 farem una descripció introductòria del component que conformen el sistema que desenvoluparem en detall en el capítol 4. Seguirem amb una descripció de la vialitat tècnica en el capítol 5. Després descriurem les conclusions explicant que s'ha aprés, que es pot millorar i quines son les pròximes evolucions del projecte. Acabarem amb un glossari de conceptes i la bibliografía.

2 Antecedents

2.1 Estat de l'art

El principal component del projecte es la placa que conté el microcontrolador MSP-EXP432P401R [2]. Aquesta placa està equipada amb un microcontrolador MSP432P401R i les seves principals característiques son:

- Arquitectura ARM Cortex-M4F
- Rellotge de fins 48MHz
- 256KB de flash, 64KB de SRAM i 32KB de ROM
- 4 timers de 16 bits amb capture, compare, PWM, 2 timers de 32 bits i un Real Time Clock
- Fins a 8 canals de comunicacions amb I2C, SPI, UART i IrDA
- Entrades analògiques de precisió, capacitive touch i comparador
- Connector JTAG

Actualment hi ha al mercat moltes plaques de desenvolupament amb característiques semblants, però la principal plataforma es Arduino. Arduino te múltiples kits de desenvolupament amb moltes arquitectures que s'adapten a les necessitats del desenvolupador. Agafem com exemple les característiques del Arduino M0 Pro que son:

- Arquitectura ARM Corex-MO+
- Rellotge de 48Mhz
- 256KB de Flash, 32KB de SRAM
- 6 entrades analógiques
- 20 sortides PWM
- Port serie, I2C, SPI
- Connector JTAG

Un altre família de plaques d'aquest tipus son le LPCXpresso de NXP que també està formada per sistemes amb diferents característiques. Analitzarem les característiques de la placa LPCCPresso54628.

- Arquitectura ARM Cortex-M4
- Rellotge fins a 220MHz
- Display 272x480 de color tactil
- USB amb debugger Segger
- Múltiples opcions d'expansions
- Connector Ethernet

2.2 Estudi de mercat

Actualment al mercat trobem el set Lego Mindstorms EV3 [5] que combina la versatilitat del sistema de construcció Lego amb tecnologia electrònica que permet construir robots amb una certa intel·ligència i coneixement del seu entorn. El sistema te un entorn de programació basat en Labview i blocs que permet fer codi molt potent d'una manera molt sencilla

Figura 6. Lego Mindstroms EV3

Les característiques d'aquest sistema son:

- El cor del set es el mòdul intel·ligent EV3, equipat amb un potent microprocessador ARM9, un port USB per proporcionar funcions WiFi i connexió a Internet, un lector de targetes Micro SD, botons retroil·luminats i 4 ports de motor.
- Inclou 3 servomotors interactius, un control remot, un sensor de color, un sensor de contacte, un sensor d'infraroig i mes de 550 elements Lego Technic.
- Aplicació intuïtiva de programació EV3, disponible per tabletes iOS, Android, PC i Mac.
- Controlable a través del sensor de infraroig.

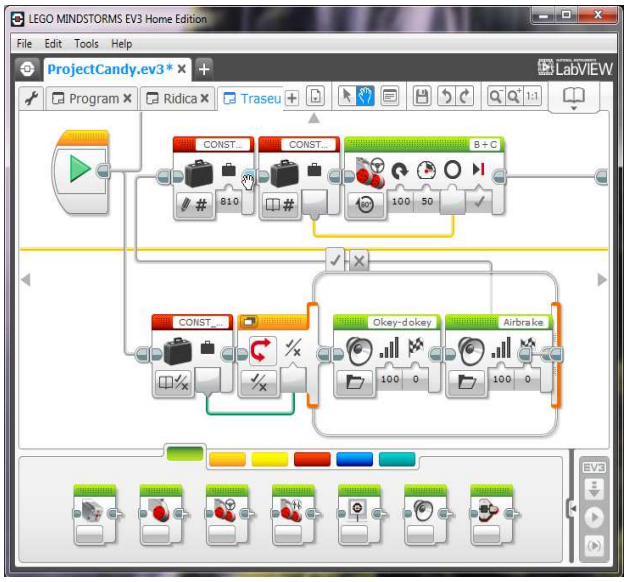


Figura 7. Aplicació Lego Mindstorms EV3 Home Edition

3 Descripció funcional

3.1 Legobot

El robot Legobot està fet amb Lego Technic clàssic i es el punt de partida del projecte [6]. El seu disseny no es part del projecte però es necessari conèixer les seves característiques. Aquestes son:

- Te una base amb dos rodes controlades individualment per dos motors pas a pas. Els motors pas a pas permeten moure el robot amb precisió
- Moviment de ombro controlat per un motor original de Lego amb mobilitat des de els 10° als 120°.
- Moviment del colze controlat per un motor original de Lego amb mobilitat des de les 90° fins als 315°.
- Apertura i tancament de la mà moguts amb un motor original de Lego.
- Mesura de la posició del ombro amb un potenciòmetre acoblat al eix.
- Mesura de la posició del colze amb un potenciòmetre acoblat al eix.
- Sensor de final de carrera en la apertura de la mà.
- Encoder que mesura polses en el moviment de la mà
- 3 sensors I2C d'ultrasons situats a la part davantera.

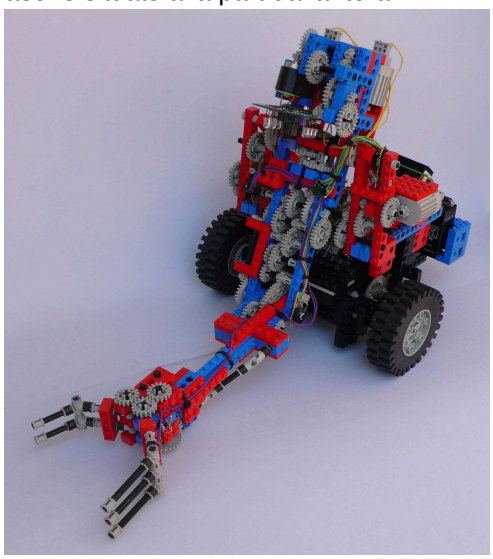


Figura 8. Imatge del Legobot

3.2 Maquinari

Passem a descriure el maquinari que controla el Legobot. El maquinari està basat en una placa MSP-EXP432P401R [2] on connectarem un Educational BoosterPack MKII [4]. Del Educational BoosterPack MKII farem servir el Joystick i el display. També utilitzarem el port EDGE_CON del MSP-EXP432P401R que dona accés als pins dels microcontrolador, per crear una expansió de maquinari per poder accedir al robot. El diagrama de blocs del maquinari del sistema es:

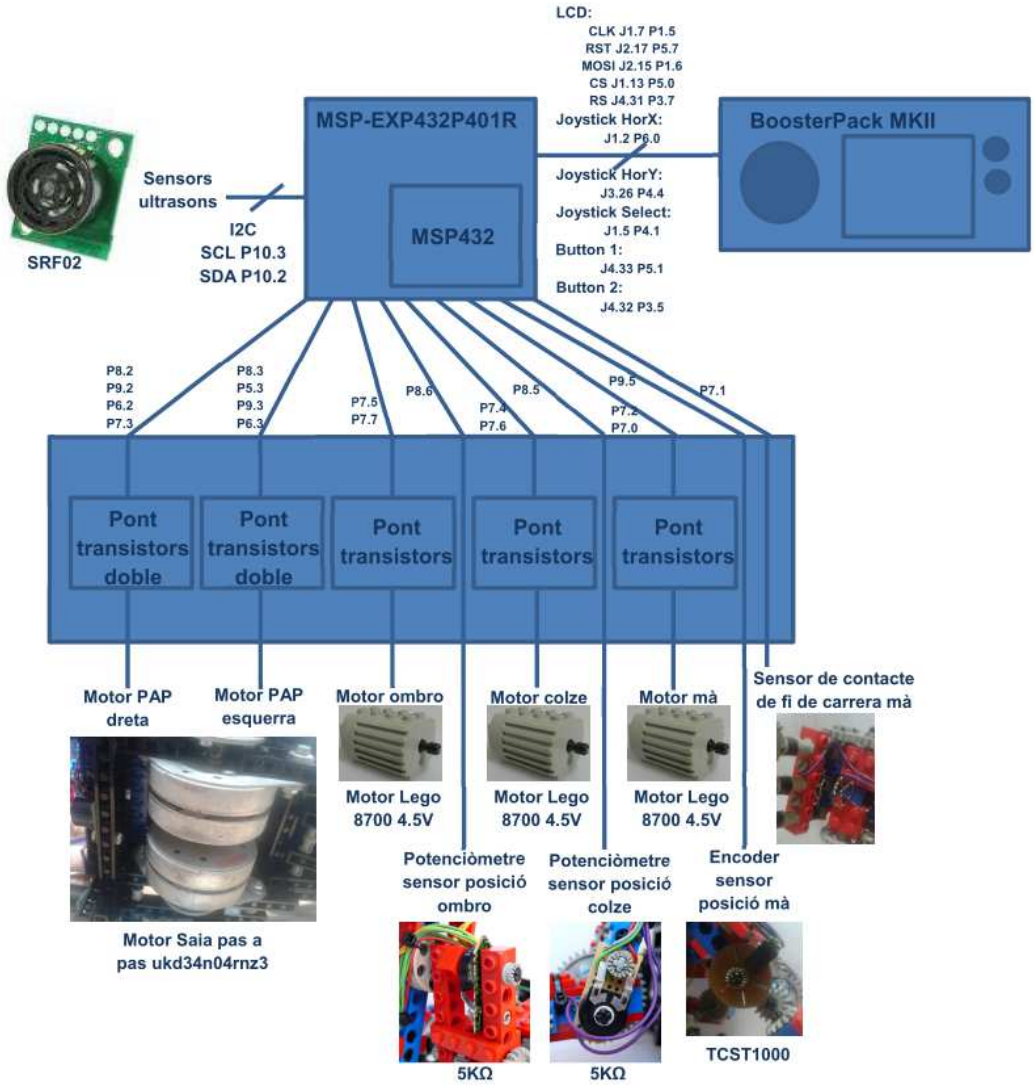


Figura 9. Diagrama de blocs del maquinari dels sistema

Les característiques del maquinari d'expansió son:

- Dos ponts de transistors dobles per gestionar els dos motors pas a pas.
- Tres ponts de transistors simples per controlar els tres motor que mouen el colze, ombro i mà.
- Maquinari per adaptar el senyal que ve dels dos potenciòmetres que mesuren la posició del colze i ombro.
- Connexions dels senyals de fi de carrera i polsos del braç cap a la placa base.

- Connexions i resistències de pull-up del bus I2C que va cap als sensors d'ultrasons.

També tenim restriccions de disseny que son:

- Com a restricció en el disseny del programari s'ha de tenir en compte que no es poden activar els dos pins d'activació d'un mateix pont de transistors a l'hora. Per tant, quan s'activa un dels pins que el controlen, s'ha d'assegurar que l'altre pin està desactivat. Connectar els dos pins al mateix moment provocaria un curtcircuit entre positiu i massa.
- Cal alimentar el maquinari d'expansió amb tres fonts de tensió diferents.
  - En primer lloc can es necessita la sortida de 3V3 de la placa MSP-EXP432P401R per polaritzar els potenciòmetres que mesuren la posició del braç al igual que el maquinari que mesura la posició i el final de carrera de la mà.
  - Alimentarem els sensors d'ultrasons amb els 5V de la placa MSP-EXP432P401R així com les resistències de polarització de bus I2C
  - Alimentació externa de 5V als ponts de transistors. Aquesta es farà a traves de una font de PC modificada.

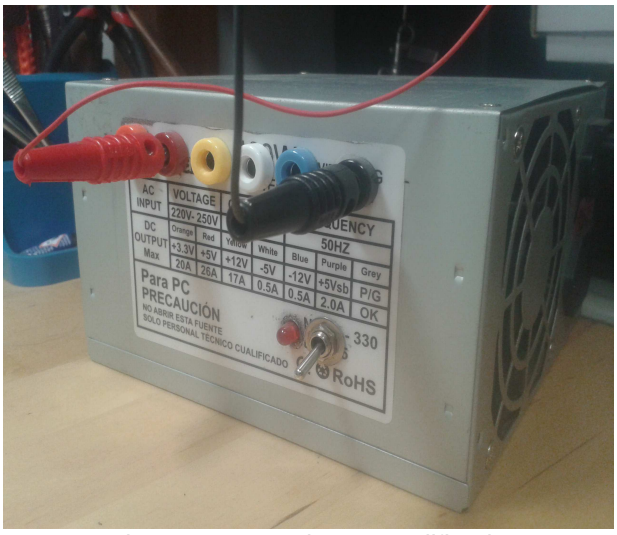


Figura 10. Font de PC modificada

3.3 Programari

El desenvolupament del programari que va a la placa MSP-EXP432P401R [2] ha estat la part principal del desenvolupament d'aquest projecte. La primera decisió pressa en el disseny del programari es fer servir un sistema operatiu en temps real. El sistema operatiu en temps real ens assegura que les tasques s'executaran dins d'uns temps determinats. En aquest cas em escollit el sistema operatiu FreeRTOS [1].

