Projecte, Robot

Programació d'Arquitectures Encastades

Ivan Torres Perez
Astor Prieto Dehghan Pour

Introducció al projecte

- Finalitat del robot
- Elements usats al projecte

Parts i elements usats del robot

- Ports d'I/O utilitzats
- Timers
- UCS
- UART

Vectors d'Interrupció i IHRs

- Interrupció dels Timers (A1 i B0)
- Interrupció de la USCI
- Interrupció dels Botons (Port2)

Rellotge del sistema i el UCS

Comunicació amb UART i els Mòduls

Llibreries i estructura de fitxers

- “llibreria_config.h”, “llibreria_config.c”
- “llibreria_robot.h”, “llibreria_robot.c”

“llibreria_config.h”

```c
/*
 * libreria_config.h
 *
 *  Created on: 06/05/2015
 *      Author: mat.aules
 */

#include <msp430x54xA.h>
#include <stdio.h>
#include "hal_lcd.h"
#include "libreria_robot.h"

#ifndef LIBRERIA_CONFIG_H_
#define LIBRERIA_CONFIG_H_

/*
 * Funcions per a configurar totes les parts del microcontrolador.
 */

void init_botons(void);

void config_P4_LEDS(void);

void init_LCD(void);

void init_timers(void);

void init_UCS(void);

void init_UART(void);

void config_motores(void);


#endif /* LIBRERIA_CONFIG_H_ */

```

“llibreria_robot.h”

```c
/*
 * libreria_robot.h
 *
 *  Created on: 06/05/2015
 *      Author: mat.aules
 */

#include <msp430x54xA.h>
#include <stdio.h>
#include "hal_lcd.h"

#ifndef LIBRERIA_ROBOT_H_
#define LIBRERIA_ROBOT_H_

typedef unsigned char byte;             //DEFINIM EL TIPUS BYTE
#define TXD0_READY (UCA0IFG & UCTXIFG)
#define RXD0_READY (UCA0IFG & UCRXIFG)

typedef struct {
    byte Packet[32];            //DEFINIM EL PAQUET QUE RETORNARA RxPacket()
} RxRes;

/* FUNCIONES DE COMUNICACION CON EL ROBOT */

void Sentit_Dades_Rx(void);

void Sentit_Dades_Tx(void);

void TxUAC0(byte bTxdData);

byte TxPacket(byte bID, byte bParameterLength, byte bInstruction);

RxRes RxPacket(void);

/* FUNCIONES PARA MOVER LOS MOTORES */

void mov_adelante(byte velocidad);

void mov_atras(byte velocidad);

void giro_derecha(void);

void giro_derecha_mod(void);

void giro_izquierda(void);

void giro_izquierda_mod(void);

void halt_motores(void);

/* FUNCIONES PARA LEER LOS SENSORES */

RxRes leer_sensor_frontal(void);

RxRes leer_sensor_derecho(void);

RxRes leer_sensor_izquierdo(void);

RxRes leer_microfono(void);

/* FUNCIONES PARA REPRODUCIR SONIDOS */

void reproducir_nota(byte nota, byte duracion);

#endif /* LIBRERIA_ROBOT_H_ */
```

Comunicació amb els Mòduls

```
26
27 void Sentit_Dades_Rx(void)
28 {
29     //Configuració del Half Duplex dels motors: Recepció
30     P3OUT &= ~0x80; //El pin P3.7 (DIRECTION_PORT) el posem a 0 (Rx)
31 }
32
33 void Sentit_Dades_Tx(void)
34 {
35     //Configuració del Half Duplex dels motors: Transmissió
36     P3OUT |= 0x80; //El pin P3.7 (DIRECTION_PORT) el posem a 1 (Tx)
37 }
38
39 /* funció TxUAC0(byte): envia un byte de dades per la UART 0 */
40 void TxUAC0(byte bTxdData)
41 {
42     while(!TXD0_READY); // Espera a que estigui preparat el buffer de transmissió
43     UCA0TXBUF = bTxdData; // Pone el byte del buffer de la UART del micro en el bus de transmision
44 }
45
```

