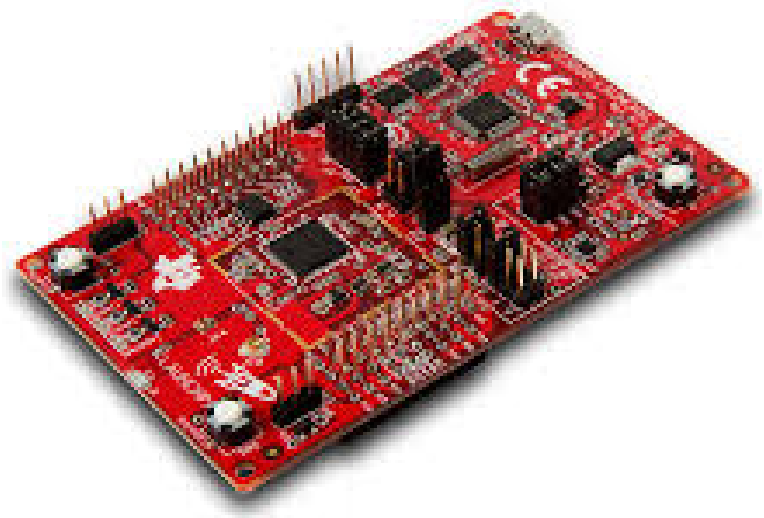
Programació d'Arquitectures Encastades

Projecte final

Semestre de Primavera 2023



David Fernàndez
Víctor Sort

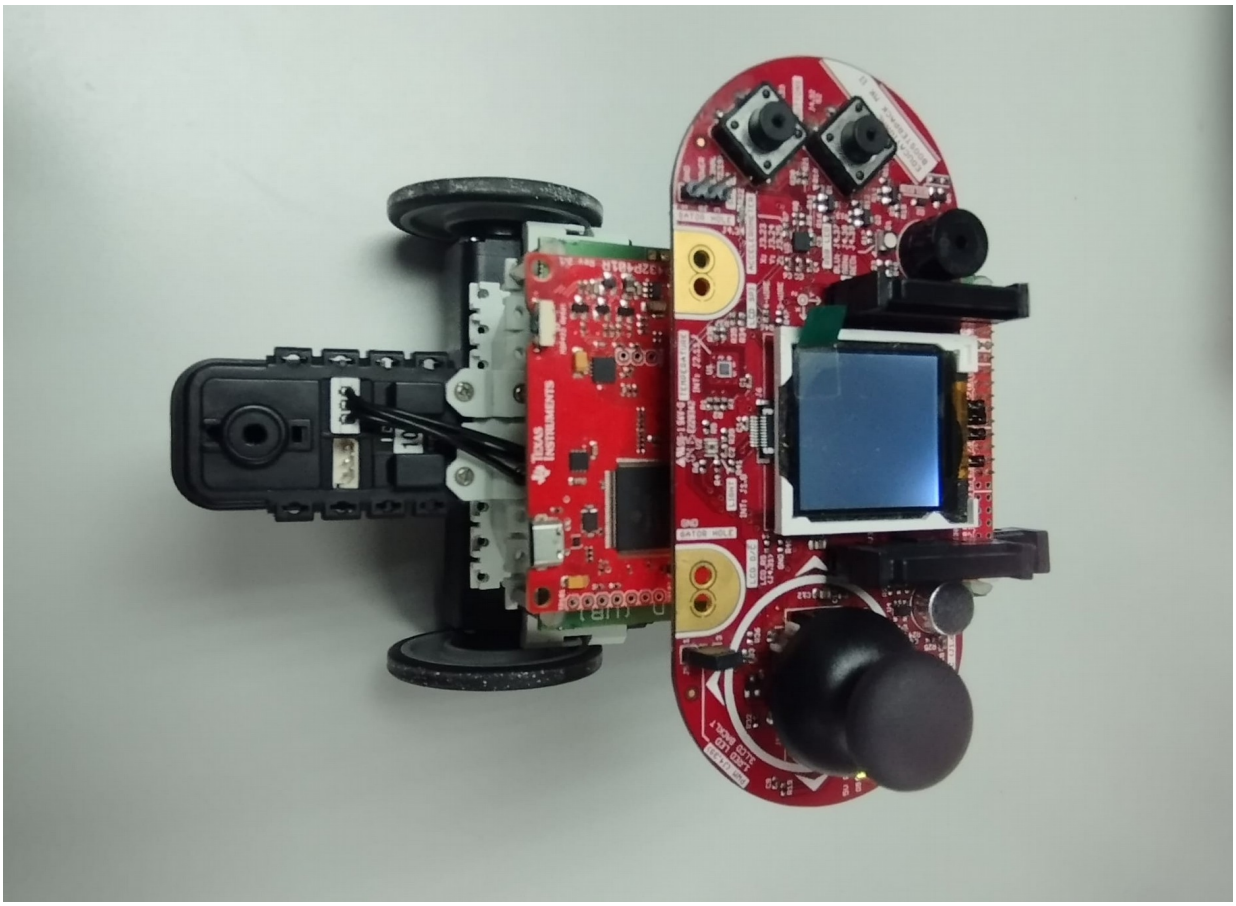
David Fernàndez
Víctor Sort

ÍNDEX

David Fernàndez
Víctor Sort

1. OBJECTIUS

En aquesta pràctica hem fet servir un robot amb les següents especificacions: placa Launchpad (amb el microcontrolador MSP432P401R), placa Boosterpack (que conté polsadors i un joystick), dos mòduls Dynamixel AX-12 i un mòdul Dynamixel AX-S1.



Robot usat a la pràctica

L'objectiu d'aquest projecte final és aconseguir un moviment autònom per part d'un robot que eviti la col·lisió amb els objectes en un espai, seguint les següents pautes:

- Inicialment, ha de buscar una paret.
- Ha de recórrer la paret en un sentit determinat.
- En cas de trobar un obstacle o una cantonada de la paret, l'ha de rodejar i continuar movent-se en el mateix sentit si és possible.
- En cas de trobar un camí sense sortida ha de fer marxa enrere i sortir.

La idea del codi, a nivell conceptual, consisteix en anar llegint periòdicament la informació dels sensors del robot i actualitzar el seu moviment en funció d'aquesta informació.

David Fernàndez
Víctor Sort

Per fer el codi s'utilitza la llibreria creada a la pràctica 4, anomenada llibreria_robot.h, ja que conté els recursos bàsics per comunicar la placa del microcontrolador MSP432P401R amb el robot mitjançant el seu mòdul eUSCI amb el protocol asíncron UART (els mètodes init_UART(), TxPacket() i RxPacket()), així com un gran conjunt de funcions per controlar el moviment del robot, que es mouen gràcies als motors Dynamixel, i la lectura dels seus sensors:

- Encendre i apagar els LEDs dels motors.
- Configurar el mode continu dels motors (endless turn).
- Aturar-se.
- Avançar recte.
- Retrocedir recte.
- Avançar amb gir cap a la dreta i cap a l'esquerra.
- Retrocedir amb gir cap a la dreta i cap a l'esquerra.
- Girar en sentit horari i antihorari.
- Pivotar sobre ell mateix en sentit horari i antihorari.
- Llegir els tres sensors de distància.

A més a més, com a objectiu extra, s'han utilitzat els components de la placa Boosterpack per millorar el codi, afegint funcionalitats que expliquem més endavant.

2. RECURSOS UTILITZATS

Com ha estat explicat a la introducció, en aquesta pràctica hem treballat amb el robot ja presentat, que conté una sèrie de recursos que hem emprat per implementar la solució a la problemàtica.

Inicialment, de la placa Launchpad del robot s'ha usat l'eUSCI en mode UART per enviar comunicacions, tant als seus 2 mòduls Dynamixel AX-12 com al mòdul Dynamixel AX-S1. Dels mòduls Dynamixel AX-12, s'utilitzen els seus LEDs, per comprovar la correcta configuració, i els seus motors, per moure el robot.

Mòduls Dynamixel AX-12

Del mòdul Dynamixel AX-S1, s'usen els sensors infrarojos esquerre, dret i frontal, per mesurar la ubicació del robot respecte a les parets o obstacles, el buzzer, per reproduir melodies, i el micròfon intern, per detectar el nivell sonor ambiental quantitativament.