En el sistema operatiu definirem les tasques que necessitarem i quina prioritat tindran. En la següent llista tenim les tasques escollides ordenades de mes prioritat a menys:

- **Tasca del driver dels motors pas a pas.** Volem que els motors es moguin de forma continuada, per tant necessitem que la tasca tingui màxima prioritat per assegurar que sempre s'executa en el moment adequat per tal es vegi una velocitat constant en el moviment. Aquesta tasca es la encarregada de canviar la fase dels motors en cada tick, per tant la velocitat d'aquesta determina la velocitat de moviment. S'ha testejat diferents velocitat i s'ha vist que les fases s'han d'incrementar amb períodes que van dels 3 als 10 ms. També es una tasca que ha d'executar se molt ràpid. Tenim dos opcions de disseny:
  - Executar una tasca periòdica de 1 ms i així podem variar el període entre 3 i 10 ms amb una precisió de 1 ms. Aquesta opció implica més carrega de CPU però fa que el període la tasca no es modifiqui. Per tant no cal que dins de la tasca del driver, es cridin a funcions del sistema operatiu.
  - Tenir una tasca amb període variable de entre 3 i 10 ms. D'aquesta manera no cal comptar el ticks per determinar quan modifiquem les fases dels motors i reduïm la carrega de CPU.

  Finalment ens em decidit per controlar la velocitat dels motors directament modificant la periodicitat de la tasca, es a dir, que em donat prioritat a a la carrega de CPU que a mantindre una arquitectura on el driver dels motors no crida a funcions del sistema operatiu.

- **Tasca d'alta prioritat dels drivers.** Aquesta tasca executa la resta de drivers que necessiten executar-se també amb alta prioritat. Els drivers necessiten executar-se ràpidament per tal de que el sistema funcioni fluidament, per aquest motiu el definim amb una prioritat alta i amb un període de 2 ms. Com veurem mes endavant, s'ha definit un tick de 2 ms per poder mostrejar el senyal que ve del sensor de posició de la mà.

- **Tasca d'aplicació.** L'aplicació te prioritat mitjana per eliminació, es a dir que després de veure les necessitats de la resta de tasques em decidit deixar aquesta en una prioritat mitjana. Aquesta tasca s'executa cada 10 ms.

- **Tasca de drivers de baixa prioritat.** Aquesta tasca gestionarà el driver I2C que es comunica amb els sensors d'ultrasons. El driver I2C conté esperes actives, es a dir que no retornen al control al sistema operatiu, i es per aquest motiu que si els sensors d'ultrasons no contesten hi ha el risc de penjar la resta del sistema. Assignant a aquesta tasca una prioritat baixa, ens assegurem que només bloquejarà a les tasques amb menys prioritat. Aquesta tasca s'executarà cada 100 ms.
- **Tasca d'actualització del display.** S'ha vist que el driver del display proporcionar per Texas Instruments [2] es molt lent i es per evitar que bloquegi la resta de tasques s'ha deixat en mínima prioritat. A més a més s'ha vist que necessita un stack més gran que la resta, aixi que mentre la resta de tasques tenen el stack mínim definit en el FreeRTOS [1] de 100 bytes, aquesta tasca te un stack de 300 bytes.

Un cop analitzades les tasques passem a veure el diagrama de blocs del sistema. En la següent figura es veuen els components de programaria que tenim en el projecte.

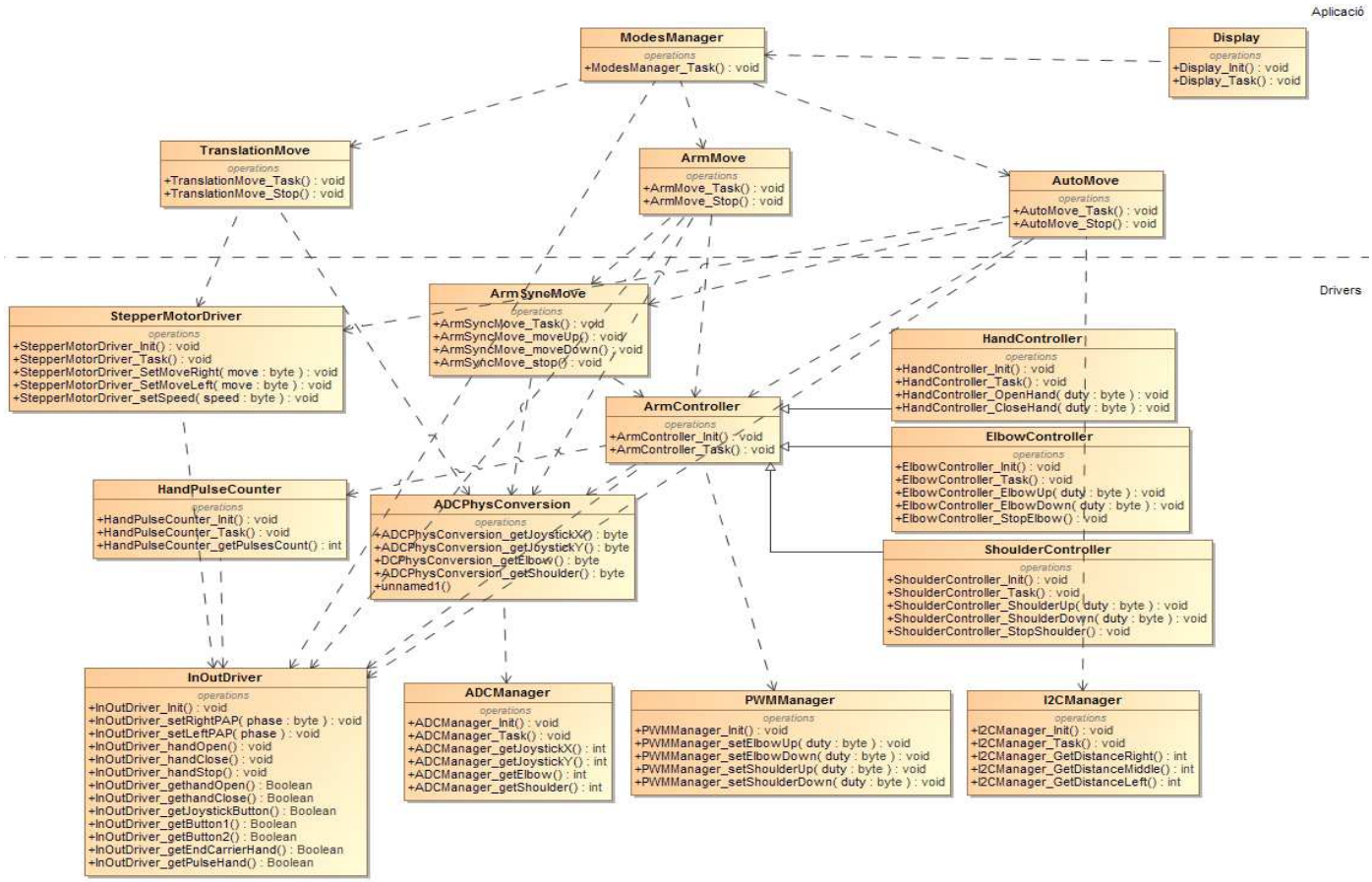


Figura 11. Diagrama de blocs del programari

En la arquitectura s'han fet uns components de baix nivell que comuniquen directament amb els drivers dels perifèrics subministrats pe Texas Instruments [4]. Aquests components fan la capa superior independents del perifèric en si que s'ha fet servir per cada funcionalitat, es a dir que es on decideix a quin port està connectat cada funcionalitat. D'aquesta manera, si volem canviar un pin de sortida per un altra, em de modificar la configuració d'aquests components. A mida que anem pujant en la arquitectura anem afegint funcionalitat augmentant el nivell d'abstracció. S'entrarà en més detall dels components en el següent capítol.

Un cop analitzades les tasques i l'arquitectura general, s'ha analitzat la transferència de dades entre components i la possible utilització de recursos compartits entre components, per tal de definir els protocols de protecció dels recursos. Em vist que els recursos de perifèrics del microcontrolador estan gestionats sempre per una única tasca, per tant, no cal afegir seccions crítiques ni mecanismes per evitar problemes per aquest motiu. En quan a la transferència de dades, si be hi ha dades que es mouen entre diferents tasques, no hi a blocs crítics de dades que s'hagin de gestionar a l'hora, es a dir com un mateix bloc. Per tant tampoc cal afegir mecanismes de sincronització de dades entre diferents tasques.

4 Descripció detallada

4.1 Legobot

En aquest apartat entrarem en els detalls del robot Legobot.

4.1.1. Sensors de posició del braç

El braç disposa de dos sensors de posició per tal de conèixer com estan situats el colze i el ombro. Aquests sensors estan fets amb uns potenciòmetres de 5KΩ creant un divisor de tensió entre 3V3 i massa on el valor de tensió de la sortida, variarà en funció del angle. Per tant, caldrà adaptar aquestes entrades abans de connectar-les als convertidors ADC del microcontrolador. En les següents imatges es veuen del potenciòmetres del colze i el ombro.

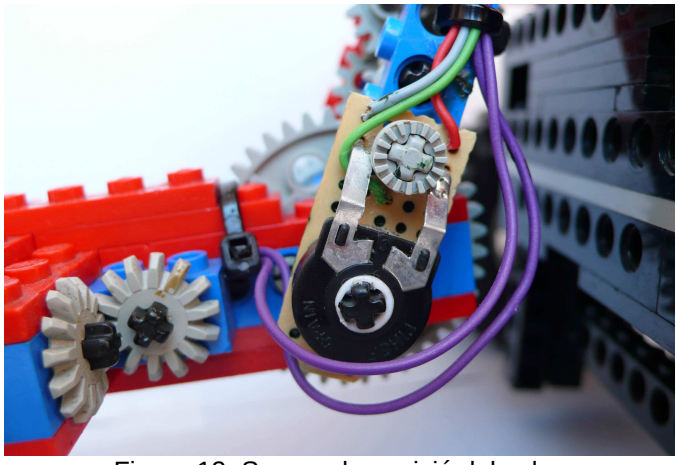


Figura 12. Sensor de posició del colze

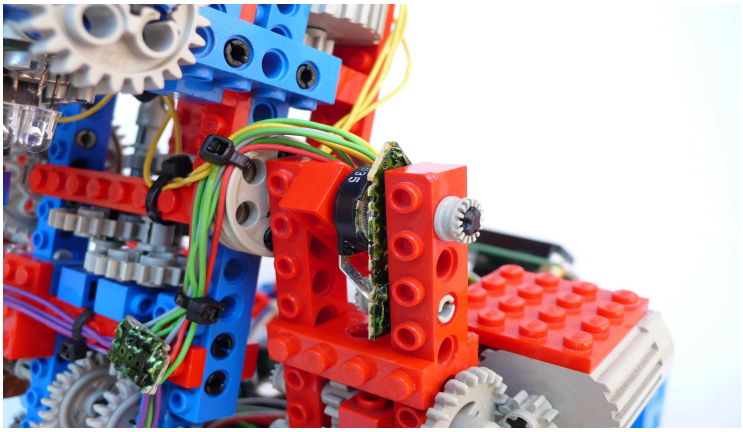


Figura 13. Sensor de posició del colze

4.1.2. Sensor de final de carrera de la mà

La mà disposa de un sensor de fi de carrera per detectar quan aquesta està oberta del tot. La construcció del sensor es un simple fil de coure que toca un altre fil quan la mà s'obra. El robot també disposa del maquinari per adaptar aquest sensor als nivells de tensió del

microcontrolador, por tant no cal fer cap adaptació i es pot connectar directament a una entrada digital del d'aquest. En la següent figura es mostra un detall del sensor.