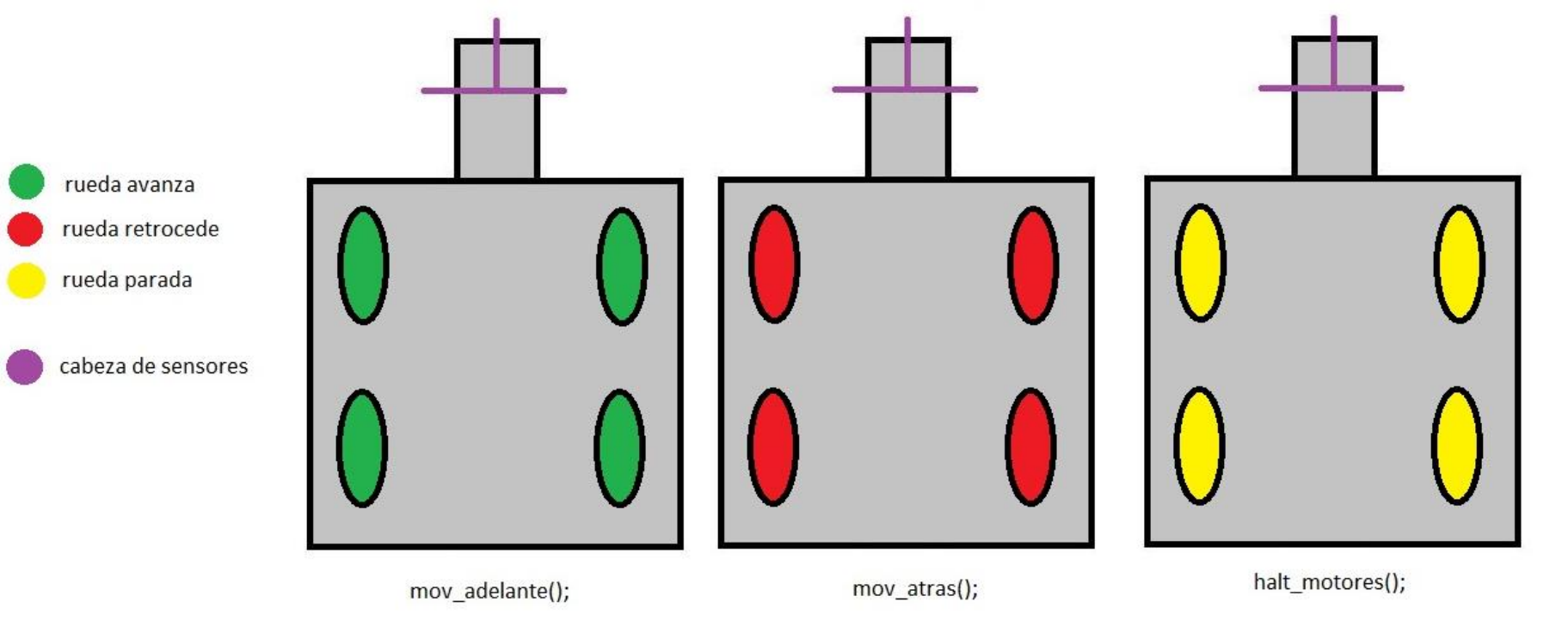
“TxPacket()”

```
/*
TxPacket() necessita 3 paràmetres; ID del Dynamixel, Instruction byte, Mida dels paràmetres.
TxPacket() torna la mida del "Return packet" des del Dynamixel.
*/
byte TxPacket(byte bID, byte bParameterLength, byte bInstruction)
{
    byte bCount,bCheckSum,bPacketLength;
    byte gbpTxBuffer[32];
    volatile int i = 50;
    char error[] = "adr. no permitida";
    if ((gbpParameter[0]<5)&&(bInstruction==3)){        //Si intentem escriure en posicions conflictives del robot
        //si se intenta escribir en una direccion <= 0x05,
        //emitir mensaje de error de direccion prohibida:
        halLcdPrintLine( error, 5, OVERWRITE_TEXT);
        //y salir de la funcion sin mas:
        return 0;
    }
    Sentit_Dades_Tx(); //El pin P3.7 (DIRECTION_PORT) el posem a 1 (Tx)
    gbpTxBuffer[0] = 0xFF; //Primers 2 bytes que indiquen inici de trama FF, FF.
    gbpTxBuffer[1] = 0xFF;
    gbpTxBuffer[2] = bID; //ID del mòdul al que volem enviar el missatge
    gbpTxBuffer[3] = bParameterLength+2; //Length(Parameter,Instruction,Checksum)
    gbpTxBuffer[4] = bInstruction; //Instrucció que enviem al mòdul

    for(bCount = 0; bCount < bParameterLength; bCount++)//Comencem a generar la trama
    {
        gbpTxBuffer[bCount+5] = gbpParameter[bCount];
    }
    bCheckSum = 0;
    bPacketLength = bParameterLength+4+2;

    for(bCount = 2; bCount < bPacketLength-1; bCount++) //Càlcul del Checksum
    {
        bCheckSum += gbpTxBuffer[bCount];
    }
    gbpTxBuffer[bCount] = ~bCheckSum; //Escriu el Checksum (complement a 1)

    for(bCount = 0; bCount < bPacketLength; bCount++) //Aquest bucle és el que envia la trama
    {
        TxUAC0(gbpTxBuffer[bCount]);
    }
    while((UCA0STAT&UCBUSY)); //Espera fins s'ha transmès el últim byte
    Sentit_Dades_Rx(); //Posem la línia de dades en Rx perquè el mòdul Dynamixel envia resposta
    return(bPacketLength);
}