S'han afegit dos mètodes a la llibreria creada a la pràctica 4 per poder llegir si el sensor detecta que s'ha picat de mans i per dir al buzzer que emeti un so. Aquests dos mètodes es poden trobar a l'annex 6.1.1. Cal esmentar que l'ID del sensor és 100.

Mòdul Dynamixel AX-S1

A continuació es dona una informació bàsica sobre aquests mòduls. Més informació detallada sobre ells, de com ha estat configurada la UART i la seva connexió amb els mòduls es pot trobar a l'entrega de l'informe de la pràctica 4.

La connexió entre la placa i els mòduls, que es connecten en daisy chain, es fa mitjançant una UART A2 del port 3, configurant els pins 3.2 i 3.3 com a GPIO (P3.2 = UART2RX i P3.3 = UART2TX). El baud rate programat dels mòduls Dynamixel és de 500 kbps, i per tant la UART s'ha de configurar per treballar a aquesta velocitat.

Els mòduls Dynamixel fan servir una comunicació asíncrona però Half-duplex (només té una línia "Data" per transmetre i rebre) mentre que la UART en principi és Full-duplex (té una línia per transmetre UCAxTXD i una per rebre UCAxRXD). Per solucionar-ho s'ha de fer un circuit que passi de dues línies a una i viceversa.

S'ha hagut de tenir en compte que des del microcontrolador, per programa, s'ha de controlar el senyal "DIRECTION_PORT". A la nostra placa l'hem connectat al port 3, al pin 3.0. S'ha d'inicialitzar com GPIO de sortida, i gestionar el senyal en funció de si volem enviar o rebre missatges.

Un altre recurs que s'ha usat és el timer A0, per controlar el temps del codi del main i el timer A1 per controlar els possibles timeouts de la comunicació amb el robot. A continuació es troba el codi comentat de configuració del timer A1, fet perquè generi interrupcions cada desena d'un microsegon. El codi de configuració del timer A0 és completament idèntic.

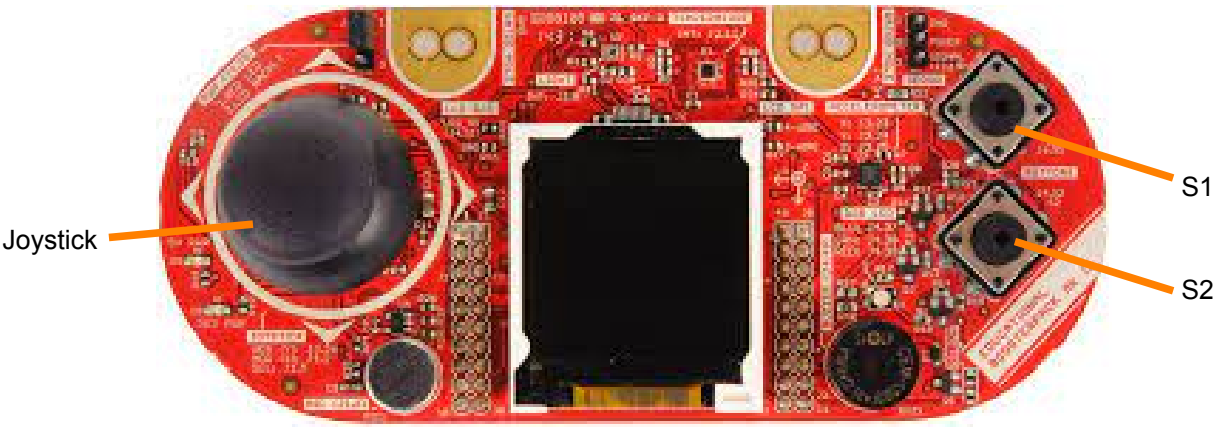
```
void init_timers(void){
    //Timer A1, used for calculating possible timeout
    //Divider = 1; CLK source is SMCLK; clear the counter; MODE is stop
    TIMER_A1->CTL = TIMER_A_CTL_ID_1 | TIMER_A_CTL_SSEL_SMCLK | TIMER_A_CTL_CLR
| TIMER_A_CTL_MC_STOP;
    TIMER_A1->CCR[0] = 240 - 1;     //100kHz 10^-5 s (decenas de microsegundos)
   TIMER_A1->CCTL[0] |= TIMER_A_CCTLN_CCIE; //Activem les interrupcions del
timer A1
}
```

Per implementar funcionalitats extres que detallarem més endavant s'han configurat els polsadors S1 i S2 de la placa Boosterpack com a GPIOs d'entrada, així com el joystick. A continuació es mostra una taula amb el pin de cada recurs i el codi de configuració dels recursos en el port 5:

Recurs	Port.pin	Recurs	Port.pin
S1	P5.1	JOY_AVALL	P5.5
S2	P3.5	JOY_DRETA	P4.5
JOY_AMUNT	P5.4	JOY_ESQUERRA	P4.7

```
P5SEL0 &= ~(BIT1 + BIT4 + BIT5); //Els recursos es configuren com a GPIO
P5SEL1 &= ~(BIT1 + BIT4 + BIT5); //Els recursos es configuren com a GPIO
P5DIR &= ~(BIT1 + BIT4 + BIT5); //Els recursos son una entrada
P5REN |= (BIT1 + BIT4 + BIT5); //Pull-up/pull-down pels recursos
P5OUT |= (BIT1 + BIT4 + BIT5); //Donat que l'altre costat es GND, volem una
pull-up
P5IE |= (BIT1 + BIT4 + BIT5); //Interrupcions activades
P5IES &= ~(BIT1 + BIT4 + BIT5); //Amb transicio L->H
P5IFG = 0; //Netegem les interrupcions anteriors
```

La configuració dels recursos en el port 3 o en el port 4 és completament anàloga a la anterior en el port 5.



Placa Boosterpack

David Fernàndez
Víctor Sort

3. PROBLEMES

Realitzant aquesta pràctica, a part de petits problemes de compilació, hem tingut com a major dificultat la realització del cas que el robot es vegi ficat en un camí sense sortida.

El primer problema important va ser que, com es pot apreciar a les imatges del robot, si posem el robot enganxat lateralment a una paret, el sensor del costat corresponent està a molta més distància que si fem el mateix frontalment. És per això que vam haver de pensar un sistema de valors per controlar aquesta diferència frontal i lateral en la utilització dels sensors. Per explicar-ho millor, si fèiem corregir el moviment del robot quan la lectura del sensor arribés a una certa xifra, frontalment reaccionava molt abans que lateralment i, per exemple, en el cas de trobar una paret davant i haver de girar, no detectava la paret lateralment en pivotar. Això provocava que pivotés sense parar i vam haver de prendre mesures per solucionar-ho.

Amb relació als camins sense sortida, hem tingut una gran dificultat per determinar una solució, per la quantitat de casos diferents existents. Es volia fer servir una solució depenent només de la lectura dels sensors, però a causa dels casos possibles, no es podia implementar una solució que funcionés. Per això, es va fer servir una solució mitjançant un timer, però no ha estat completament satisfactòria per la variació de les velocitats possibles dels motors.

Vam trobar, per últim, una gran dificultat un cop ja estaven implementades totes les funcionalitats del robot per poder calibrar els valors de referència pel moviment del robot, emprats per la correcció dels moviments respecte la proximitat o allunyament del robot respecte les parets o obstacles. Això es va demanar per part del professorat, ja que s'havia de poder emprar qualsevol robot, amb sensors més o menys sensibles. Primer de tot, es va intentar fer servir els joysticks per calibrar cadascuna de les distàncies, però el resultat en cada intent era molt diferent. Vam fer servir dues sessions senceres per intentar arreglar-ho, fins que vam implementar la calibració dels tres sensors simultàniament.

4. EXPLICACIÓ DE MÈTODES

4.1 Main

El mètode main té la següent estructura:

```
void main(void){
    //Aquí estarien totes les inicialitzacions / configuracions
    Activa_TimerA0_TimeOut(); //Activem el timer A0
    while(true){ //Bucle infinit
        while(!palmada()); //Esperem a rebre una palmada per començar
        melodia(); //Fem sonar una melodia abans de començar
        trobar_paret(); //Trobem una paret
        parar_robot = false; //Posem aquesta variable global a false
        while(!parar_robot){ //Mentres segueixi a false, el robot es mou
            moviment_robot_infinit();
        }
        parar(); //Si sortim del bucle, parem el robot
        melodia2(); //Fem sonar una melodia de final d'execució
    }
}
```

Així doncs, primer es fan totes les inicialitzacions i configuracions, les quals no estan mostrades per un tema d'espai però si es poden veure als annexos 6.1.2, on es troba tot el codi del main.c o 6.2.1, on es troba el seu diagrama de flux.

