Universidad de Alcalá
Escuela Politécnica Superior

Grado en Ingeniería en Electrónica y Automática Industrial

Trabajo Fin de Grado

Diseño de un robot autónomo. Posicionamiento por balizas y comunicación interna del robot.

Autor: Rodrigo Gutiérrez Moreno

Tutor/es: Julio Pastor Mendoza

2017

UNIVERSIDAD DE ALCALÁ

Escuela Politécnica Superior

Grado en Ingeniería en Electrónica y Automática Industrial

Trabajo Fin de Grado

Diseño de un robot autónomo. Posicionamiento por balizas y comunicación interna del robot.

Autor: Rodrigo Gutiérrez Moreno

Tutor/es: Julio Pastor Mendoza

TRIBUNAL:

Presidente: José Manuel Villadangos Carrizo

Vocal 1º: Ernesto Martín Gorostiza

Vocal 2º: Julio Pastor Mendoza

FECHA: 16/10/2017

Este trabajo fin de grado está dedicado a mi familia, en especial a mi madre, mi padre, mi hermano y mis abuelos, por su apoyo y por la educación que me han dado; a los integrantes del equipo Miguel, Gonzalo y Mario, por su ayuda; a mis compañeros de grado Eduardo, Juan, Pablo y Sergio, ya que gracias a ellos el camino ha sido más fácil; a Julio, por su paciencia y ayuda en todo momento; y a José Manuel y a Pedro por sus aportaciones al proyecto.

Índice

Lista de Figuras

Lista de tablas

Resumen

Este proyecto surge de la idea de crear un equipo formado por estudiantes, cuyo objetivo es participar en un concurso internacional de robótica. El desafío es diseñar un robot autónomo. En este trabajo se refleja el estudio, diseño y realización del posicionamiento por balizas y la comunicación interna.

En este documento se tratan los dos temas principales. La comunicación interna del robot se desarrolla primeramente utilizando el protocolo BUS CAN, pero finalmente se sustituye por comunicación a través del puerto serie. El posicionamiento por balizado proporciona tanto la posición del robot dentro del campo, como la posición del robot oponente.

Abstract

This project emerges from the idea of creating a working group formed by students. Attending an international robotic contest is their objective. Designing an autonomous robot is the challenge. This project contains the studies, designs and implementations of the determination of the location using beacons and the internal communication.

The project covers the two main aspects. The internal communication is first developed using CAN BUS, but the selected option is serial communication. Beacons system provides both the robot position in the field and the opponent robot location.

Palabras clave: Robot, posicionamiento, balizas, CAN, UART.

Resumen extendido

Este proyecto surge a partir de la necesidad del diseño e implementación de un robot autónomo. Los temas que se desarrollan en este documento son el sistema de posicionamiento por balizas y la comunicación interna del robot. Se sigue una misma distribución para tratar los distintos temas, hablando de diseño, implementación hardware, implementación software, pruebas y error del sistema. Finalmente se discute el éxito en el sistema y las aportaciones de este trabajo a su autor.

El sistema de comunicación es imprescindible en el robot, cada parte del mismo debe enviar y recibir información. En este proyecto se desarrollan dos sistemas de comunicación válidos para implementar en un robot: BUS CAN, que es robusto y rápido; y UART (comunicación serie asíncrona), que es sencillo y funcional. Los dos sistemas están explicados y se ha logrado su implementación física, pero la opción utilizada finalmente en la competición es la de UART. La estrategia del robot, que no pertenece a este proyecto, es la encargada de decidir qué mensajes interesan en cada momento, cada placa envía mensajes periódicamente y recibe otros, enviados cuando la estrategia decide. El bus de datos es un sistema de comunicación que se usa principalmente en la industria, por lo que es ideal para un sistema con interferencias como es un robot, en el trabajo se explica el uso de las librerías RL_ARM de Keil y la implementación del sistema tanto en un microcontrolador como en otra tarjeta más potente. La comunicación por puerto serie, es más sensible a fallos, pero más sencilla de implementar.

En lo respectivo al sistema de balizado, cabe destacar que se divide en dos partes: posicionamiento del robot dentro del campo y posicionamiento del robot oponente. Para posicionar el robot dentro del campo se utilizan leds infrarrojos situados en balizas estáticas; se emiten pulsos modulados en distintas frecuencias, recibidos en el robot por un fototransistor y conectados a un pin de un microcontrolador para que la información sea procesada. El sensor gira constantemente de modo que cada vuelta detecta la posición de las tres balizas. Por medio de un algoritmo que calcula distancias a partir de los ángulos y utilización del método de trilateración se calculan las coordenadas del robot. Posicionar el robot del oponente es una tarea prioritaria en una competición de robótica, en caso de colisión, se expulsaría al causante de la misma. Para ello se utiliza un sensor situado en

el robot, que envía luz mientras gira y una baliza pasiva catadióptrica; cuando el haz de luz del sensor está apuntando a la baliza es reflejado, siendo así detectado el robot oponente. Conociendo el diámetro de la baliza y midiendo el ancho del pulso recibido se calcula la distancia de la misma. Conociendo la distancia y el ángulo de giro en el momento de la detección se determina la posición del oponente. Ambos sensores son movidos por un motor paso a paso. Todos los sistemas están conectados al microcontrolador.