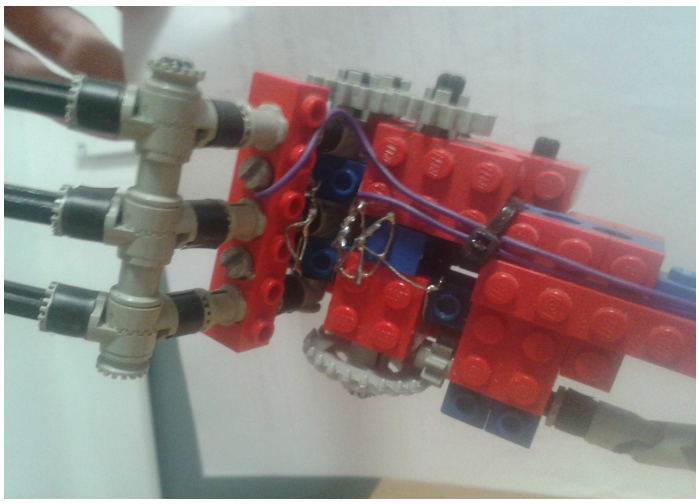


Figura 14. Sensor de final de carrera de la mà

4.1.3. Sensor de posició de la mà

També es possible es possible mesura la posició de la mà a través d'un encoder que genera polsos quan el motor de la mà està en funcionament. Aquest sensor està construït amb un optoacoblador TCST 1000 i una roda amb forats. Quan el sensor passa per el forat genera uns polsos amb una periodicitat d'uns 50 ms i 10 ms a nivell baix.

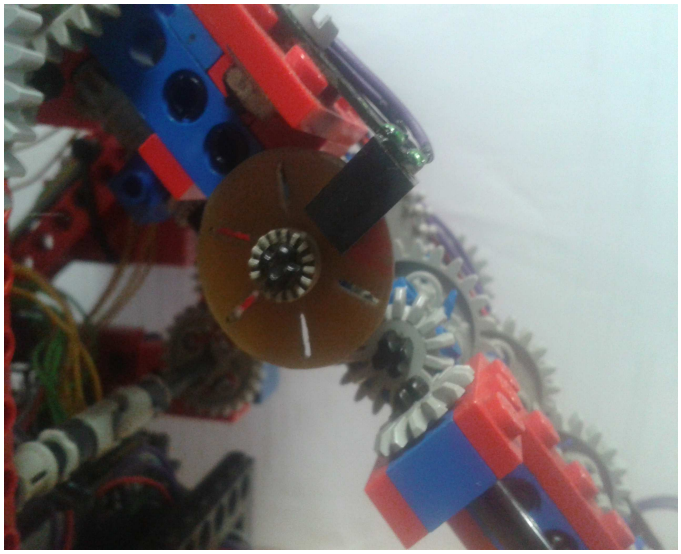


Figura 15. Sensor de posició de la mà

El robot ja te el maquinari necessari per adaptar el senyal d'aquest sensor, per tant la sortida d'aquest pota anar directament connectada a una de les entrades digitals del microcontrolador.

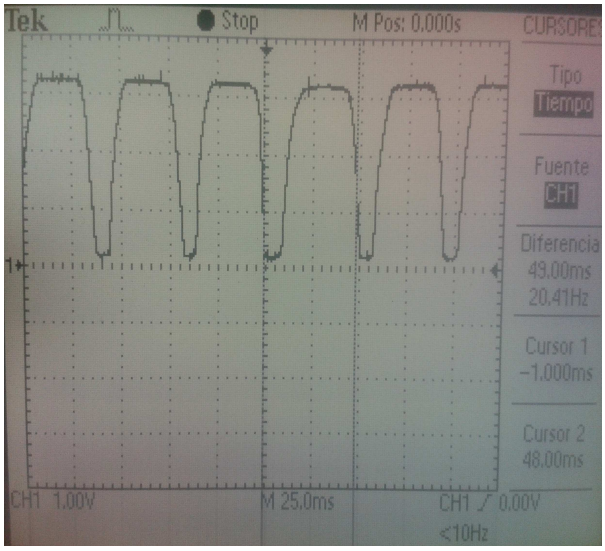


Figura 16. Captura del senyal del sensor de posició de la mà

4.1.4. Sensors de proximitat

El robot te tres sensors de d'ultrasons que mesuren la distancia al objecte més proper. Aquests sensors han d'estar alimentats a 5V i es comuniquen a través de I2C o UART. En aquest cas ja estan configurats per funcionar en mode I2C i tenen les adreces configurades perquè siguin diferents.

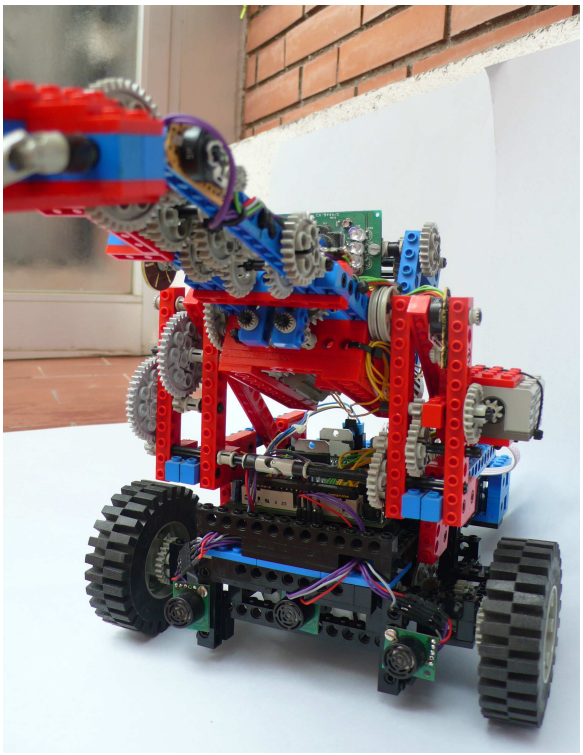


Figura 17. Sensors d'ultrasons muntats al Legobot

Les adreces I2C configurades en els sensors d'ultrasons son:

- Adreça sensor de la dreta: 0xE0
- Adreça sensor del mig: 0xE4
- Adreça sensor de la esquerra: 0xE2

4.1.5. Motors de corrent continua

Els motors de corrents continua que fa servir el robot per moure el braç i la mà son 3 motors 8700 clàssics de Lego Technic que funcionen a 4.5V. Aquest poden controlar la seva velocitat mitjançant controladors PWM.

Figura 18. Motor 8700 de Lego

4.1.6. Motors pas a pas

El robot es pot desplaçar mitjançant dos motors pas a pas model Saia ukd34n04rnz3 que fa servir dos bovines per ser polaritzat. Aquests motors es controlen amb un pont de transistor per cada bovina i en funció de si s'activa una, dos bovines i el sentit de polarització pot girar fent servir vuit fases diferents

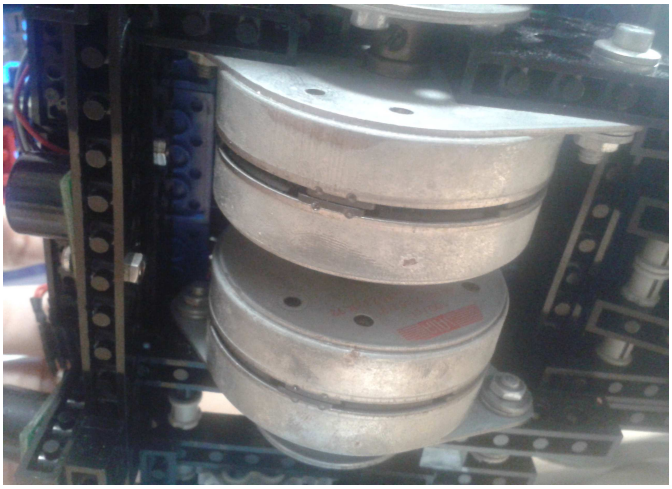
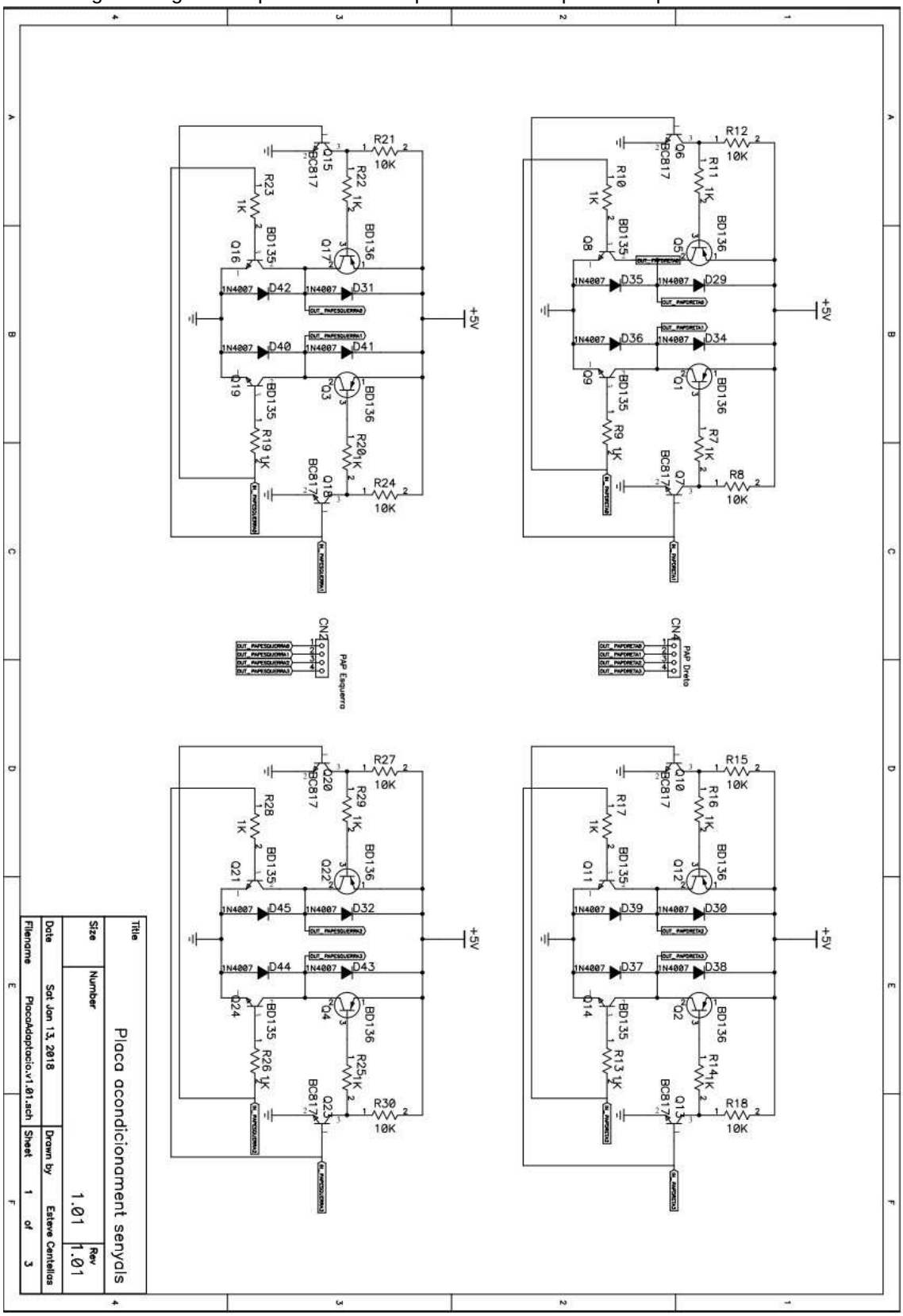


Figura 19. Detall dels motors Saia ukd34n04rnz3

4.2 Maquinari

4.2.1. Esquemàtic de la placa d'expansió

En les següents figures es pot veure els esquemàtics de la placa d'expansió.