A continuació s'activa el timer A0 i s'entra dintre un bucle infinit. Aquest bucle infinit permet que es pugui parar i reiniciar el robot sense necessitat de tornar a carregar el codi. El robot començarà el seu moviment quan el sensor de soroll del Dynamixel AX-S1 detecti un picar de mans, fent sonar primer una melodia inicial amb el buzzer i cridant al mètode perquè trobi la paret.

Després es posa una variable global, parar_robot, a false, i fins que no estigui a true es crida al moviment_robot_infinit. Quan parar_robot es posi a true, ja explicarem posteriorment com, el robot pararà i sonarà una melodia final. Tornarem a repetir doncs el procés, esperant un altre cop l'activació del robot amb un picar de mans.

Abans d'arribar al bucle inifinit, es troba el següent codi:

```
    while(!calibracio_feta){
```

```
        let = leer_sensores();
        Reset_TimeoutA0();
        while(!TimeOutA0(TEMPS_ESPERA_LECTURA));
    }
    buzzer(6, 255); //calibrat
```

La variable global calibracio_feta és un booleà que es defineix inicialment a false. Mentre no es posi a true, s'aniran llegint de manera periòdica els sensors infrarojos, i un cop s'hi posi, es farà sonar una curta melodia per indicar que s'ha fet la caliibració. Posteriorment explicarem com es fa aquesta calibració i com es posa la variable a true.

4.2 Trobar_paret

No incloem l'estructura del mètode per la seva llargària. Es pot veure però als annexos 6.1.2, on es troba tot el codi del main.c o 6.2.2, on es troba el seu diagrama de flux.

El que fa el mètode és dir-li al robot que avanci fins que algun dels seus sensor de distància detectin que tenen la paret a prop.

En el cas de que la paret la tingui a un dels dos costats (que no és el cas habitual) es marcarà que la paret que seguirà el robot és aquella.

En el cas de que detecti paret als dos costats a la vegada (que és encara un cas molt més estrany), es marcarà que seguirà la que tingui més a prop.

Finalment, en el cas de que no detecti paret als costats, sinó que la detecta davant, que és el que succeeix la gran majoria de vegades, es farà pivotar el robot en sentit horari o antihorari fins que tingui la paret a la dreta o esquerra, depenent del que s'ha predefinit prèviament amb un define a l'inici del codi, i es marcarà que la paret a seguir és aquella.

4.3 Moviment_robot_infinit

L'estructura d'aquest mètode és el següent:

```
void moviment_robot_infinit(void){
    Reset_TimeoutA0();
    while(!TimeOutA0(TEMPS_ESPERA_LECTURA)); //Esperem TEMPS_ESPERA_LECTURA
segons.
    let = leer_sensores(); //Llegim els sensors.
    if(costat_paret){    ...    }
    else{    ...    }
}
```

El que fa aquest mètode és esperar-se un temps (de l'ordre de 10^-3 segons), llegir els sensors de distància i depenent de la paret que estiguem seguint i depenent de la lectura dels sensors el robot farà un acció o una altre. Aquest mètode serveix per veure les diferents possibilitats i fer la classificació. Es pot veure en les descripcions dels mètodes posteriors en quin cas exacte ens trobem, i com es resolen.

El codi complet es pot veure en l'annex 6.1.2, on es troba tot el codi del main.c o 6.2.3, on es troba el seu diagrama de flux. A més a més, en la presentació entregada es pot veure de manera visual quins són els 6 casos possibles en els que es pot trobar el robot quan recorre la paret esquerra i com actua depenent d'ells.

4.4 Interrupcions joystick i polsadors

S'han fet 3 mètodes per gestiones les interrupcions (PORTX_IRQHandler), ja que els polsadors S1 i S2 i el joysticks es troben en els ports 3, 4 o 5. S'explica en aquest apartat que succeeix quan es produeixen les interrupcions, sense fer cap distinció pels ports.

Al prémer el polsador S1 simplement es posa la variable global para_robot a true, que és la que ens serveix per sortir del bucle infinit en el main que crida al mètode moviment_robot_infinit().

Al prémer el polsador S2 es fa la calibració del robot. La idea és posar al robot a una posició on tingui paret a l'esquerra, dreta i al front i un cop es prem el polsador les distàncies que llegeix amb els sensors d'infrarojos es converteixen en els valors de les variables globals usades per el moviment del robot.

```
if(!calibracio_feta){
      {if(let.Left +5 < 255 && let.Right +5 < 255 && let.Left-5 > 0 && let.Right-5 >0 && let.Center > 0 && let.Center < 255){
            proximitat_maxima_frontal = let.Center;
            proximitat_minima_esquerra = let.Left - 5;
            proximitat_maxima_esquerra = let.Left + 5;
            proximitat_minima_dreta = let.Right - 5;
            proximitat_maxima_dreta = let.Right + 5;
            proximitat_no_paret = proximitat_minima_dreta/2;
            if(proximitat_no_paret > proximitat_minima_esquerra/2){
                proximitat_no_paret > proximitat_minima_esquerra/2;
            }
            calibracio_feta = true;
            }
}
```

Com es veu, primer es comprova que no s'hagi fet ja prèviament (aquesta condició és necessària doncs no es vol calibrar el robot en moviment) i que els valors que llegeix, en cas de suma o resta de 5, no passin els límits dels valors permesos. Finalment, tal i com es veu, s'assignen els valors a les variables. En els laterals, hi ha una distància mínima i una de màxima, doncs així el robot no està constantment rectificant el seu recorregut.

Finalment es posa la variable global calibracio_feta a true, que és la que ens serveix per sortir del bucle infinit que espera a que es faci la calibració.

Al moure el joystick cap amunt, la variable global proximitat_maxima_frontal s'incrementa en 5 (si no supera el valor màxim permès).

Al moure el joystick cap avall, la variable global proximitat_maxima_frontal es decrementa en 5 (si no supera el valor mínim permès).

Al moure el joystick cap a la dreta, les variables globals proximitat_maxima_dreta, proximitat_maxima_esquerra, proximitat_minima_dreta i proximitat_minima_esquerra s'incrementen en 5 (si no superen el valor màxim permès).

Al moure el joystick cap a l'esquerra, les variables globals proximitat_maxima_dreta, proximitat_maxima_esquerra, proximitat_minima_dreta i proximitat_minima_esquerra es decrementen en 5 (si no superen el valor mínim permès).

Així doncs, el joystick serveix també per fer calibració però durant l'execució del robot. Com és molt difícil encertar a la primera la calibració correcta del robot perquè funcioni adequadament i es perdria bastant de temps per la demo del robot fer la calibració exacta, les variables globals de control s'han definit ja inicialment perquè funcionin amb el robot número 10 i s'ha inicialitzat calibracio_feta com a true, així que no cal refer la calibració.

David Fernàndez
Víctor Sort

5. CONCLUSIONS

Per començar, volem dir que estem molt satisfets amb els nostres codis finals ja que creiem que satisfan l'objectiu del projecte en tots els casos possibles. Hem aconseguit configurar bé el robot i programar-lo perquè trobi una paret i la ressegueixi en un sentit, evitant tots els obstacles i complicacions que pugui tenir.

Creiem que hem intentat també anar bastant més enllà del demanat, implementant totes les altres funcionalitats que ja hem anat explicant en aquest informe. D'aquesta manera, considerem que aquest projecte final ha sigut una bona manera d'acabar l'assignatura, ja que és una manera de veure com tot el que hem anat treballant durant el curs, tant a les classes de teoria com a les de laboratori, es pot combinar en un mateix projecte.

Respecte el temps invertit en la realització del projecte, creiem que ha sigut clarment superior a les anteriors pràctiques però hem notat satisfets que el treball que ja hem fet anteriorment, com la configuració de ports com a GPIOs, timers o interrupcions l'hem realitzat de manera molt més ràpida en aquest projecte final.