El sistema completo incorporado al robot está formado por un microcontrolador; los sensores que giran con el motor y están unidos a este con piezas impresas en 3D; la fuente de alimentación de la placa, que son pilas alcalinas; y el controlador del motor. Los dos sistemas que necesitan 12V están conectados a la batería principal del robot. Las masas de todos los elementos están unidas.

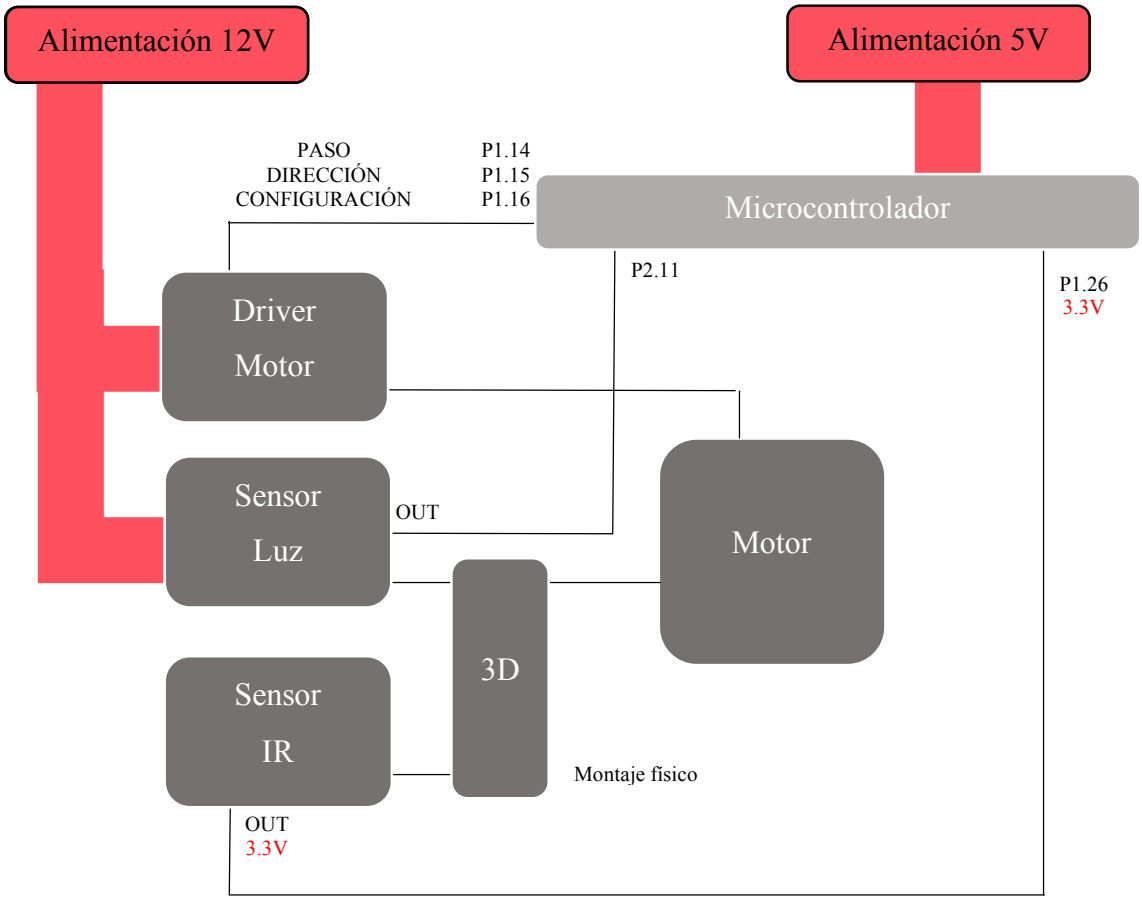


Figura 1: Esquema de conexionado del sistema.

En la figura 1, se muestra un diagrama de bloques que representa las partes que forman el sistema y las conexiones entre ellas. Las entradas y salidas señaladas en los diferentes bloques se corresponder con los elementos seleccionados para la realización del proyecto. Como sensor de las balizas activas se utiliza un sensor TSOP4838; para la detección del oponente se utiliza un sensor de luz difusa 42JS-PNP; como tarjeta para realizar el control de todo el sistema se utiliza una mini-dk2; el motor seleccionado es un NEMA-17 y su controlador un A4988. Se muestra el sistema completo con todos sus elementos en la figura 2. Las hojas de características de todos estos componentes se encuentran en los anexos.