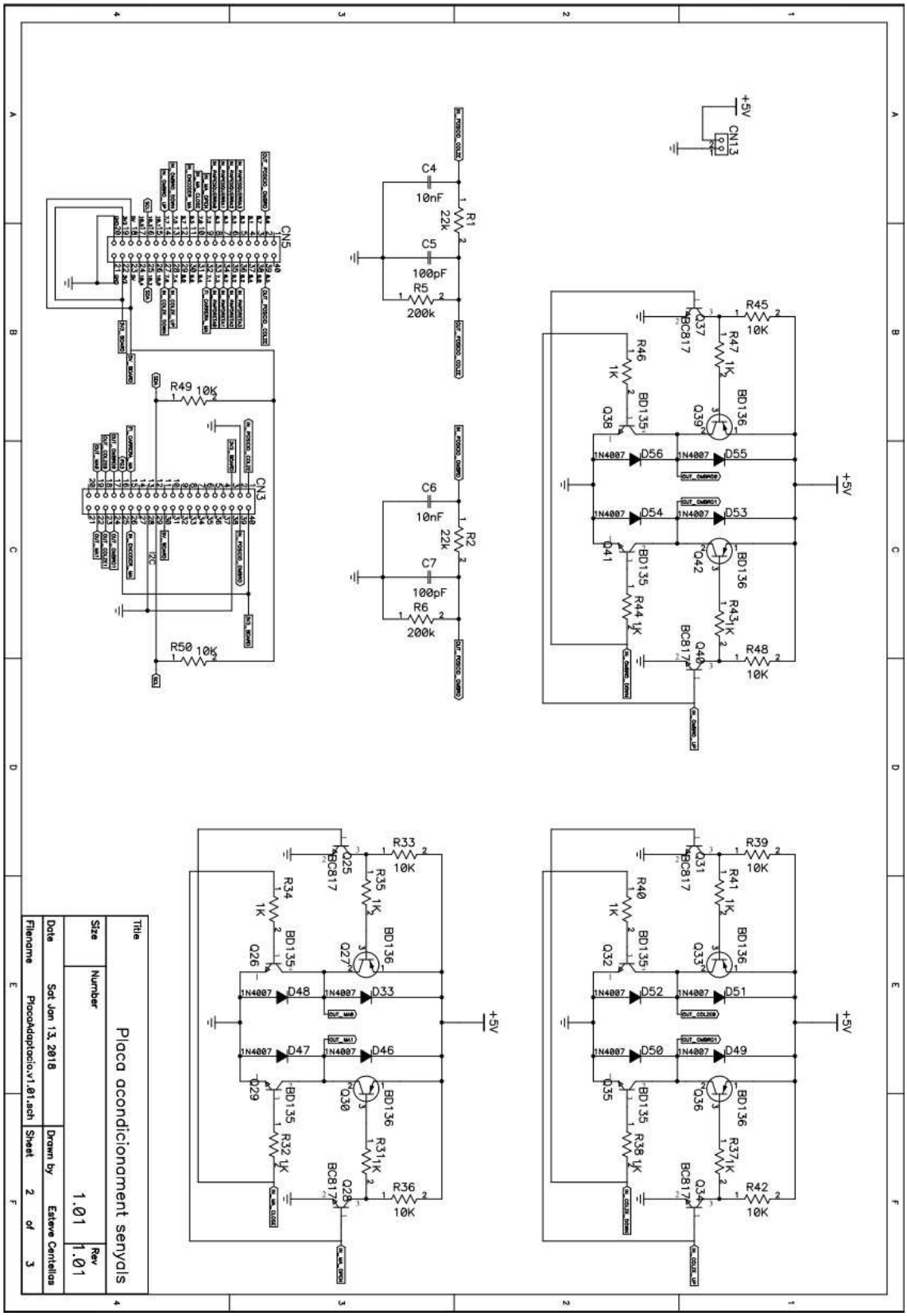


Figura 20. Esquemàtics de la placa d'expansió

4.2.2. Adaptació del sensors del braç

El maquinari d'adaptació de les senyals que venen dels sensors del braç es el següent:

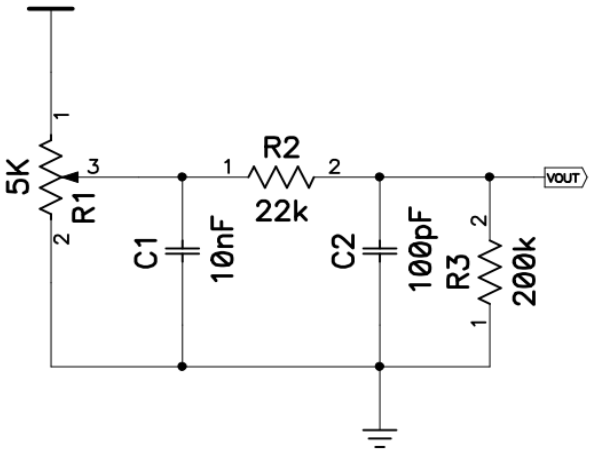


Figura 21. Maquinari d'adaptació dels sensors de posició del braç

Si analitzem el maquinari es pot arribar a la conclusió que la tensió Vout te el valor següent

$$V_{out} = \frac{VCC \cdot (R_1 + R_2)}{\quad} \cdot R_2$$

On x es la resistència entre la massa i la sortida del potenciòmetre.

4.2.3. Ponts de transistors

La estructura dels ponts de transistors es repeteix fins set vegades en els esquemàtic, que son els necessaris per controlar els dos motor pas a pas i tres motors de continua. El esquema de tots aquests ponts es tal i com es veu en la següent figura:

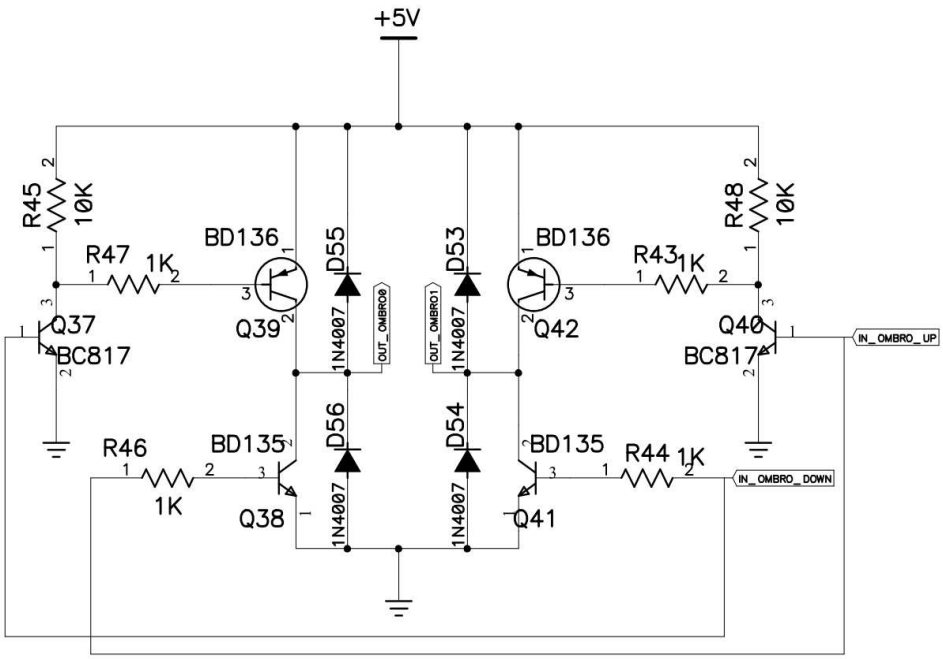


Figura 22. Esquema dels ponts de transistors

Esquema dels ponts de transistors

Em escollit els transistors BD136 per els mig ponts superiors i els BD135 per els inferiors. Aquests transistors suporten corrents de fins 1,5 A, i per tant son aptes per alimentar els motors que com a molt consumeixen 200 mA. Per un altre banda tenim dos transistors BC817 que son transistors de senyals utilitzats per invertir el senyal de control que provenen del microcontrolador. També tenim diodes 1N4007 que serveixen per descarregar les bovines dels motors quan aquesta s'apaguen.

Amb tot això, cada pont es controlat per dos senyals, que cada una fa circular el corrent de sortida en un sentit o l'altre. Es molt important que programari de control no activi mai els dos senyal de control a l'hora, ja que provocaria un curtcircuit entre la tensió s'alimentació i massa.

4.2.4. Connexitat de la placa d'expansió i la placa MSP-EXP432P401R

Per connectar la placa MSP-EXP432P401R [2] i la placa d'expansió s'ha optat per fer servir el connector Edge Connector (J5) on es pot accedir a tots els pins que no es fan servir. Així es deixaran lliures els pins del connectors J1, J2, J3 i J4 per connectar la Educational BoosterPack MKII [4].

Per assignar els ports em començat mirant els que tenen alguna funcionalitat especial. Així la assignació s'ha fet en el ordre següent:

- **Bus I2C dels sensors d'ultrasons.** En el port Edge Connector de la placa només hi accessible el bus I2C UCB3 que es connecta a través dels ports 10.2 i 10.3 del microcontrolador. Aquests son els pins 29 i 30 del connector.

- **Entrades analògiques.** Hem de connectar les entrades analògiques provinents dels sensors de posició del braç. Hem connectat:
  - El sensor de ombro en l'entrada A19 corresponent a port 8.6 i al pin 1 del connector
  - El sensor del colze a l'entrada A20 corresponent al port 8.5 i al pin 2 del connector.

  En aquest cas teníem més possibles entrades i em escollit les primeres que hi havia al connector.

- **Sortides PWM.** El connector no disposa de gaires sortides PWM així que hem aprofitat les que hi havia. Amb tot això hem connectat:
  - La sortida PWM per moure el colze a dalt al PM_TA1.4 corresponent al port 7.4 i pin 24 del connector
  - La sortida PWM per moure el colze a baix al PM_TA1.2 corresponent al port 7.6 i pin 26 del connector
  - La sortida PWM per moure el ombro a dalt al PM_TA1.1 corresponent al port 7.7 i pin 25 del connector.
  - La sortida PWM per moure el ombro a baix al PM_TA1.3 corresponent al port 7.5 i pin 23 del connector.

- **Sortides digitals.** Aquestes sortides ara ja es poden connectar a qualsevol del pins restants i ara s'han escollit aleatòriament agrupant-los i evitant fer servir ports que poguessin ser útils per futures expansions, com per exemple altres entrades analògiques. Els pins escollits son:
  - La sortida 0 de motor pas a pas de l'esquerra al port 6.3 corresponent al pin 13 del connector.
  - La sortida 1 de motor pas a pas de l'esquerra al port 9.3 corresponent al pin 11 del connector.

- La sortida 2 de motor pas a pas de l'esquerra al port 5.3 corresponent al pin 9 del connector.
- La sortida 3 de motor pas a pas de l'esquerra al port 8.3 corresponent al pin 7 del connector.
- La sortida 0 de motor pas a pas de la dreta al port 7.3 corresponent al pin 14 del connector.
- La sortida 1 de motor pas a pas de la dreta al port 6.2 corresponent al pin 12 del connector.
- La sortida 2 de motor pas a pas de la dreta al port 9.2 corresponent al pin 10 del connector.
- La sortida 3 de motor pas a pas de la dreta al port 8.2 corresponent al pin 8 del connector.
- La sortida per obrir la mà al port 7.2 corresponent al pin 15 del connector.
- La sortida per tancar la mà al port 7.0 corresponent al pin 17 del connector.

- **Entrades digitals.** Aquestes entrades es poden connectar a qualsevol del pins restants i s'han escollit aleatòriament. Els pins escollits son:
  - La entrada del sensor de final de carrera de la mà al port 7.1 corresponent al pin 16 del connector.
  - La entrada del sensor de posició de la mà al port 9.5 corresponent al pin 19 del connector.

4.3 Programari

4.3.1. Components bàsics de programari

En primer lloc analitzarem els components que fan les funcions bàsiques del robot, es a dir els que estan dins de drivers, fent una descripció del que fa cada un. Aquests components conformen el paquet de control del robot i son el punt de partida de noves implementacions.

4.3.1.1. InOutDriver

Aquest component abstreu la resta del codi de en quins pins estan connectats les entrades i sortides digitals. Fa servir les llibreries de Texas Instruments [2] per escriure i llegir dels ports i a partir d'aquí els ports s'accedeixen a partir del seu nom fent que el programador no hagi de saber a quin pin està connectada cada funcionalitat. Centralitzar el accés a les funcions de Texas Instruments de gestió dels ports digitals.

4.3.1.2. ADCManager

Fa servir les llibreries de Texas Instruments per gestionar els convertidors ADC. Té una tasca periòdica que s'executa dins de la tasca de alta prioritat driverMainTask. Com a mínim s'ha d'executar cada 10ms però es fa cada 2ms. Es el únic component que crida a les funcions de gestió dels ADC de Texas Instruments i fa la resta del codi independent de quin pin físic està connectat cada por analògic. A partir d'aquí s'accedeix a les lectures analògiques per el seu nom.

4.3.1.3. PWMManager

Utilitza a les funcions de Texas Instruments per configurar i accedir als timers que gestionen les sortides PWM. Al igual que els components anteriors fa la resta de components independents dels pins on estan connectades les sortides PWM i partir d'aquí s'accedeix a aquestes per nom.

4.3.1.4. I2CManager

Gestiona les comunicacions amb els sensors d'ultrasons utilitzant les funcions del driver de I2C de Texas Instruments. Degut a que la seva implementació te esperes actives, i per això s'ha decidit que es cridi des de la tasca de baixa prioritat driverLowPriTask. D'aquesta manera s'evita bloquejar a la resta del sistema. Aquest component fa que la resta del codi sigui independent del origen de les lectures de les distancies.