```

“RxPacket()”

```
/*
RxPacket() necessita 0 paràmetres.
RxPacket() retorna el "Status packet" demanat des del Dynamixel.
*/
RxRes RxPacket(void)
{
    RxRes respuesta;        // Variable on guardem el packet de resposta del robot
    byte bCount, bLength, bCheckSum;
    bCheckSum = 0;          // Inicialitzacio del CheckSum
    timeOut = 1000;
    byte_recibido = 0;      // Netejem el flag de byte_recibido de la UART
    Sentit_Dades_Rx();

    for (bCount = 0; bCount < 4; bCount++) {    //Rebem els primers 4 bytes de la trama
        timeOut = 1000;
        while (!byte_recibido) {        //Esperem a la interrupcio de la UART
            if (timeOut <= 0)break;     //Si es passa el temps del TimeOut, acabem la funció
        }
        if (timeOut <= 0) {
            halLcdPrintLine("Error TimeOut", 4, OVERWRITE_TEXT);    //Amb un missatge de error de TimeOut
            return respuesta;
        }
        byte_recibido = 0;
        respuesta.Packet[bCount] = byte_UART;   //En cas de "no TimeOut", rebem el byte de la UART
    }
    bLength = respuesta.Packet[3] + 4;      //Trobem el tamany total de la trama amb el bLength
    for (bCount = 4; bCount < bLength; bCount++) {  //Rebem la resta de bytes de la trama
        timeOut = 1000;
        while (!byte_recibido) {        //Esperem a la interrupcio de la UART
            if (timeOut <= 0)break;     //Si es passa el temps del TimeOut, acabem la funció
        }
        if (timeOut <= 0) {
            halLcdPrintLine("Error TimeOut", 4, OVERWRITE_TEXT);    //Amb un missatge de error de TimeOut
            return respuesta;
        }
        byte_recibido = 0;
        respuesta.Packet[bCount] = byte_UART;   //En cas de "no TimeOut", rebem el byte de la UART
    }
    for(bCount = 2; bCount < bLength - 1; bCount++) { //Càlcul del Checksum
        bCheckSum += respuesta.Packet[bCount];
    }
    bCheckSum = bCheckSum + respuesta.Packet[bLength - 1]; //Si el CheckSum es correcto, esto dara 0xFF
    if (bCheckSum != 0xFF) {
        halLcdPrintLine("Error CheckSum", 4, OVERWRITE_TEXT);       //Si hi ha error de CheckSum, es mostra un mis
        return respuesta;
    }
    return respuesta;
}
```