Finalment, creiem que hem treballat molt bé en equip, que ens hem repartit les tasques equitativament, en hem ajudat mútuament i esforçat per aprendre i optimitzar al màxim, dins les nostres capacitats, el codi d'aquest projecte. En resum, i en la nostra opinió, hem fet un molt bon treball.

6. ANNEXOS

6.1 Programa Comentat

6.1.1 Llibreria_robot.c nous mètodes

Els dos mètodes afegits a la llibreria creada a la practica 4 són els següents:

```
bool palmada(void){
    RxReturn ret;
    byte bID = ID_SENSOR;
    byte bParameterLength = 2;
    byte bInstruction = READ_INSTRUCTION; //READ-DATA (0x02)
    byte Parametros[16];
    Parametros[0] = 0x23; //Llegim la posicio 0x23
    Parametros[1] = 0x01;
    ret = enviarInstruccion(bID, bParameterLength, bInstruction, Parametros);
    //Enviem la instruccio i recuperem la resposta
    if(ret.StatusPacket[5]>200)
        return true; //retornem true si el sensor ha detectat un soroll de valor
        més gran que 200
    return false; //retornem fals en cas contrari
}

void buzzer(uint16_t index, uint16_t time){
    byte bID = ID_SENSOR;
    byte bParameterLength = 3;
    byte bInstruction = WRITE_INSTRUCTION; //WRITE-DATA (0x03)
    byte Parametros[16];
    Parametros[0] = 0x28; //Escrivim a les posicions 0x28 i 0x29
    Parametros[1] = index;
    Parametros[2] = time;
    enviarInstruccion(bID, bParameterLength, bInstruction, Parametros); //enviem
la instruccio i recuperem la resposta
}
```