Per implementar aquest component cal conèixer com funcionen els sensors d'ultrasons. En primer lloc tenim que les adreces del tres sensors estan ja configurades i aquestes son:

- Adreça sensor de la dreta: 0xE0
- Adreça sensor del mig: 0xE4

- Adreça sensor de la esquerra: 0xE2

Per fer una mesure primer s'ha d'enviar l'ordre de començar-la, i per ferla en centímetres cal escriure el valor 0x51 en el registre 0. Un cop enviada l'ordre, el driver ha d'esperar un mínim de 70ms i llegir els registres 2 i 3 on estaran els valors de la mesura. En les dos figures següents tenim imatges de les captures de les trames I2C, la primera de la ordre de començar la mesura i la segona de la lectura d'aquesta.

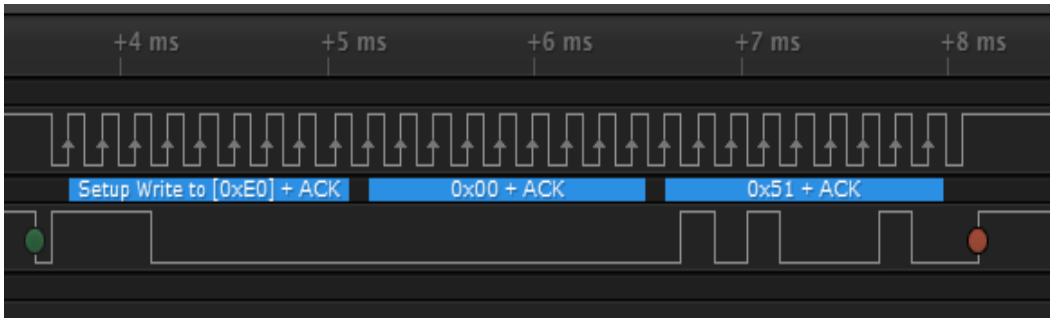


Figura 23. Ordre de començar mesura

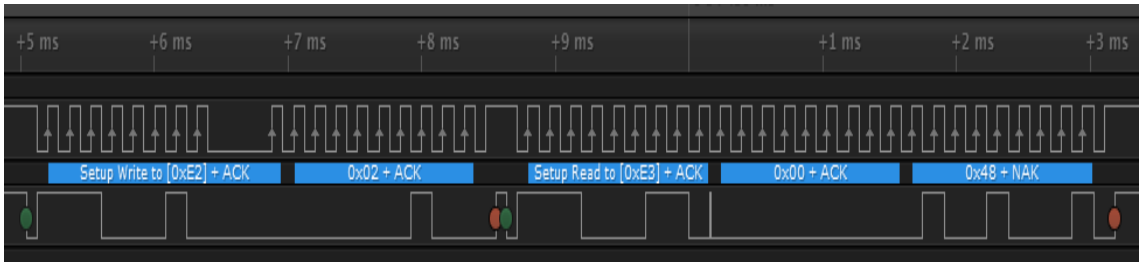


Figura 24. Lectura d'una mesura

La tasca periòdica d'aquest component envia l'ordre de llegir un dels sensors, espera 100ms, llegeix el resultat de la operació i tot seguit fa les lectures dels altres sensors. Per tant triga al voltant de 300ms en mesurar les distancies dels 3 sensors.

4.3.1.5. HandPulseCounter

Compta els polsos del sensor de posició de la mà per determinar la seva posició. Llegeix el port del sensor a través del component InOutDriver i es el únic component que accedeix al port del sensor de posició. Te una tasca periòdica que s'executa dins de la tasca de alta prioritat driverMainTask i que s'encarrega de fer el mostreig del sensor. Com que els polsos que venen del sensor tenen una periodicitat d'uns 50ms i estan 10ms en estat baix, em escollit un temps de mostreig de 2ms. Aquest es el motiu per el qual la tasca driverMainTask s'executa cada 2ms.

4.3.1.6. ADCPhysConversion

Converteix les lectures digitals de ADCManager a magnituds físiques suposant que son lineals. Es podria ampliar aquest component implementant altres tipus de relacions entre els valors dels convertidors i les magnituds físiques. Actualment aquest component converteix les lectures de la posició del braç en graus i la posició del joystick al interval -100% 100%. Per configurar el convertidor tenim la estructura ADCPhysConversion on es defineix de quina variable es llegeix el valor del ADC, el pendent i el offset de la conversió.

```
T_ADCPhysConversion_item
ADCPhysConversion_Vector[ADCPHYSCONVERSION_CHANNELS_NUM] = {
        {
             &ADCManager_getJoystickX(),
             ADCPHYSCONVERSION_SLOPE_NUM_JOYSTICKX,
             ADCPHYSCONVERSION_SLOPE_DEN_JOYSTICKX,
             ADCPHYSCONVERSION_OFFSET_JOYSTICKX
        },
        {
             &ADCManager_getJoystickY(),
             ADCPHYSCONVERSION_SLOPE_NUM_JOYSTICKY,
             ADCPHYSCONVERSION_SLOPE_DEN_JOYSTICKY,
             ADCPHYSCONVERSION_OFFSET_JOYSTICKY
        },
        {
             &ADCManager_getElbow(),
             ADCPHYSCONVERSION_SLOPE_NUM_ELBOW,
             ADCPHYSCONVERSION_SLOPE_DEN_ELBOW,
             ADCPHYSCONVERSION_OFFSET_ELBOW
        },
        {
             &ADCManager_getShoulder(),
             ADCPHYSCONVERSION_SLOPE_NUM_SHOULDER,
             ADCPHYSCONVERSION_SLOPE_DEN_SHOULDER,
             ADCPHYSCONVERSION_OFFSET_SHOULDER
        }
    };
```

Per evitar fer operacions en coma flotant s'ha optat per definir els pendents de les conversions amb un numerador i denominador. D'aquesta manera es fan les conversions multiplicant el valor del ADC per el numerador i després es divideix per el denominador.

Un altre decisió que s'ha pres en la utilització d'aquest component, es que s'ha aproximat el calcul dels angles dels sensors dels braç a una relació lineal amb els valors dels convertidors, malgrat que em vist en el capítol 4.2.2 que no ho es. Com que tampoc necessitem una precisió molt alta en les mesures d'aquests angles, sinó que només volem tenir una idea per obtenir uns moviments mes síncrons, em tingut prou amb aquesta aproximació per obtindre un resultat satisfactori.

El procediment per calibrar les conversions, ha estat mirar els valors del ADC en les posicions extrem de l'escala i calcular el pendent i el offset. Aquests càlcul es fan en temps de compilació amb unes constants definides a partir de les mesures. Aquí tenim un exemple de la configuració d'un dels convertidors.

```
/**
 * Elbow conversion configuration
 */
#define ADCPHYSCONVERSION_MINADC_ELBOW 1470
#define ADCPHYSCONVERSION_MAXADC_ELBOW 13510
#define ADCPHYSCONVERSION_MINPHYS_ELBOW 90
#define ADCPHYSCONVERSION_MAXPHYS_ELBOW 315

#define ADCPHYSCONVERSION_SLOPE_NUM_ELBOW
(ADCPHYSCONVERSION_MAXPHYS_ELBOW - ADCPHYSCONVERSION_MINPHYS_ELBOW)
#define ADCPHYSCONVERSION_SLOPE_DEN_ELBOW
(ADCPHYSCONVERSION_MAXADC_ELBOW - ADCPHYSCONVERSION_MINADC_ELBOW)
#define ADCPHYSCONVERSION_OFFSET_ELBOW
(ADCPHYSCONVERSION_MINPHYS_ELBOW - ((ADCPHYSCONVERSION_SLOPE_NUM_ELBOW * \
ADCPHYSCONVERSION_MINADC_ELBOW)/ADCPHYSCONVERSION_SLOPE_DEN_ELBOW))
```

4.3.1.7. StepperMotorDriver

Aquest component gestiona els ponts de transistors que controlen els motors pas a pas per obtenir moviment en aquests. Per obtenir aquest moviment es va polaritzant les dos bovines del motor en les dues direccions tal i com es mostra en la figura

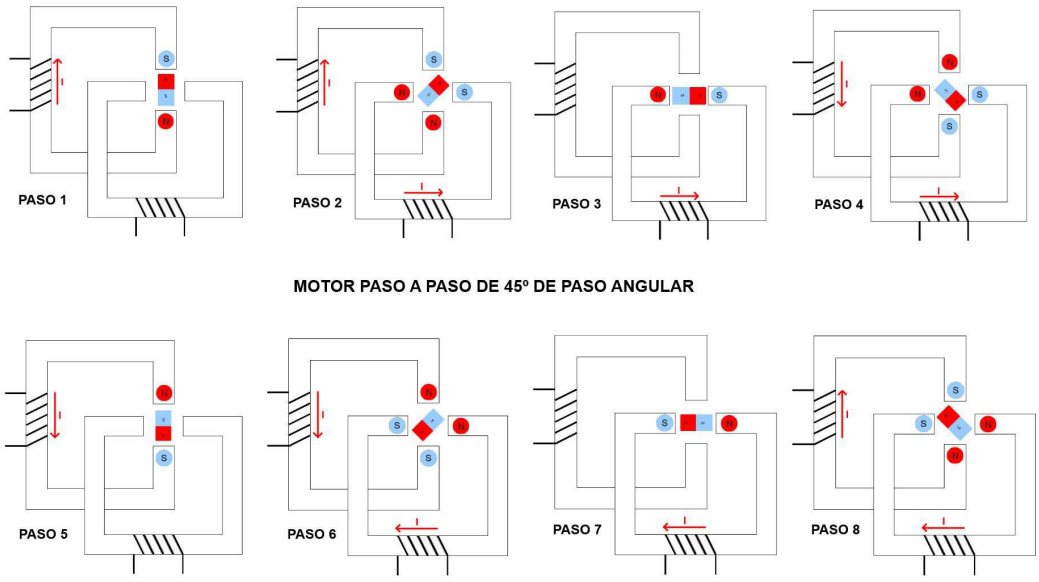


Figura 25. Funcionament de motor pas a pas

El component StepperMotorDriver permet configurar el funcionament per que funcioni amb 4 o 8 fases, fent-ho funcionar amb les fases que només s'activa una bovina o amb totes. Es pot canviar la configuració definint la constat STEPPERMOTORDRIVER_4PHASES. Aquest component te funcions per fer anar els motors en les dos direccions i una tasca periòdica que

varia la seva periodicitat entre en 3 i 30ms, en funció de la velocitat que volem donar-li al moviment. Per evitar que el seu moviment pugui ser irregular s'executa en una tasca de màxima prioritat de sistema operatiu,

4.3.1.8. ArmController

Aquest component es el encarregat de controlar els moviments del braç parant-los en els finals de carrera o quan es troben un obstacle. Esta construït amb una maquina de estat que d'instància per cada un dels tres motors del braç, es a dir colze, ombro i mà.

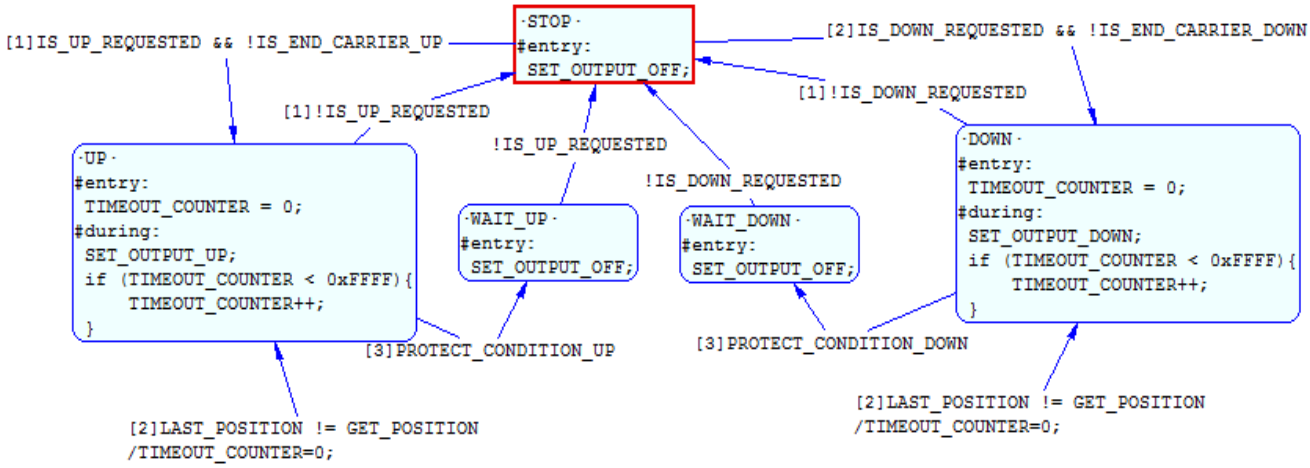


Figura 26. Màquina d'estats de ArmController

La màquina d'estats te com a paràmetres el estat actual i dos estructures que apunten a una serie de funcions i variables que aquesta necessita. Cada instancia guarda el seu propi estat i proporciona aquestes variables i funcions. Anem a veure el detall de les macros definides en la màquina d'estats. En primer lloc tenim una serie d'accions con son encendre o apagar les sortides o llegir si s'ha demanat activar alguna cosa. Les accions criden a les funcions definides dins de cada instancia i estan definides així.

```
/* Actions */
#define SET_OUTPUT_OFF          (ArmController_data_const->set_output_off())
#define SET_OUTPUT_UP           (ArmController_data_const->set_output_up())
#define SET_OUTPUT_DOWN         (ArmController_data_const->set_output_down())
#define IS_UP_REQUESTED         (ArmController_data_const->is_up_requested())
#define IS_DOWN_REQUESTED       (ArmController_data_const-
>is_down_requested())
#define IS_OFF_REQUESTED        (ArmController_data_const-
>is_off_requested())
#define IS_END_CARRIER_UP       (ArmController_data_const-
>is_end_carrier_up())
#define IS_END_CARRIER_DOWN     (ArmController_data_const-
>is_end_carrier_down())
#define GET_POSITION            (ArmController_data_const->get_position())
```