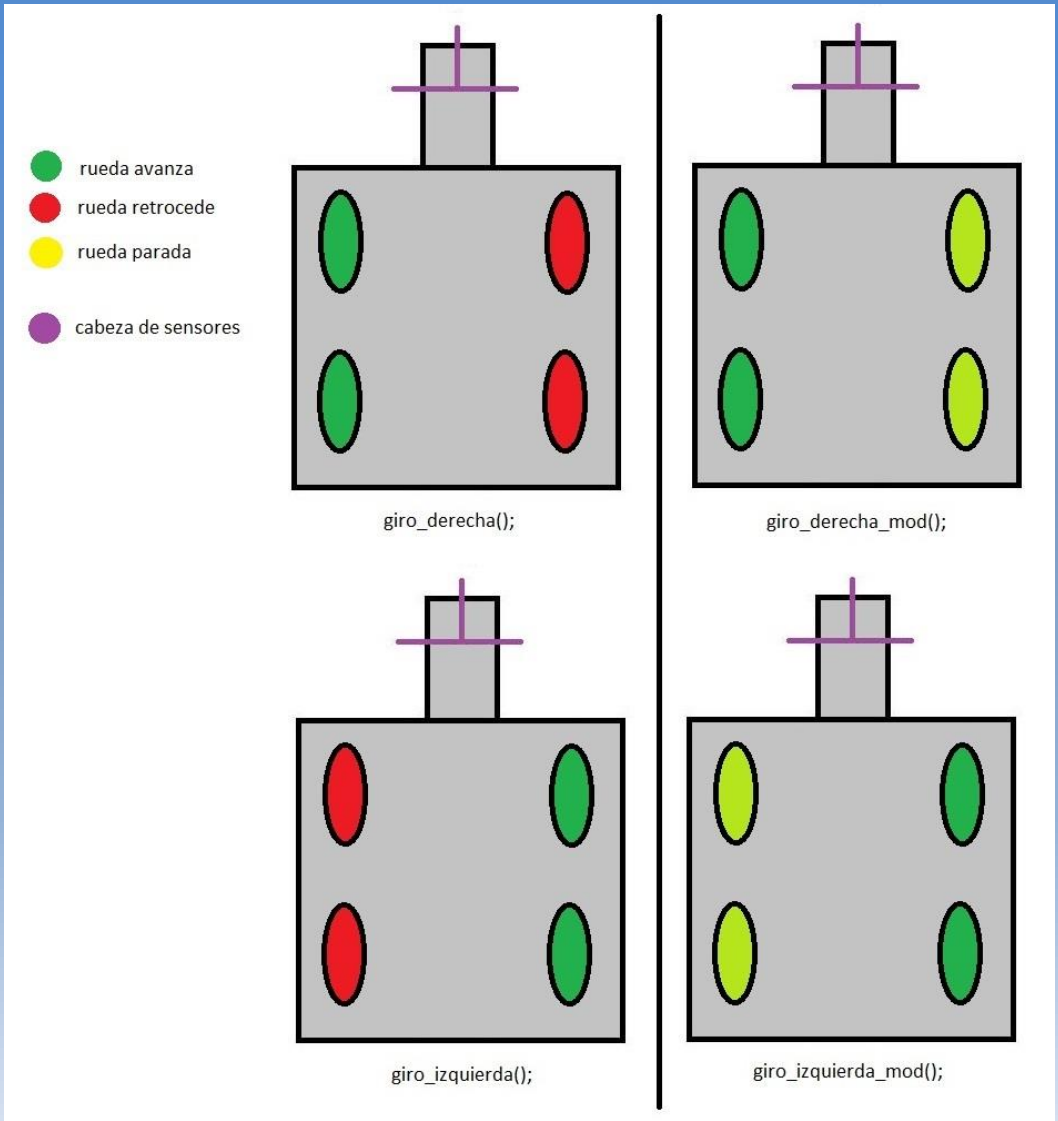
Moviments disponibles del Robot



```
/* Mover para adelante */
void mov_adelante(byte velocidad)
{
    gbpParameter[0] = 0x20;         // Escribimos en las posiciones de memoria 20 y 21 (Velocidad del motor)
    gbpParameter[1] = velocidad;    // Pasamos por parametro la velocidad a poner en el motor
    gbpParameter[2] = 0x04;
    TxPacket(0x01, 3, 0x04);        // Enviamos REG_WRITE al motor 1
    paquete = RxPacket();
    TxPacket(0x02, 3, 0x04);        // Enviamos REG_WRITE al motor 2
    paquete = RxPacket();

    gbpParameter[0] = 0x20;         // Hacemos lo mismo con los otros dos motores, pero en sentido contrario
    gbpParameter[1] = velocidad;    // ya que estan girados fisicamente
    gbpParameter[2] = 0x00;
    TxPacket(0x03, 3, 0x04);        // Enviamos REG_WRITE al motor 3
    paquete = RxPacket();
    TxPacket(0x04, 3, 0x04);        // Enviamos REG_WRITE al motor 4
    paquete = RxPacket();

    TxPacket(0xFE, 0, 0x05);        // Instruccion ACTION con el ID de Broadcast para iniciar el movimiento hacia
}

/* Mover para atras */
void mov_atras(byte velocidad)
{
    gbpParameter[0] = 0x20;         // Escribimos en las posiciones de memoria 20 y 21 (Velocidad del motor)
    gbpParameter[1] = velocidad;    // Pasamos por parametro la velocidad a poner en el motor