6.1.2 Main.c

```
/*****
* Projecte final PAE
* David Fernàndez i Víctor Sort
* 06/2023
*****/
#include "llibreria_robot.h"
#define VELOCITAT 800
#define TEMPS_ESPERA_LECTURA 1000 //0.01s
#define SENTIT_GIR 1 //Pren els valors: 0  per esquerra, 1 per dreta
volatile int proximitat_maxima_frontal = 120;
volatile int proximitat_minima_esquerra = 60;
volatile int proximitat_maxima_esquerra = 70;
volatile int proximitat_minima_dreta = 60;
volatile int proximitat_maxima_dreta = 70;
volatile int proximitat_no_paret = 30;
volatile bool calibracio_feta = false;
volatile bool parar_robot;
volatile int cont_time_out;
volatile LecturaSensores let; //3 bytes, per els sensors esquerra, central i
dret
volatile int costat_paret; //Pren els valors: 0  per esquerra, 1 per dreta
/*****
* INIT_TIMER_MAIN
* Métode usat per inicializar el timer A0 que s'utilitzarà per calcular els
timeouts
*****/
void init_timer_main(void){
    //Divider = 1; CLK source is SMCLK; clear the counter; MODE is stop
    TIMER_A0->CTL = TIMER_A_CTL_ID_1 | TIMER_A_CTL_SSEL_SMCLK | TIMER_A_CTL_CLR
| TIMER_A_CTL_MC_STOP;
    TIMER_A0->CCR[0] = 240 - 1; //100kHz 10^-5 s (decenas de microsegundos)
    TIMER_A0->CCTL[0] |= TIMER_A_CCTLN_CCIE; //Activem les interrupcions del
timer A0
}
/*****
* INIT_INTERRUPCION_MAIN
* Métode usat per inicializar les interrupcions del timer A0
*****/
void init_interrupcion_main(){
    //Int. TA0 de CCTL0, corresponde al bit 10 del primer registro ISER0
    NVIC->ICPR[0] |= 1 << TA0_0_IRQn; //Primero, me aseguro de que no quede
ninguna interrupcion residual pendiente para este puerto,
    NVIC->ISER[0] |= 1 << TA0_0_IRQn; //y habilito las interrupciones del puerto
    //Int. port 3 = 37 corresponde al bit 6 del segundo registro ISER1:
    NVIC->ICPR[1] |= 1 << (PORT3_IRQn & 31); //Primero, me aseguro de que no
quede ninguna interrupcion residual pendiente para este puerto,
    NVIC->ISER[1] |= 1 << (PORT3_IRQn & 31); //y habilito las interrupciones del
puerto
    //Int. port 4 = 38 corresponde al bit 6 del segundo registro ISER1:
```

```
    NVIC->ICPR[1] |= 1 << (PORT4_IRQn & 31); //Primero, me aseguro de que no
quede ninguna interrupcion residual pendiente para este puerto,
    NVIC->ISER[1] |= 1 << (PORT4_IRQn & 31); //y habilito las interrupciones del
puerto
    //Int. port 5 = 39 corresponde al bit 7 del segundo registro ISER1:
    NVIC->ICPR[1] |= 1 << (PORT5_IRQn & 31); //Primero, me aseguro de que no
quede ninguna interrupcion residual pendiente para este puerto,
    NVIC->ISER[1] |= 1 << (PORT5_IRQn & 31); //y habilito las interrupciones del
puerto
}
/*****
* ACTIVA_TIMERA0_TIMEOUT
* Posa el timer A0 en mode UP, activant-lo
*****/
void Activa_TimerA0_TimeOut(void){
    Reset_TimeoutA0();
    TIMER_A0->CTL |= TIMER_A_CTL_MC__UP;
}
/*****
* DESACTIVA_TIMERA0_TIMEOUT
* Posa el timer A0 en mode STOP, desactivant-lo
*****/
void Desactiva_TimerA0_TimeOut(void){
    cont_time_out = 0;
    TIMER_A0->CTL |= TIMER_A_CTL_MC__STOP;
}
/*****
* RESET_TIMEOUTA0
* Fa reset al comptador que controla el timeout
*****/
void Reset_TimeoutA0(void){
    cont_time_out = 0;
}
/*****
* TIMEOUTA0
* Retorna 1 si el comptador de timeout supera el valor passat per parametre, 0
altrament
*****/
byte TimeOutA0(int TimeOut){
    if (cont_time_out > TimeOut){ return 1; }
    else{ return 0; }
}
/*****
* INIT_BOTONS
* Inicialitza els polsadors S1 I S2 i el joystick
* Polsadors = P3.5, P5.1
* Joystick = P4.5, P4.7, P5.4, P5.5
*****/
void init_botons(){
```

```
    P3SEL0 &= ~(BIT5); //El polsador es configura com GPIO
    P3SEL1 &= ~(BIT5); //El polsador es configura com GPIO
    P3DIR &= ~(BIT5); //Un polsador es una entrada
    P3REN |= (BIT5); //Pull-up/pull-down pel polsador
    P3OUT |= (BIT5); //Donat que l'altre costat es GND, volem una pull-up
    P3IE |= (BIT5); //Interrupcions activades
    P3IES &= ~(BIT5); // amb transicio L->H
    P3IFG = 0; //Netegem les interrupcions anteriors
    P4SEL0 &= ~(BIT5 + BIT7); //El joystick es configura com GPIO
    P4SEL1 &= ~(BIT5 + BIT7); //El joystick es configura com GPIO
    P4DIR &= ~(BIT5 + BIT7); //Un joystick es una entrada
    P4REN |= (BIT5 + BIT7); //Pull-up/pull-down pel joystick
    P4OUT |= (BIT5 + BIT7); //Donat que l'altre costat es GND, volem una pull-up
    P4IE |= (BIT5 + BIT7); //Interrupcions activades
    P4IES &= ~(BIT5 + BIT7); // amb transicio L->H
    P4IFG = 0; //Netegem les interrupcions anteriors
    P5SEL0 &= ~(BIT1 + BIT4 + BIT5); //El polsador i joystick es configuren com
GPIO
    P5SEL1 &= ~(BIT1 + BIT4 + BIT5); //El polsador i joystick es configuren com
GPIO
    P5DIR &= ~(BIT1 + BIT4 + BIT5); //Un polsador i un joystick son una entrada
    P5REN |= (BIT1 + BIT4 + BIT5); //Pull-up/pull-down pel polsador
    P5OUT |= (BIT1 + BIT4 + BIT5); //Donat que l'altre costat es GND, volem una
pull-up
    P5IE |= (BIT1 + BIT4 + BIT5); //Interrupcions activades
    P5IES &= ~(BIT1 + BIT4 + BIT5); // amb transicio L->H
    P5IFG = 0; //Netegem les interrupcions anteriors
}
/*****
* MELODIA
* Fa sonar el buzzer per fer una melodia
*****/
void melodia(){
    uint16_t tiempo = 0;    //0.3s
    uint16_t timepo_espera = 30000; //0.3s
    buzzer(5, tiempo);
    Reset_TimeoutA0();
    while(!TimeOutA0(timepo_espera));
    buzzer(7, tiempo);
    Reset_TimeoutA0();
    while(!TimeOutA0(timepo_espera));
    buzzer(8, tiempo*2);
    Reset_TimeoutA0();
    while(!TimeOutA0(timepo_espera*2));
    buzzer(5, tiempo);
    Reset_TimeoutA0();
    while(!TimeOutA0(timepo_espera));
    buzzer(8, tiempo);
    Reset_TimeoutA0();
```

```
    while(!TimeOutA0(timepo_espera));
    buzzer(7, tiempo);
    Reset_TimeoutA0();
    while(!TimeOutA0(timepo_espera));
    buzzer(5, tiempo);
    Reset_TimeoutA0();
    while(!TimeOutA0(timepo_espera));
    buzzer(3, tiempo);
    Reset_TimeoutA0();
    while(!TimeOutA0(timepo_espera));
    buzzer(3, tiempo*2);
    Reset_TimeoutA0();
    while(!TimeOutA0(timepo_espera*2));
    buzzer(5, tiempo);
    Reset_TimeoutA0();
    while(!TimeOutA0(timepo_espera));
    buzzer(5, tiempo*2);
    buzzer(0, 255); //melodia predeterminada
    Reset_TimeoutA0();
    while(!TimeOutA0(timepo_espera));
}
/*****
* MELODIA2
* Fa sonar el buzzer per fer una melodia
*****/
void melodia2(){
    buzzer(3, 255); //melodia predeterminada
    Reset_TimeoutA0();
    while(!TimeOutA0(500000));
}
/*****
* TROBAR_PARET
* Métode que condueix al robot a trobar una paret
*****/
void trobar_paret(void){
    Reset_TimeoutA0();
    let = leer_sensores();
    avanzar(VELOCITAT); //Avancem continuament fins que es trobi una paret aprop
    while(let.Left < proximitat_minima_esquerra && let.Center <
proximitat_maxima_frontal && let.Right < proximitat_minima_dreta){
        Reset_TimeoutA0();
        while(!TimeOutA0(TEMPS_ESPERA_LECTURA));
        let = leer_sensores();
    }
    if(let.Right >= proximitat_minima_dreta || let.Left >=
proximitat_minima_esquerra){ //Si s'ha detectat paret en algun costat.
        if(let.Right >= let.Left){ costat_paret = 1; } //Indiquem que seguirem
aquella paret, depenent del costat.
        else{ costat_paret = 0; }
```

```
    }
    else{ //En el cas de que la paret no s'ha detecat a cap costat i es troba a
davant,
        if(SENTIT_GIR){ //Si s'ha predefinit que volem tenir la paret a la
dreta,
            pivotar_sobre_si_mismo_antihorario(VELOCITAT); //Pivotem de manera
antihoraria fins tenir la paret a la dreta.
            while(let.Right < proximitat_minima_dreta){
                Reset_TimeoutA0();
                while(!TimeOutA0(TEMPS_ESPERA_LECTURA));
                let = leer_sensores();
                costat_paret = 1; //Indiquem que seguirem la paret dreta.
            }
        }
        else{ //Si s'ha predefinit que volem tenir la paret a l'esquerra,
            pivotar_sobre_si_mismo_horario(VELOCITAT); //Pivotem de manera
antihoraria fins tenir la paret a la dreta.
            while(let.Left < proximitat_minima_esquerra){
                Reset_TimeoutA0();
                while(!TimeOutA0(TEMPS_ESPERA_LECTURA));
                let = leer_sensores();
                costat_paret = 0; //Indiquem que seguirem la paret esquerra.
            }
        }
    }
}
/*****
* MOVIMENT_ROBOT_INFINIT
* Métode que controla el moviment continu del robot
*****/
void moviment_robot_infinit(void){
    Reset_TimeoutA0();
    while(!TimeOutA0(TEMPS_ESPERA_LECTURA)); //Esperem TEMPS_ESPERA_LECTURA
segons.
    let = leer_sensores(); //Llegim els sensors.
    //Depenent de la paret que estiguem seguint i depenent de la lectura dels
sensors el robot farà una
    //acció o una altre. Aquest métode serveix per veure les diferents
possibilitats i fer la classificació.
    //Es pot veure en les descripcions dels métodes posteriors en quin cas
exacte ens trobem.
    if(costat_paret){
        if(let.Center >= proximitat_maxima_frontal){
            if(let.Left >= proximitat_minima_esquerra){ caso_2_derecha(); }
            else{ caso_3_derecha(); }
        }
        else if(let.Left >= proximitat_maxima_esquerra){ caso_2_derecha(); }
        else{
            if(let.Right >= proximitat_maxima_dreta){ caso_6_derecha(); }
```

```
            else if(let.Right < proximitat_minima_dreta){
                if(let.Right < proximitat_no_paret){ caso_4_derecha(); }
                else{ caso_5_derecha(); }
            }
            else{ caso_1_derecha(); }
        }
    }
    else{
        if(let.Center >= proximitat_maxima_frontal){
            if(let.Right >= proximitat_minima_dreta){ caso_2_izquierda(); }
            else{ caso_3_izquierda(); }
        }
        else if(let.Right >= proximitat_maxima_dreta){ caso_2_izquierda(); }
        else{
            if(let.Left >= proximitat_maxima_esquerra){ caso_6_izquierda(); }
            else if(let.Left < proximitat_minima_esquerra){
                if(let.Left < proximitat_no_paret){ caso_4_izquierda(); }
                else{ caso_5_izquierda(); }
            }
            else{ caso_1_izquierda(); }
        }
    }
}
/*****
* CASO_1_DERECHA
* El robot segueix la paret de la dreta i està dintre els valors de distància
correctes
*****/
void caso_1_derecha(){
    avanzar(VELOCITAT); //Tot correcte, avancem.
}
/*****
* CASO_2_DERECHA
* El robot segueix la paret de la dreta i no pot seguir avançant
****/
void caso_2_derecha(){
    //Retrocedim fins que no tinguem paret a un costat, cuidant no col·lisionar
amb cap paret.
    while(let.Right >= proximitat_no_paret && let.Left >= proximitat_no_paret){
        if(let.Right >= proximitat_maxima_dreta && let.Left <
proximitat_maxima_esquerra){
            retroceder_con_giro(VELOCITAT/1.5, VELOCITAT);
        }
        else if(let.Right < proximitat_maxima_dreta && let.Left >=
proximitat_maxima_esquerra){
            retroceder_con_giro(VELOCITAT, VELOCITAT/1.5);
        }
        else{
            retroceder(VELOCITAT);
```

```
        }
        Reset_TimeoutA0();
        while(!TimeOutA0(TEMPS_ESPERA_LECTURA));
        let = leer_sensores();
    }
    //Si deixem de tenir paret a l'esquerra, girem uns 90 graus cap a l'esquerre
i avancem.
    if(let.Left < proximitat_no_paret){
        pivotar_sobre_si_mismo_antihorario(VELOCITAT);
        while(!TimeOutA0(30000)); //0.5s
        let = leer_sensores();
        avanzar(VELOCITAT);
        if(let.Right < proximitat_minima_dreta){ //Si no detectem paret a la
dreta...
            while(let.Center < proximitat_maxima_frontal && let.Right <
proximitat_minima_dreta){ //Seguim avançant fins topar-nos amb una paret.
                Reset_TimeoutA0();
                while(!TimeOutA0(TEMPS_ESPERA_LECTURA));
                let = leer_sensores();
            }
            caso_3_derecha(); //Ens trobem en el cas 3, així que el cridem.
        }
    }
    //Si deixem de tenir paret a la dreta, retrocedim girant mig segon i
avancem.
    else{
        retroceder_con_giro(VELOCITAT, 0);
        while(!TimeOutA0(30000)); //0.5s
        let = leer_sensores();
        avanzar(VELOCITAT);
        if(let.Right < proximitat_minima_dreta){ //Si no detectem paret a la
dreta...
            while(let.Center < proximitat_maxima_frontal && let.Right <
proximitat_minima_dreta){  //Seguim avançant fins topar-nos amb una paret.
                Reset_TimeoutA0();
                while(!TimeOutA0(TEMPS_ESPERA_LECTURA));
                let = leer_sensores();
            }
            caso_3_derecha(); //Ens trobem en el cas 3, així que el cridem.
        }
    }
}
/*****
* CASO_3_DERECHA
* El robot segueix la dreta i té una paret davant
*****/
void caso_3_derecha(){
    pivotar_sobre_si_mismo_antihorario(VELOCITAT); //Pivotem antihorariament
fins tenir la paret de davant a la dreta.
```

```
    while(let.Center >= proximitat_maxima_frontal || let.Right <
proximitat_minima_dreta){
        Reset_TimeoutA0();
        while(!TimeOutA0(TEMPS_ESPERA_LECTURA));
        let = leer_sensores();
        }
}
/*****
* CASO_4_DERECHA
* El robot segueix la paret de la dreta i ha perdut de cop de detectar la paret
****/
void caso_4_derecha(){
    avanzar_con_giro(VELOCITAT,VELOCITAT/2); //Seguim avançant amb un gir fins
que trobem paret a la dreta.
    while(let.Right < proximitat_minima_dreta){
        if(let.Center >= proximitat_maxima_frontal){
            caso_3_derecha(); //Ens trobem en el cas 3, així que el cridem.
        }
        Reset_TimeoutA0();
        while(!TimeOutA0(TEMPS_ESPERA_LECTURA));
        let = leer_sensores();
    }
}
/*****
* CASO_5_DERECHA
* El robot segueix la paret de la dreta i s'està allunyant massa de la paret
****/
void caso_5_derecha(){
    avanzar_con_giro(VELOCITAT,VELOCITAT/2); //Avancem girant una mica cap a la
dreta per apropar-nos.
}
/*****
* CASO_6_DERECHA
* El robot segueix la paret de la dreta i s'està apropant massa a la paret
****/
void caso_6_derecha(){
    avanzar_con_giro(VELOCITAT/5,VELOCITAT); //Avancem girant bastant cap a
l'esquerra per allunyar-nos.
}
/*****
* CASO_1_IZQUIERDA
* El robot segueix la paret de l'esquerra i està dintre els valors de distància
correctes
*****/
void caso_1_izquierda(){
    avanzar(VELOCITAT);  //Tot correcte, avancem.
}
/*****
* CASO_2_IZQUIERDA
```

```
* El robot té paret a l'esquerra, dreta i davant
****/
void caso_2_izquierda(){
    //Retrocedim fins que no tinguem paret a un costat, cuidant no col·lisionar
amb cap paret.
    while(let.Right >= proximitat_no_paret && let.Left >= proximitat_no_paret){
        if(let.Right >= proximitat_maxima_dreta && let.Left <
proximitat_maxima_esquerra ){
            retroceder_con_giro(VELOCITAT/1.5, VELOCITAT);
        }
        else if(let.Right < proximitat_maxima_dreta && let.Left >=
proximitat_maxima_esquerra ){
            retroceder_con_giro(VELOCITAT, VELOCITAT/1.5);
        }
        else{
            retroceder(VELOCITAT);
        }
        Reset_TimeoutA0();
        while(!TimeOutA0(TEMPS_ESPERA_LECTURA));
        let = leer_sensores();
    }
    //Si deixem de tenir paret a la dreta, girem uns 90 graus cap a la dreta i
avancem.
    if(let.Right < proximitat_no_paret){
        pivotar_sobre_si_mismo_horario(VELOCITAT);
        while(!TimeOutA0(30000)); //0.5s
        let = leer_sensores();
        avanzar(VELOCITAT);
        if(let.Left < proximitat_minima_esquerra){ //Si no detectem paret a
l'esquerra...
            while(let.Center < proximitat_maxima_frontal  && let.Left <
proximitat_minima_esquerra){  //Seguim avançant fins topar-nos amb una paret.
                Reset_TimeoutA0();
                while(!TimeOutA0(TEMPS_ESPERA_LECTURA));
                let = leer_sensores();
            }
            caso_3_izquierda(); //Ens trobem en el cas 3, així que el cridem.
        }
    }
    //Si deixem de tenir paret a l'esquerra, retrocedim girant mig segon i
avancem.
    else{
        retroceder_con_giro(0, VELOCITAT);
        while(!TimeOutA0(30000)); //0.5s
        let = leer_sensores();
        avanzar(VELOCITAT);
        if(let.Left < proximitat_minima_esquerra){  //Si no detectem paret a
l'esquerra...
```

```
            while(let.Center < proximitat_maxima_frontal && let.Left <
proximitat_minima_esquerra){  //Seguim avançant fins topar-nos amb una paret.
                Reset_TimeoutA0();
                while(!TimeOutA0(TEMPS_ESPERA_LECTURA));
                let = leer_sensores();
            }
        caso_3_izquierda(); //Ens trobem en el cas 3, així que el cridem.
        }

    }
}
/*****
* CASO_3_IZQUIERDA
* El robot segueix la paret de l'esquerra i té una paret davant però no a la
dreta
*****/
void caso_3_izquierda(){
    pivotar_sobre_si_mismo_horario(VELOCITAT); //Pivotem horariament fins tenir
la paret de davant a l'esquerra.
    while(let.Center >= proximitat_maxima_frontal || let.Left <
proximitat_minima_esquerra){
        Reset_TimeoutA0();
        while(!TimeOutA0(TEMPS_ESPERA_LECTURA));
        let = leer_sensores();
    }
}
/*****
* CASO_4_IZQUIERDA
* El robot segueix la paret de l'esquerra i ha perdut de cop de detectar la
paret
*****/
void caso_4_izquierda(){ //Seguim avançant amb un gir fins que trobem paret a
l'esquerra.
    avanzar_con_giro(VELOCITAT/2,VELOCITAT);
    while(let.Left < proximitat_minima_esquerra){
        if(let.Center >= proximitat_maxima_frontal){
            caso_3_izquierda(); //Ens trobem en el cas 3, així que el cridem.
        }
        Reset_TimeoutA0();
        while(!TimeOutA0(TEMPS_ESPERA_LECTURA));
        let = leer_sensores();
        }
}
/*****
* CASO_5_IZQUIERDA
* El robot segueix la paret de l'esquerra i s'està allunyant massa de la paret
*****/
void caso_5_izquierda(){
```

```
    avanzar_con_giro(VELOCITAT/2,VELOCITAT); //Avancem girant una mica cap a
l'esquerra per apropar-nos.
}
/*****
* CASO_6_IZQUIERDA
* El robot segueix la paret de l'esquerra i s'està apropant massa a la paret
*****/
void caso_6_izquierda(){
    avanzar_con_giro(VELOCITAT,VELOCITAT/5); //Avancem girant cap a la dreta per
allunyar-nos.
}
/*****
* MAIN
*****/
void main(void){
    WDT_A->CTL = WDT_A_CTL_PW | WDT_A_CTL_HOLD; // Stop watchdog timer
    init_ucs_24MHz();
    init_timers(); // Configurem els Timers
    init_timer_main();
    initUART(); //Inicialitzem la eUSCI Ax com a UART
    init_botons();
    init_interrupciones(); //Configurem i activem les interrupcions (TA1 i
eUSCIAx)
    init_interrupcion_main();
    __enable_interrupts(); //Activem les interrupcions a nivell global
    enableTorque();
    configurar_CONT_MODE();
    parar(); //por si hacemos reset
    Activa_TimerA0_TimeOut(); //Activem el timer A0
    while(!calibracio_feta){
        let = leer_sensores();
        Reset_TimeoutA0();
        while(!TimeOutA0(TEMPS_ESPERA_LECTURA));
    }
    buzzer(6, 255); //calibrat
    Reset_TimeoutA0();
    while(!TimeOutA0(500000));
    while(true){ //Bucle infinit
        while(!palmada()); //Esperem a rebre una palmada per començar
        melodia(); //Fem sonar una melodia abans de començar
        trobar_paret(); //Trobem una paret
        parar_robot = false; //Posem aquesta variable global a false
        while(!parar_robot){ //Mentres segueixi a false, el robot es mou
            moviment_robot_infinit();
        }
        parar(); //Si sortim del bucle, parem el robot
        melodia2(); //Fem sonar una melodia de final d'execució
    }
}
```