Després tenim les variables necessàries que es fan servir per comptar temps i verificar si el sistema s'ha mogut

```
/* Variables */
#define TIMEOUT_COUNTER         ArmController_data_var->timeout_counter
#define TIME_TO_STOP            ArmController_data_var->time_to_stop
#define LAST_POSITION           ArmController_data_var->last_position
```

Per últim tenim les condicions per determinar si ha d'executar certes accions com parar algun motor per haver trobat un objecte.

```
/* Conditions */
#define PROTECT_CONDITION_UP (TIMEOUT_COUNTER>=TIME_TO_STOP && TIME_TO_STOP!
=0xFFFF) || IS_END_CARRIER_UP
#define PROTECT_CONDITION_DOWN (TIMEOUT_COUNTER>=TIME_TO_STOP &&
TIME_TO_STOP!=0xFFFF) || IS_END_CARRIER_DOWN
```

En configurar algun dels motor em de definir les estructures que necessita la màquina d'estats, es ha dir el estat en si, les variables i funcions. Aquí es por veure un exemple de configuració.

```
static T_ArmController_instance ElbowController_instance = {
    (T_bit) 0,      /* fsm_father_entry_flag */
    (unsigned char) 0 /* fsm_father_state */
};


static T_ArmController_data_var ElbowController_data_var = {
    (uint16_t) 0,   /* timeout_counter */
    (uint16_t) 0,   /* time_to_stop */
    (int16_t) 0     /* last position */
};


static const T_ArmController_data_const ElbowController_data_const = {
    ElbowController_set_output_off,
    ElbowController_set_output_up,
    ElbowController_set_output_down,
    ElbowController_is_up_requested,
    ElbowController_is_down_requested,
    ElbowController_is_off_requested,
    ElbowController_is_end_carrier_up,
    ElbowController_is_end_carrier_down,
    ElbowController_get_position
};
```

Veiem també un exemple de configuració de les funcions.

```
static void ElbowController_set_output_off(void){
    PWMManager_setElbowUp(0);
    PWMManager_setElbowDown(0);
}

static void ElbowController_set_output_up(void){
    PWMManager_setElbowUp(ElbowController_duty);
}

static void ElbowController_set_output_down(void){
```

```
    PWMManager_setElbowDown(ElbowController_duty);
}

static T_bit ElbowController_is_up_requested(void){
    return (ElbowController_targetStatus == ELBOWCONTROLLER_UP);
}

static T_bit ElbowController_is_down_requested(void){
    return (ElbowController_targetStatus == ELBOWCONTROLLER_DOWN);
}

static T_bit ElbowController_is_off_requested(void){
    return (ElbowController_targetStatus == ELBOWCONTROLLER_STOP);
}

static T_bit ElbowController_is_end_carrier_up(void){
    return (ELBOWCONTROLLER_ENDCARRIER_UP_CONDITION);
}

static T_bit ElbowController_is_end_carrier_down(void){
    return (ELBOWCONTROLLER_ENDCARRIER_DOWN_CONDITION);
}

static int16_t ElbowController_get_position(void){
    return (ADCPhysConversion_getElbow());
}
```

Un altre punt a tenir en compte en la configuració d'aquest component, es que la variable time_to_stop determina el temps que ha d'estar la variable que determina la posició sense canviar de valor per protegir el motor o parar-lo. Si aquesta variable te el valor 0xFFFF la protecció està deshabilitada.

La tasca principal d'aquest component s'executa dins de la tasca de alta prioritat driverMainTask cada 2ms. També cal notar que es el únic component que hauria de tenir accés als controls dels motors dels braç i mà.

4.3.1.9. ArmSyncMove

Aquest component mou el braç a munt i abaix mantenint el avant braç paral·lel al terra. Això o fa controlant la velocitat de moviment dels motors del colze i ombro.

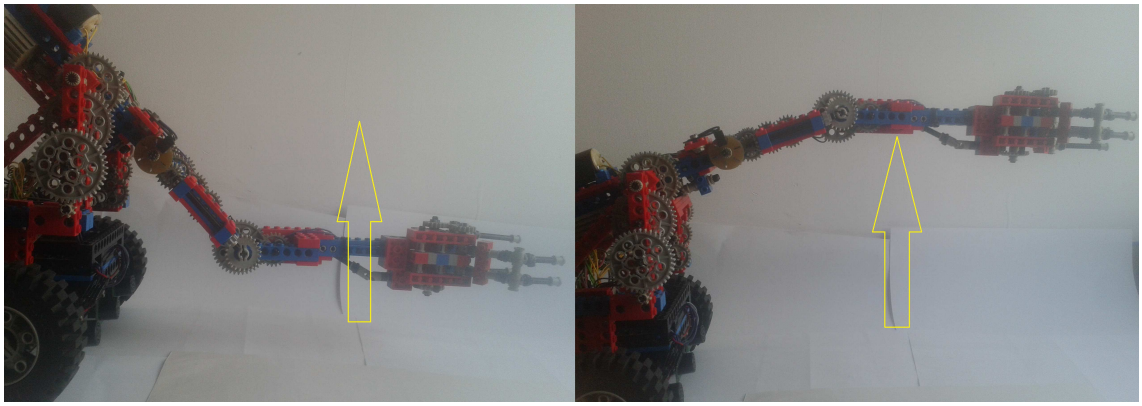


Figura 27. Moviment sincronitzat

El seu funcionament es basa en una màquina d'estats que s'executa dins de la tasca de alta prioritat driverMainTask cada 2ms. La màquina d'estats es la següent.

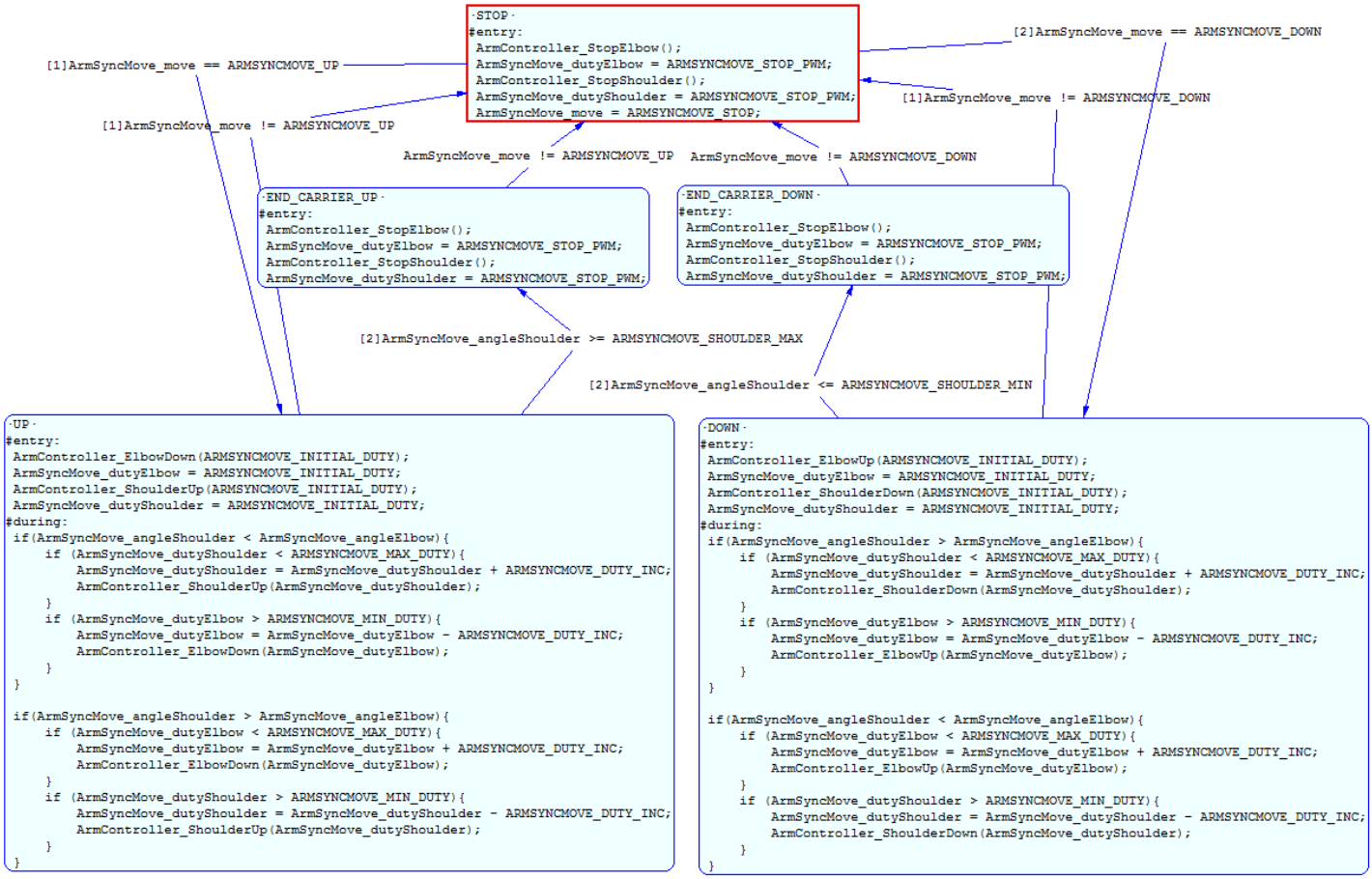


Figura 28. Maquina d'estat de ArmSyncMove

Quan se li demana que mogui el braç, la maquina d'estats ajusta la velocitat dels motors modificant el duty cycle d'aquest per fer que els angles del ombro i el colze es mantinguin amb una diferencia de 90º

Aquí tenim una il·lustració de un digrama de seqüencia on es veu com es criden les funcions per controlar el moviment síncron. ArmSyncMove rep una ordre de moviment i aquest ajusta el PWM dels controladors del colze i ombro en funció de les lectures dels angles provinents de ADCPhysConversion.