    gbpParameter[2] = 0x00;
    TxPacket(0x01, 3, 0x04);
    paquete = RxPacket();
    TxPacket(0x02, 3, 0x04);
    paquete = RxPacket();

    gbpParameter[0] = 0x20;
    gbpParameter[1] = velocidad;
    gbpParameter[2] = 0x04;
    TxPacket(0x03, 3, 0x04);
    paquete = RxPacket();
    TxPacket(0x04, 3, 0x04);
    paquete = RxPacket();

    TxPacket(0xFE, 0, 0x05);
}
```

```
/* PARAR LOS MOTORES */
void halt_motores(void)
{
    gbpParameter[0] = 0x20;         // Escribimos en las posiciones de memoria 20 y 21 (Velocidad del motor)
    gbpParameter[1] = 0x00;         // Pasamos por parametro la velocidad a poner en el motor
    gbpParameter[2] = 0x00;
    TxPacket(0x01, 3, 0x04);        // Enviamos REG_WRITE al motor 1
    paquete = RxPacket();
    TxPacket(0x02, 3, 0x04);        // Enviamos REG_WRITE al motor 2
    paquete = RxPacket();

    gbpParameter[0] = 0x20;         // Hacemos lo mismo con los otros dos motores, pero en sentido contrario
    gbpParameter[1] = 0x00;         // ya que estan girados fisicamente
    gbpParameter[2] = 0x00;
    TxPacket(0x03, 3, 0x04);        // Enviamos REG_WRITE al motor 3
    paquete = RxPacket();
    TxPacket(0x04, 3, 0x04);        // Enviamos REG_WRITE al motor 4
    paquete = RxPacket();

    TxPacket(0xFE, 0, 0x05);        // Instruccion ACTION con el ID de Broadcast que para todo el movimiento
}
```