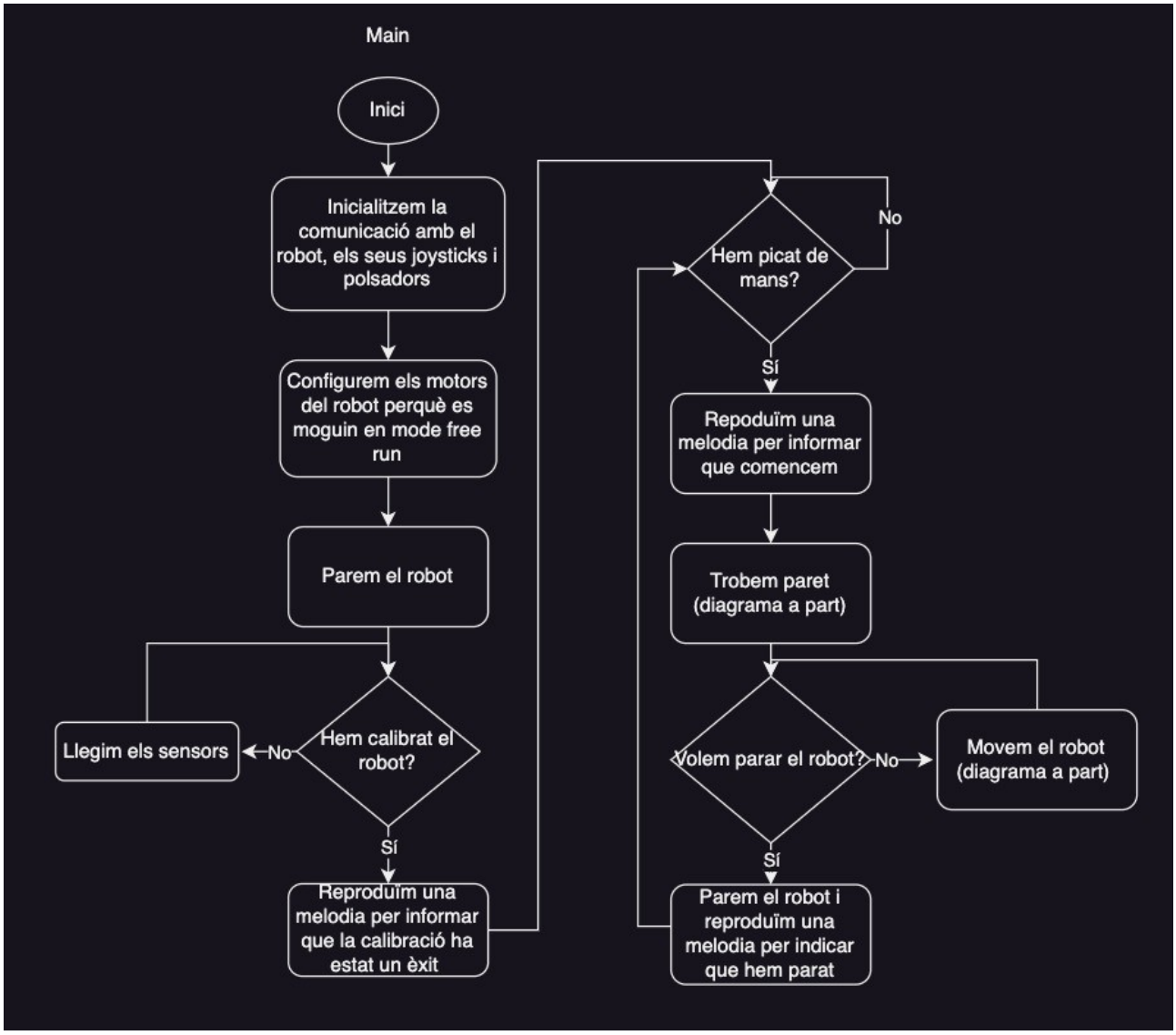
```
/*****
* TA0_0_IRQHANDLER
* Rutina d'atenció a la interrupció del timer A0
*****/
void TA0_0_IRQHandler(void){
    TA0CCTL0 &= ~TIMER_A_CCTLN_CCIFG; //Clear interrupt flag
    cont_time_out++; //Sumem 1 al comptador per controlar el timeout
}
/****
* PORT3_IRQHANDLER
* Rutina d'atenció a la interrupció del port 3
****/
void PORT3_IRQHandler(void){
    uint8_t flag = P3IV; //guardamos el vector de interrupciones. De paso, al
acceder a este vector, se limpia automaticamente.
    P3IE &= ~(BIT5); //interrupciones del pulsador en port 3 desactivadas
    switch (flag){
        case 0x0C: //Si la interrupció ha estat generada pel polsador S2, es fa
una recalibració
            if(!calibracio_feta){
                if(let.Left +5 < 255 && let.Right +5 < 255 && let.Left-5 > 0 &&
let.Right-5 >0 && let.Center > 0 && let.Center < 255){
                proximitat_maxima_frontal = let.Center;
                proximitat_minima_esquerra = let.Left - 5;
                proximitat_maxima_esquerra = let.Left + 5;
                proximitat_minima_dreta = let.Right - 5;
                proximitat_maxima_dreta = let.Right + 5;
                proximitat_no_paret = proximitat_minima_dreta/2;
                if(proximitat_no_paret > proximitat_minima_esquerra/2){
                    proximitat_no_paret > proximitat_minima_esquerra/2;
                }
                calibracio_feta = true;
                }
            }
            break;
        default:
            break;
    }
    P3IE |= (BIT5); //interrupciones del pulsador en port 3 reactivadas
}
/****
* PORT4_IRQHANDLER
* Rutina d'atenció a la interrupció del port 4
****/
void PORT4_IRQHandler(void){
    uint8_t flag = P4IV; //guardamos el vector de interrupciones. De paso, al
acceder a este vector, se limpia automaticamente.
    P4IE &= ~(BIT5 + BIT7); //interrupciones del pulsador en port 3 desactivadas
    switch (flag){
```

```
        case 0x0C: //Si la interrupció ha estat generada pel joystick de la
dreta, pugem 5u les calibracions dels costats
            if(proximitat_maxima_dreta+5 < 255 && proximitat_maxima_esquerra+5 <
255){
            proximitat_maxima_dreta += 5;
            proximitat_minima_dreta += 5;
            proximitat_maxima_esquerra += 5;
            proximitat_minima_esquerra += 5;
            proximitat_no_paret = proximitat_minima_esquerra/2;
            }
            break;
        case 0x10: //Si la interrupció ha estat generada pel joystick de
l'esquerra, baixem 5u les calibracions dels costats
            if(proximitat_maxima_dreta-5 > 0 && proximitat_maxima_esquerra-5 > 0
|| proximitat_minima_esquerra/2 > 0 ){
            proximitat_maxima_dreta -= 5;
            proximitat_minima_dreta -= 5;
            proximitat_maxima_esquerra -= 5;
            proximitat_minima_esquerra -= 5;
            proximitat_no_paret = proximitat_minima_esquerra/2;
            }
            break;
        default:
            break;
    }
    P4IE |= (BIT5 + BIT7); //interrupciones del pulsador en port 3 reactivadas
}
/****
* PORT5_IRQHANDLER
* Rutina d'atenció a la interrupció del port 5
****/
void PORT5_IRQHandler(void){
    uint8_t flag = P5IV; //guardamos el vector de interrupciones. De paso, al
acceder a este vector, se limpia automaticamente.
    P5IE &= ~(BIT1 + BIT4 + BIT5); //interrupciones del pulsador en port 5
desactivadas
    switch (flag){
        case 0x04: //Si la interrupció ha estat generada pel polsador S1
            parar_robot = true; //Parem el robot
            break;
        case 0x0A: //Si la interrupció ha estat generada pel joystick d'amunt,
pugem 5u la calibracio de davant
            if(proximitat_maxima_frontal+5 < 255){
            proximitat_maxima_frontal += 5;
            }
            break;
        case 0x0C: //Si la interrupció ha estat generada pel joystick d'avall,
baixem 5u la calibracio de davant
            if(proximitat_maxima_frontal-5 > 0){
```