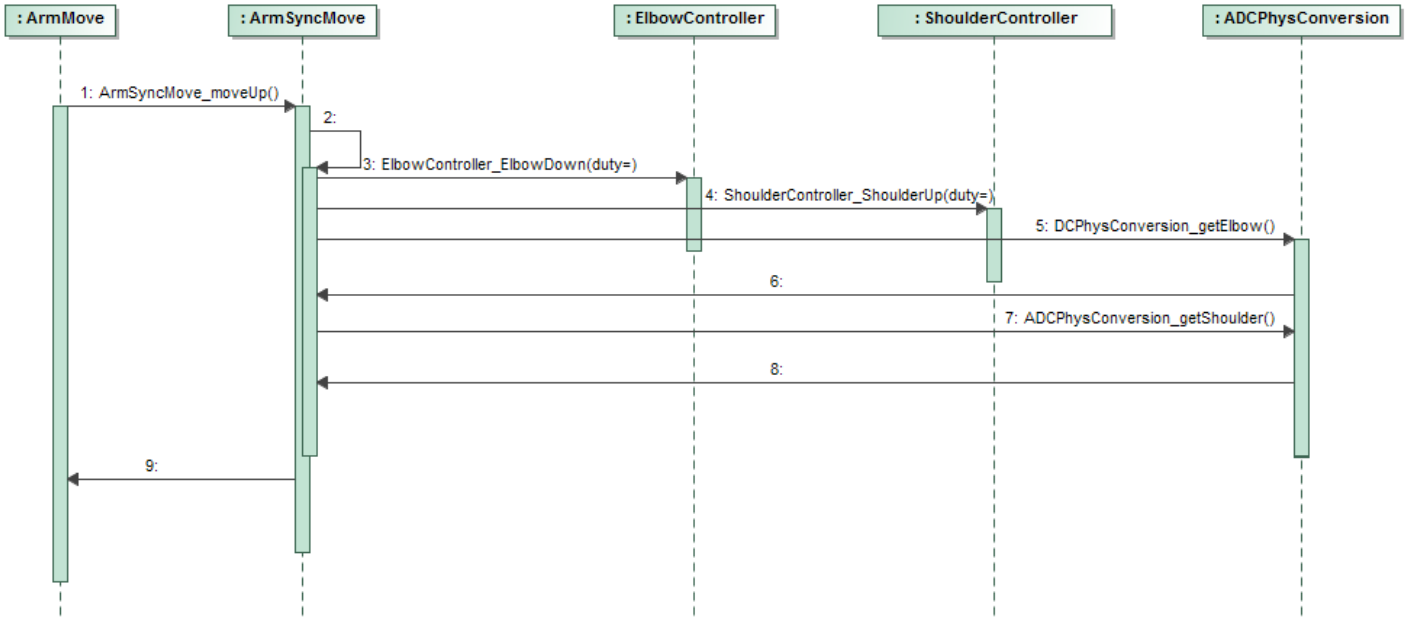


Figura 29. Diagrama se seqüencia d'una ordre a ArmSyncMove

4.3.2. Components de l'aplicació

Un cop definit tot el que seria el programari bàsic que controla el robot passarem a analitzar l'aplicació. Per aquest projecte s'ha definit una aplicació bàsica on fem sevir el joystick i els botons del Educational BoosterPack MKII [4] per enviar ordres al robot i el display per mostrar informació del sistema.

Qualsevol aplicació que s'implementi només hauria de tenir accés als següents components:

- StepperMotorDriver: Per enviar les ordres de moviment als motors pas a pas.
- ArmSyncMove: Per si es vol moure el braç sincrònicament.
- ArmController: Per moure els motors del braç i mà.
- ADCPhysConversion: Per si es vol conèixer el valor de alguna mesura analògica.
- HandPulseCounter: Per conèixer la posició de la mà.
- I2CManager: Per conèixer la distancia els objectes propers.
- InOutDriver: Només per llegir el estat del botons.

4.3.2.1. ModesManager

Aquest component llegeix el estat del botó del joystick i decideix en quin mode ha d'estar funcionant el robot. Es una màquina d'estats molt simple que segons el mode crida a unes funcions o a unes altres

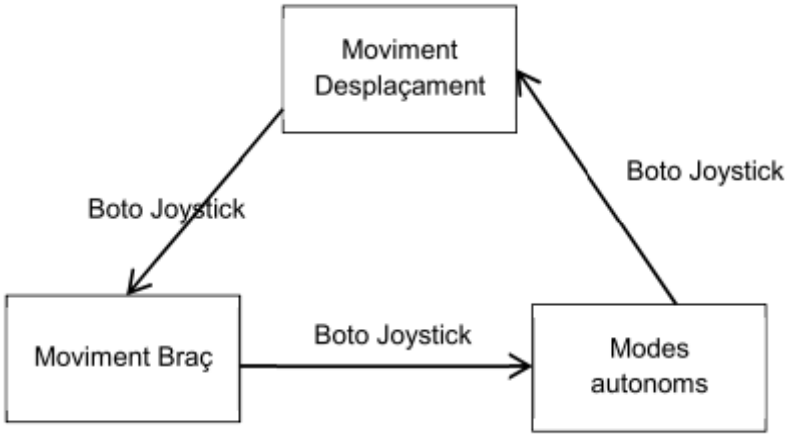


Figura 30. Màquina d'estats de ModesManager

4.3.2.2. TranslationMode

Aquest mode fa moure els motors pas a pas per tal de desplaçar el robot en funció de les ordres rebudes per el joystick. Segons el grau de pressió del joystick regula la velocitat de moviment. El diagrama de flux d'aquest mode es el següent:

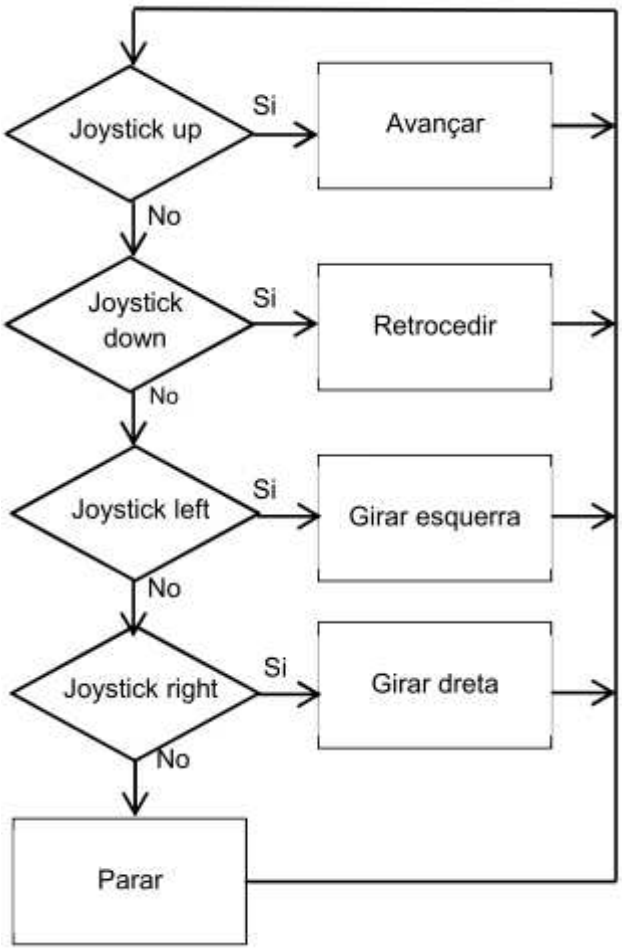


Figura 31. Diagrama de flux de TranslationMode

4.3.2.3. ArmMode

Aquest mode fa moure el braç i la mà. Amb el joystick a dalt i baix es mou tot el braç sincrònicament mantenint el avant braç paral·lel al terra, amb el joystick als costat mou el motor del colze i amb els botons 1 i 2 obre i tanca la mà. El diagrama de flux d'aquest mode es el següent:

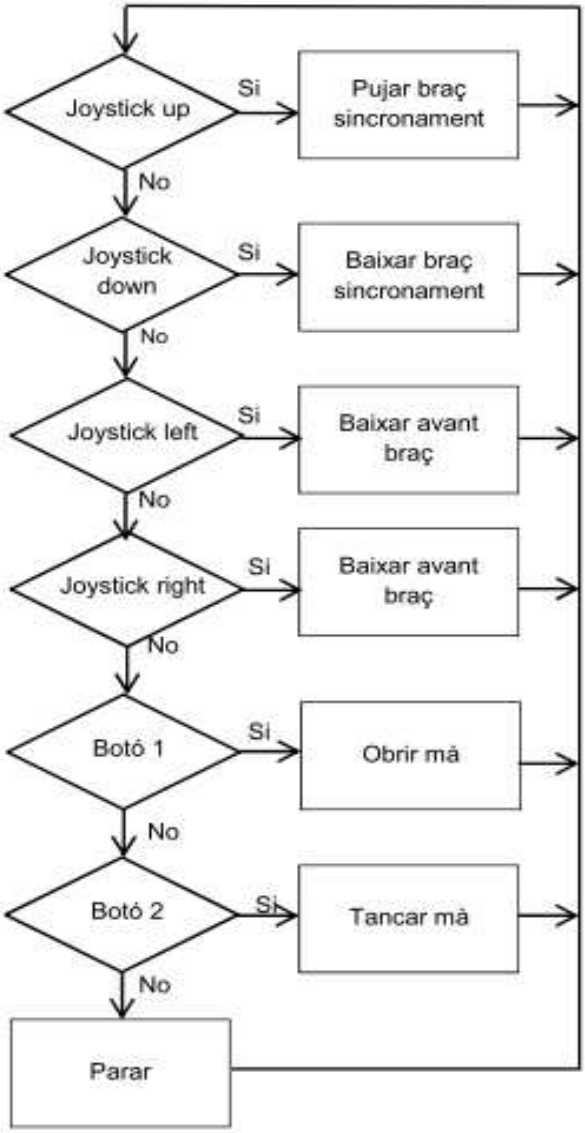


Figura 32. Diagrama de flux de ArmMode

4.3.2.4. AutoMode

Por últim tenim un mode que executa accions autònomes. La idea inicial era que tingués un menú on el usuari pogués escollir el mode amb el Joystick, però degut a la falta de temps només s'ha implement un mode únic que busca un objecte situat davant i l'agafa. El programari bàsic d'aquest projecte està pensat per poder desenvolupar nous modes autònoms d'una manera sencilla.

4.3.2.5. Display

Aquest component simplement llegeix el mode actual i actualitza el display amb les instruccions de com fer anar el mode. En les següents imatges es veuen els missatges que apareixen en cada mode.

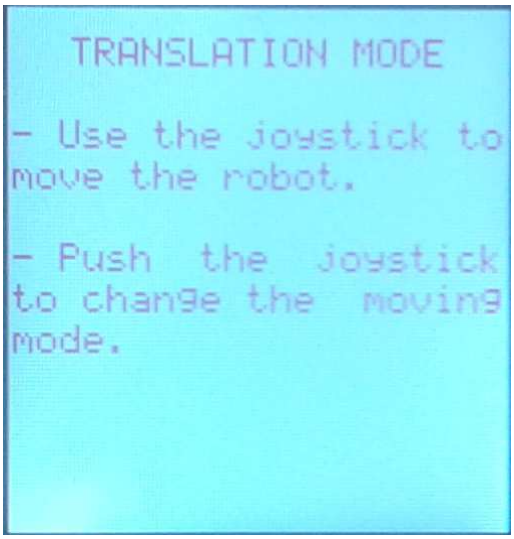


Figura 33. Display en TranslationMode

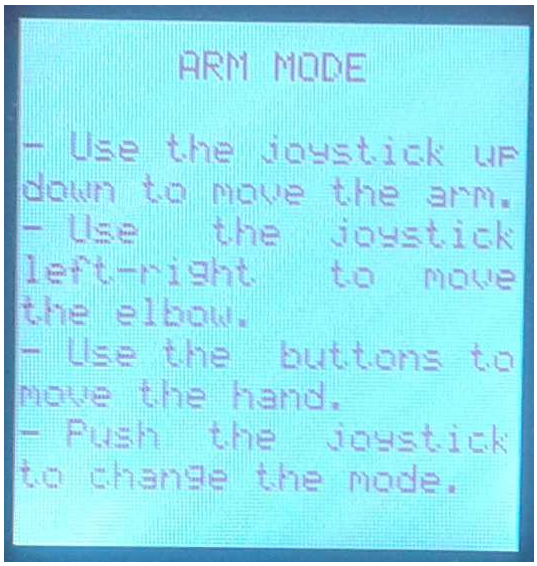


Figura 34. Display en ArmMode

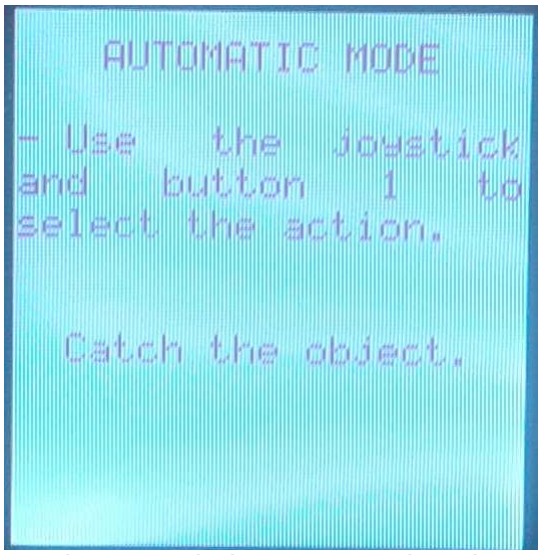


Figura 35. Display en AutomaticMode

5 Viabilitat tècnica

5.1 Estudi de viabilitat

Per veure la viabilitat del projecte primer anem a veure quins recursos del microcontrolador s'han fet servir. Els recursos de perifèrics utilitzats han estat els següents:

- 4 entrades analògiques
- 15 ports digitals
- 1 port I2C
- 4 ports PWM

Com ja em vist en capítols anteriors el microcontrolador te perifèrics de sobres per la implementació i a més a més estan disponibles en els connectors. Per tant des de el punt de vista dels perifèrics el projecte es viable.