Moviments disponibles del Robot II

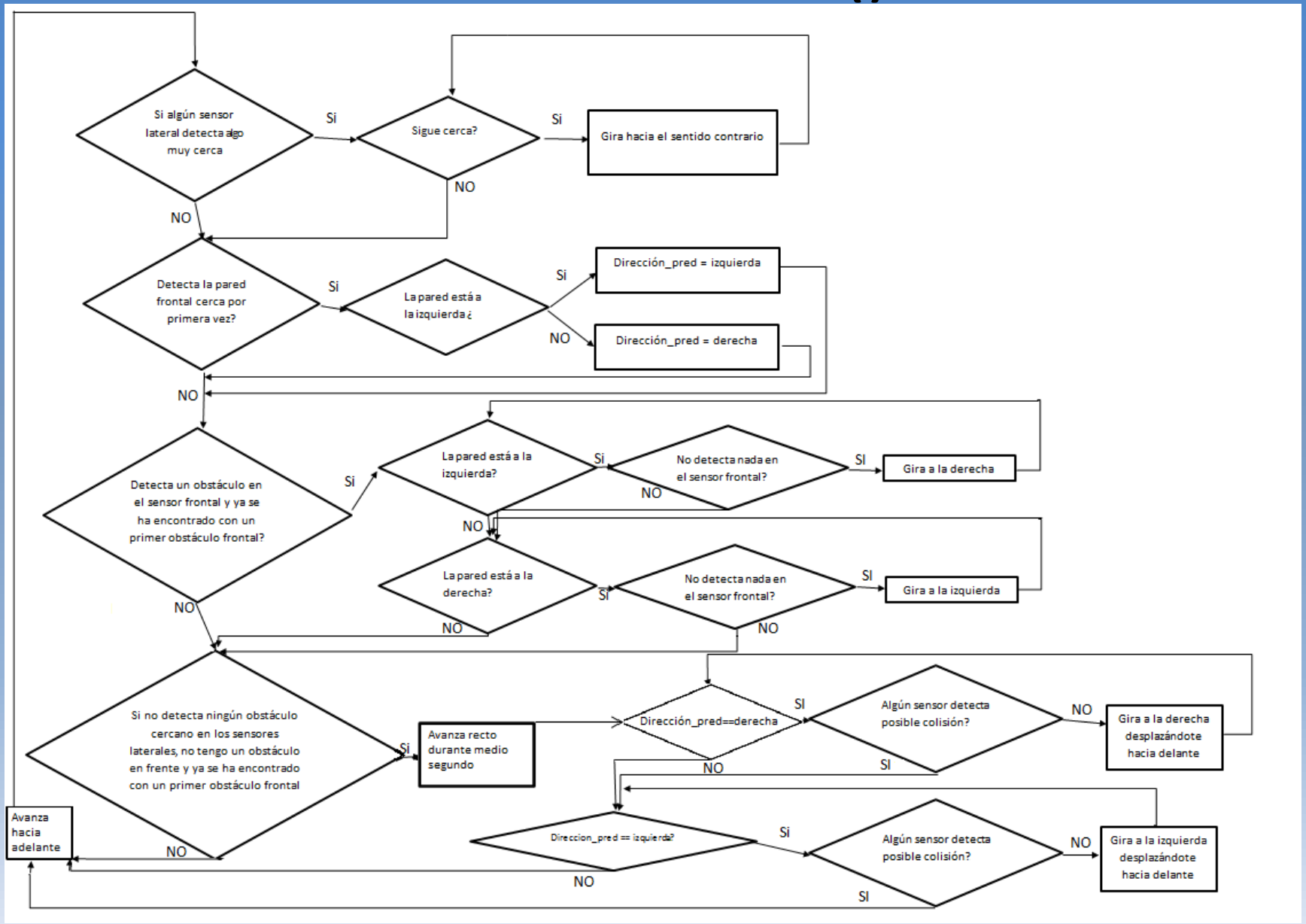


```
195
196 /* Giro hacia la derecha
197 void giro_derecha(void)
198 {
199     gbpParameter[0] = 0x20;
200     gbpParameter[1] = 0xFF;
201     gbpParameter[2] = 0x00;
202     TxPacket(0x01, 3, 0x04);
203     paquete = RxPacket();
204     TxPacket(0x02, 3, 0x04);
205     paquete = RxPacket();
206
207     gbpParameter[0] = 0x20;
208     gbpParameter[1] = 0xFF;
209     gbpParameter[2] = 0x00;
210     TxPacket(0x03, 3, 0x04);
211     paquete = RxPacket();
212     TxPacket(0x04, 3, 0x04);
213     paquete = RxPacket();
214
215     TxPacket(0xFE, 0, 0x05);
216 }
217
218 /* Giro para la derecha
219 void giro_derecha_mod(void)
220 {
221     gbpParameter[0] = 0x20;
222     gbpParameter[1] = 0x70;
223     gbpParameter[2] = 0x04;
224     TxPacket(0x01, 3, 0x04);
225     paquete = RxPacket();
226     TxPacket(0x02, 3, 0x04);
227     paquete = RxPacket();
228
229     gbpParameter[0] = 0x20;
230     gbpParameter[1] = 0xFF;
231     gbpParameter[2] = 0x00;
232     TxPacket(0x03, 3, 0x04);
233     paquete = RxPacket();
234     TxPacket(0x04, 3, 0x04);
235     paquete = RxPacket();
236
237     TxPacket(0xFE, 0, 0x05);
238 }
239
```

```
240 /* Giro hacia la izquierda
241 void giro_izquierda(void)
242 {
243     gbpParameter[0] = 0x20;
244     gbpParameter[1] = 0xFF;
245     gbpParameter[2] = 0x04;
246     TxPacket(0x01, 3, 0x04);
247     paquete = RxPacket();
248     TxPacket(0x02, 3, 0x04);
249     paquete = RxPacket();
250
251     gbpParameter[0] = 0x20;
252     gbpParameter[1] = 0xFF;
253     gbpParameter[2] = 0x04;
254     TxPacket(0x03, 3, 0x04);
255     paquete = RxPacket();
256     TxPacket(0x04, 3, 0x04);
257     paquete = RxPacket();
258
259     TxPacket(0xFE, 0, 0x05);
260 }
261
262 /* Giro para la izquierda
263 void giro_izquierda_mod(void)
264 {
265     gbpParameter[0] = 0x20;
266     gbpParameter[1] = 0xFF;
267     gbpParameter[2] = 0x04;
268     TxPacket(0x01, 3, 0x04);
269     paquete = RxPacket();
270     TxPacket(0x02, 3, 0x04);
271     paquete = RxPacket();
272
273     gbpParameter[0] = 0x20;
274     gbpParameter[1] = 0x70;
275     gbpParameter[2] = 0x00;
276     TxPacket(0x03, 3, 0x04);
277     paquete = RxPacket();
278     TxPacket(0x04, 3, 0x04);
279     paquete = RxPacket();
280
281     TxPacket(0xFE, 0, 0x05);
282 }
283
```