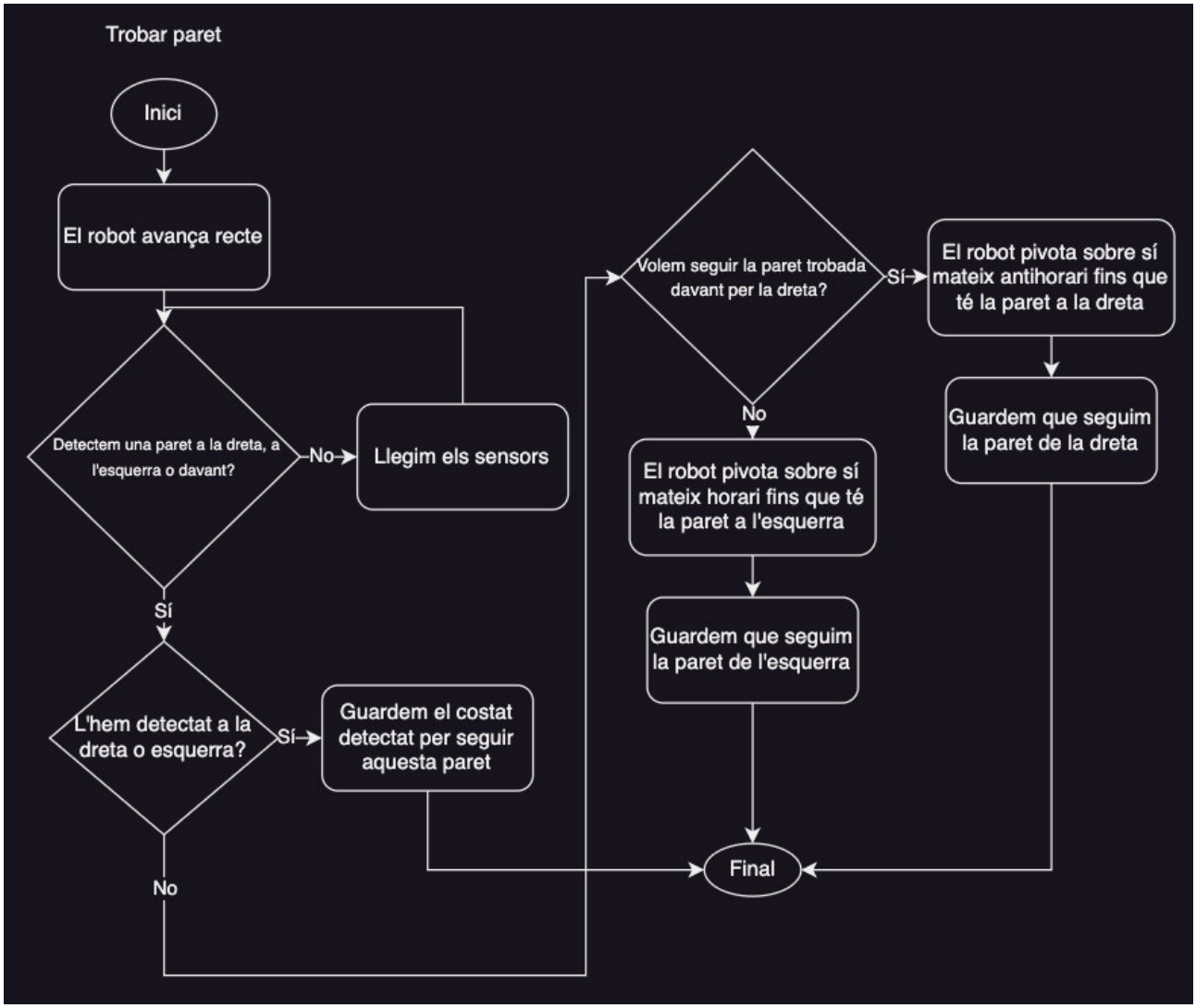
```
            proximitat_maxima_frontal -= 5;
            }
            break;
        default:
            break;
    }
    P5IE |= (BIT1 + BIT4 + BIT5); //interrupciones del pulsador en port 5
reactivadas
}
```