Si ens fixem ara en el map file de la compilació del projecte trobem la següent utilització de la memòria:

```
         name            origin    length      used     unused   attr    fill
----------------------  --------  ---------  --------  --------  ----  --------
  MAIN                  00000000   00040000  00013026  0002cfda  R  X
  INFO                  00200000   00004000  00000000  00004000  R  X
  SRAM_CODE             01000000   00010000  00000000  00010000  RW X
  SRAM_DATA             20000000   00010000  0000d7ba  00002846  RW
```

Per tant em fet servir un 30% de la memòria flash i un 84% de la memòria RAM. Per tant encara tenim molts recursos de memòria disponible. El que si que veiem es que el sistema operatiu fa servir casi la totalitat del la memòria RAM, i per tant hauríem de estudiar si es pot reduir aquest ús. Amb tot això arribem a la conclusió que el projecte es viable des de el punt de vista de recursos de memòria.

El següent punt a analitza es la carrega de CPU que tenim al sistema. Per mesurar-la s'ha fet servir un pin per cada tasca que canvia d'estat cada cop que comença a executar-la i quan acaba. En la següent figura es veuen les tasques ordenades per prioritat de mes a menys i la seva utilització de la CPU. Les tasques son aquestes:

- **Tasca del driver dels motors pas a pas.** La seva execució es veu en el canal 2 (vermell) de l'analitzador lògic, triga 178us en executar-se i es crida en intervals que van dels 3 als 30ms. La seva carrega de CPU oscil·la entre el 0,6% i el 6%

- **Tasca d'alta prioritat dels drivers.** Aquesta tasca es veu en el canal 3 (taronja) de l'analitzador lògic i triga en executar-se 522us cada 2ms. Per tant la seva carrega de CPU es del 26%
- **Tasca d'aplicació.** Es pot veure aquesta tasca en el canal 4 (grog) de l'analitzador lògic, s'executa en 145us cada 10ms. La seva carrega de CPU es del 1,5%.
- **Tasca de drivers de baixa prioritat.** Aquesta tasca es visualitza en el canal 5 (verd) de l'analitzador lògic i triga 12ms cada 100ms. La llarga durada d'aquesta tasca es deu a les esperes actives que te el driver I2C. De totes formes com que te prioritat baixa no provoca bloqueig del sistema. Amb tot això la seva carrega de CPU es del 12%
- **Tasca d'actualització del display.** Per últim tenim la tasca del display en el canal 6 (blau) pero que només s'activa quan ha d'actualitzar el display, per tant no la tindrem en compta a l'hora de calcular la carrega de CPU.

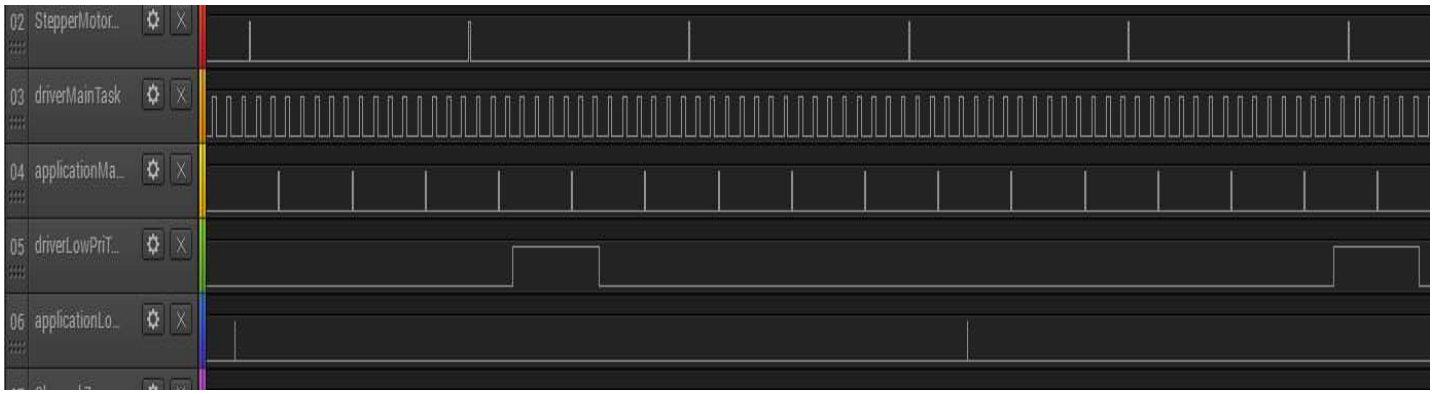


Figura 36. Carrega de CPU

Si calculem la carrega de CPU total trobem que està al voltant de 6%+26%+1,5%+12%=45,5%. Així encara tenim molt marge per afegir més funcionalitat i a més a més podem reduir-la un 10% millorant la tasca que gestiona el I2C. Amb tot això trobem que el projecte es viable per carrega de CPU.

5.2 Estudi de dels punts dèbils i millores

El principal punt dèbil de sistema està en el driver I2C dels sensors d'ultrasons. Els problemes d'aquest driver son:

- El primer consisteix en que la tasca està molt de temps esperant a que s'enviïn les trames bloquejant les tasques de menys prioritat que en aquest cas es el display. Si volguéssim afegir alguna altre tasca de menys prioritat podríem tenir problemes d'execució. Per tant caldria eliminar aquestes esperes.

- El segon problema que te aquest driver es que s'espera infinitament a la recepció de les confirmacions dels esclaus, per tant, en cas que no contestin el driver deixa de funcionar i ja no torna a comunicar-se amb els sensors. En aquest cas, el display deixa

d'actualitzar-se. De fet s'ha detectat que el motor de la mà introdueix soroll en el sistema fent que es perdin confirmacions i quan això passa ja no torna a comunicar-se amb els sensors. Per reduir el soroll introduït per aquest motor, s'ha afegit un varistor en paral·lel al motor que minimitza el problema. Per solucionar el problema definitivament s'ha d'introduir temps màxims d'esperes en la lectures de les confirmacions.

- Un altre possibles problema del sensor d'ultrasons es que triga 300ms, que es un temps massa llarg, en llegir els tres sensors. Es pot optimitzar la lectura, enviant la ordre de començar a fer la mesura, espera 80ms i llegir les tres respostes.

Tenim altre possibles millores en el disseny del programari que tenen a veure en la gestió dels recursos.

- El programari bàsic que s'ha fet no està pensat per tenir més d'una aplicació per sobre. Per millorar el accés a aquests recursos es podríem afegir semàfors i així evitaríem que dos aplicacions volguessin fer la mateix acció.
- Un altre possible millora es agrupar totes les mesures que provenen de diferents perifèrics en un component únic. Així l'aplicació només hauria d'incloure una API única que accedís als convertidors ADC, als sensors I2C i als polsadors i d'aquesta manera el desenvolupador de l'aplicació no necessitaria saber d'on ve el recurs.

6 Conclusions

6.1 Conclusions del treball

Gracies al desenvolupament d'aquest treball he ampliat coneixent sobre el sistema operatiu en temps real FreeRTOS i he aprés a fer servir els recursos línia de Texas Instruments[4] de la placa MSP-EXP432P401R[2]. També he adquirit coneixement en la gestió de projectes com per exemple com planificar-lo o decidir quins processos es faran servir en el desenvolupament.

6.2 Autoavaluació

Finalment s'ha pogut assolir els objectius bàsics i només han quedat pendent algun objectiu secundari. Aquests objectius han estat:

- El desenvolupament dels modes Low Power: Aquest objectiu només aportava aprendre a fer configurar i fer servir aquests modes però no afegia cap valor al producte final.

- Implementació de gràfics en el Display: Com en el cas anterior aquest objectiu estava destinat a adquirir més coneixement de les llibreries de Texas Instruments.

- Implementació més extensa dels modes automàtics: Només s'ha desenvolupat un mode automàticament però tenim totes les eines per afegir més funcionalitats.

Els motius per no haver assolit els objectius anteriors ha estat falta de temps i els problemes apareguts en el desenvolupament del I2C. Inicialment tenia problemes de configuració i d'adaptació de nivells de senyals que vaig trigar massa en resoldre.

Pel que fa a la planificació del projecte, aquesta s'ha anat seguint amb matisos. En primer lloc al principi del projecte disposava de poc temps, així que les fites s'anaven retardant por a poc. I va ser gracies a les vacances de nadal que vaig poder concloure el projecte. Per arribar al objectiu s'ha hagut de fer canvis en la planificació per tal de tenir un producte a més funcionalitat en les fites intermitges. Un exemple d'això ha estat el fet de fer servir el joystick abans del previst per tal de poder començar a fer proves. També vaig deixar de costat el desenvolupament dels sensors I2C per poder tenir abans possible un producte que es mogués.

En quant a la metodologia, per una banda vaig prendre bones decisions com fer servir Subversion i tenir un històric del projecte o comentara al estil Doxygen per tal de tindre un documentació del codi. Per un altre banda no he mantingut adequadament una documentació sobre els testos fets del codi i sobre els informes a entregar en las PAC on faltava molta feina. També hagués hagut de treballar des de molt abans en la elaboració de la memòria final

6.3 Línies de treball de futur

Queda pendent d'implementar els modes Low Power, optimitzar el driver I2C i implementar més modes automàtics.

En quant a la evolució del robot s'ha d'eliminar el cablejat que hi ha entre la placa de control i el robot afegint una segona placa. Una placa contendria tot el programari bàsic de control i es comunicaria amb una segona que enviaria les ordres. Un altre evolució seria dotar al robot amb una webcam i amb una tercera placa afegir funcionalitat de visió per computador. Actualment ja estic treballant en la adquisició de coneixement en aquesta camp.

7 Glossari

CPU: Unitat central de processat

CPU load: Proporció del temps que la CPU està ocupada

ADC: Convertidor Analògic Digital

PWM: Modulat per amplada de polse. Es un mètode per controlar la intensitat en una sortida digital encenent i apagant la sortida a una certa velocitat.

Duty cycle: Proporció de temps que una sortida està activada en una modulació PWM.

PAC: Prova d'avaluació continuada

RTOS: Sistema operatiu en temps real. Aquests sistemes operatius asseguren que una tasca s'executa dins d'un temps determinat.

8 Bibliografia

[1] FreeRTOS. FreeRTOS Documentation [en línia].
https://www.freertos.org/Documentation/RTOS_book.html [data de consulta 23/10/2017]

[2] Texas Instruments. SimpleLink™ MSP432P401R LaunchPad™ Development Kit [en línia].
http://www.ti.com/tool/MSP-EXP432P401R [data de consulta 18/10/2017]

[3] Energia. Prototyping Software to Make Things Easy [en línia].
http://energia.nu/ [data de consulta 21/11/2018]

[4] Texas Instruments. Educational BoosterPack MKII [en línia].
http://www.ti.com/tool/BOOSTXL-EDUMKII [data de consulta 21/11/2017]

[5] Tenda Lego. Lego Mindstorms EV3 [en línia].
https://shop.lego.com/es-ES/LEGO-MINDSTORMS-EV3-31313 [data de consulta 07/01/2018]

[6] Centellas Vela, E. (2013). Blog de projectes personals [en línia].
http://svbot.blogspot.com.es/ [data de consulta 13/01/2018]

[7] Doxygen. Generate documentation from source code [en línia].
http://www.stack.nl/~dimitri/doxygen/ [data de consulta 09/10/2017]

[8] FSM. Finite State Machine Editor and C-code generator [en línia].
http://robertolanuza.tripod.com/fsm/fsm.htm [data de consulta 25/11/2017]

[9] Subversion. Apache™ Subversion® [en línia].
https://subversion.apache.org/ [data de consulta 9/10/2017]

[10] Robot electronics. SRF02 Ultrasonic range finder [en línia].
https://www.robot-electronics.co.uk/htm/srf02techSer.htm [data de consulta 20/12/2017]

[11] CSD. The I2C bus specification [en línia].
http://www.csd.uoc.gr/~hy428/reading/i2c_spec.pdf [data de consulta 23/12/2017]

[12] Arduino M0 Pro. Arduino [en línia].
https://store.arduino.cc/arduino-m0-pro [data de consulta 13/01/2018]

[13] LPCXpresso54628. NXP [en línia]. https://www.nxp.com/support/developer-resources/hardware-development-tools/lpcxpresso-boards/lpcxpresso54628-development-board:OM13098 [data de consulta 13/01/2018]

[14] Uso de motores con Arduino. Aprendiendo Arduino [en línia]. https://aprendiendoarduino.wordpress.com/tag/motor-paso-a-paso/ [data de consulta 14/01/2018]