Heurística del Robot

Funció “main()”


Si algún sensor lateral detecta algo muy cerca
Si
Sigue cerca?
Si
Gira hacia el sentido contrario
NO
NO
Detecta la pared frontal cerca por primera vez?
Si
La pared está a la izquierda¿
Si
Dirección_pred = izquierda
NO
Dirección_pred = derecha
NO
Detecta un obstáculo en el sensor frontal y ya se ha encontrado con un primer obstáculo frontal?
Si
La pared está a la izquierda?
Si
No detecta nada en el sensor frontal?
SI
Gira a la derecha
NO
NO
La pared está a la derecha?
SI
No detecta nada en el sensor frontal?
SI
Gira a la izquierda
NO
NO
Si no detecta ningún obstáculo cercano en los sensores laterales, no tengo un obstáculo en frente y ya se ha encontrado con un primer obstáculo frontal
Si
Avanza recto durante medio segundo
Dirección_pred==derecha
SI
Algún sensor detecta posible colisión?
NO
Gira a la derecha desplazándote hacia delante
NO
SI
Direccion_pred == izquierda?
Si
Algún sensor detecta posible colisión?
NO
Gira a la izquierda desplazándote hacia delante
NO
SI
Avanza hacia adelante
NO

Extres Implementats

- Lectura del micròfon (inici amb palmades)
- Reproducció de sons amb el Mòdul AX-S1
- Funcionalitats vinculades als botons d'E/S

Preguntes del Projecte