6.2 Diagrames de Flux

6.2.1 Main()

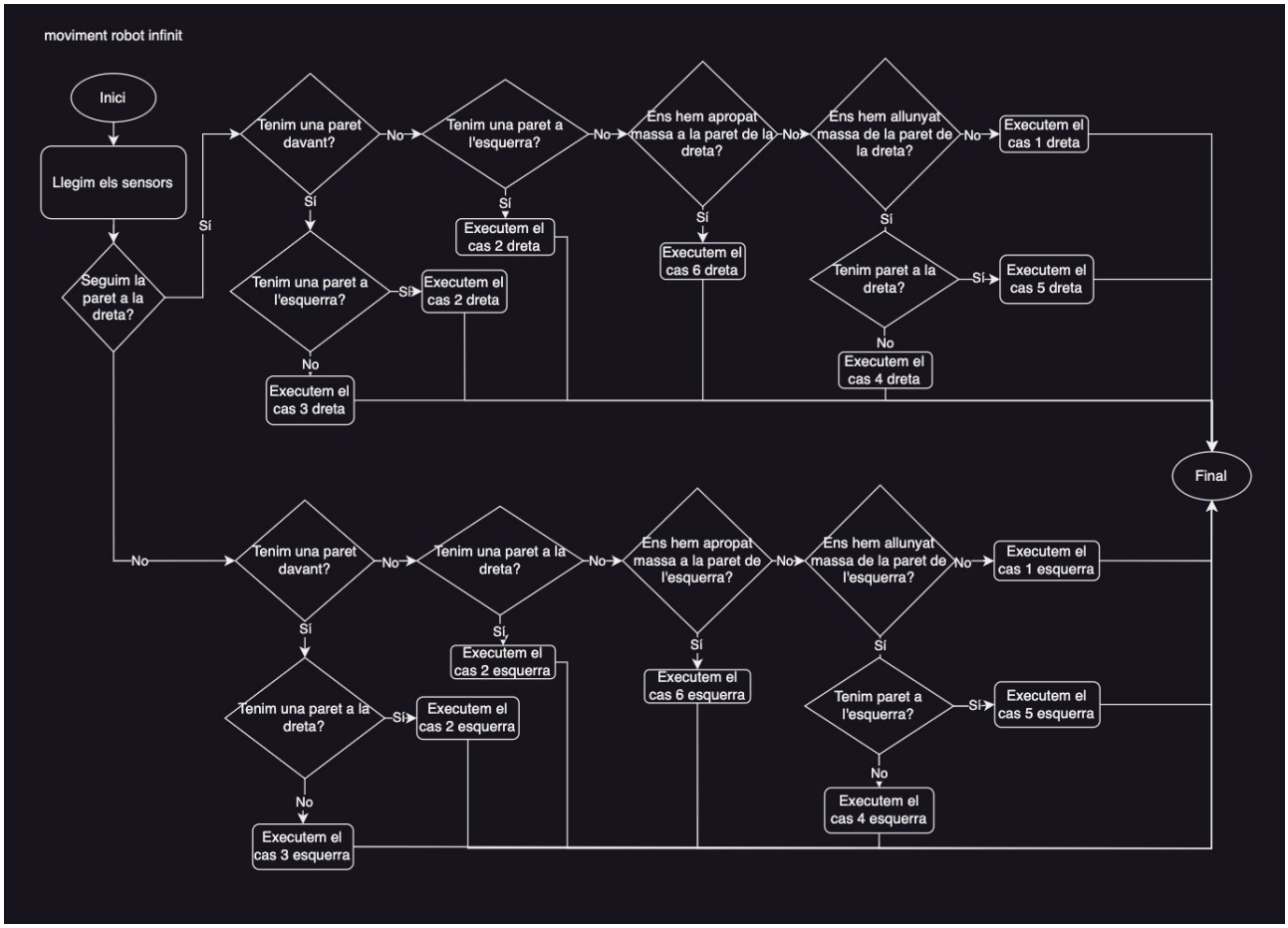


6.2.2 Trobar_paret()



6.2.3 Moviment_robot_infinit()

David Fernàndez
Víctor Sort



6.2.4 nterrupcions joystick i polsadors

David Fernàndez
Víctor Sort