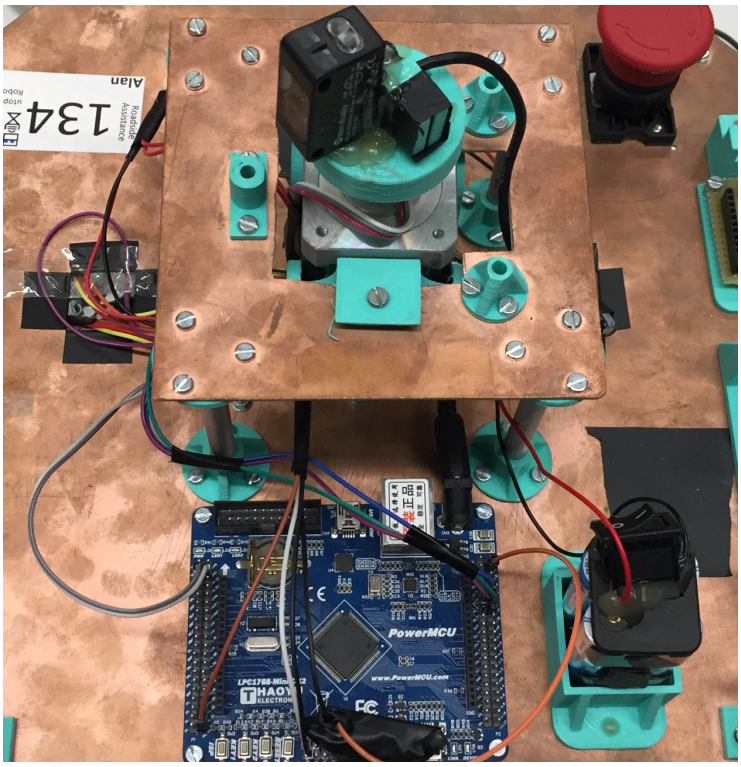


Figura 2: Montaje completo del sistema de posicionamiento.

Memoria

1 Introducción

1.1 Presentación

Este trabajo de fin de grado trata del desarrollo de dos sistemas integrados dentro de un robot autónomo. El sistema de posicionamiento basado en balizas, tanto de propio robot como del oponente; y la comunicación entre los sistemas que controlan las diferentes partes del robot.

1.2 Competición y objetivos

El desarrollo de un robot autónomo surge a partir de la participación en un concurso internacional de robótica: Roadside Assistance [6]. Es una competición compleja en la que el robot debe realizar una serie de tareas actuando de forma independiente. Durante el desarrollo del proyecto se trabaja en diferentes campos relacionados con la ingeniería, como mecánica, electrónica y programación.

Se puede diferenciar entre objetivos del equipo y objetivos del proyecto. Construir un robot desde cero supone un reto interesante para un grupo de estudiantes de ingeniería, por lo que el principal objetivo del equipo es aprender lo máximo posible en un entorno de grupo de trabajo. Respecto a la competición, el objetivo mínimo es superar la fase de homologación; siguiendo los criterios de seguridad, dimensiones y funcionamiento. El robot debe contar con un mecanismo de apagado de emergencia, no colisionar con un supuesto oponente y lograr al menos puntuar en alguna de las tareas. Por otro lado, los objetivos de este proyecto son:

- Proporcionar un sistema de posicionamiento que complemente los datos obtenidos por la odometría.

- Detectar la posición del robot oponente, garantizando en todo momento que no se produzca una colisión y ayudando a la estrategia en la toma de decisiones.

- Comunicar las placas que controlan los actuadores del robot.

Durante la realización del proyecto ha sido fundamental la coordinación con el resto de integrantes del equipo, de forma que los resultados finales de este trabajo están adaptados a los condicionantes de material, presupuesto y tiempo.

1.3 Tarjetas de desarrollo

A lo largo del documento se mencionan varias tarjetas de desarrollo que se han utilizado en el proyecto. Las especificaciones de cada una de ellas se encuentran en los anexos; aun así, se dedica este apartado para explicar brevemente cada una de ellas.

- Mini-dk2: Es una placa de evaluación desarrollada por “Haoyu Electronic" [3]. Está basada en un procesador LPC1768. Se utiliza en este proyecto ya que tiene un número suficiente de pines y todos los periféricos necesarios. Además, el autor del proyecto está familiarizado con el entorno de programación y el uso de la placa.

- Raspberry pi: Es una tarjeta computadora de bajo coste. Cumple con el requerimiento por parte del equipo del uso de un sistema operativo Linux. También existen gran cantidad de ejemplos desarrollados para esta placa, lo que supone una ayuda a la hora de desarrollar el robot.

- MCB2300: Es una placa de evaluación similar a la mini-dk2, con la particularidad de que tiene la interfaz física de CAN. Se utiliza para realizar las primeras pruebas de este apartado, ya que el procesador es también el LPC1768.

1.4 Distribución del libro y metodología de trabajo

La distribución del libro se realiza diferenciando claramente los dos temas que se trabajan en el proyecto. De esta manera, quedan separados todos los aspectos de metodología de trabajo, diseño, pruebas y resultados. Aun así, se tratan los mismos puntos en todos los temas: se explica el diseño teórico, la implementación hardware y software, los resultados y la evaluación de errores. Se dividen en cuatro, en lo referente al balizado se hace una diferenciación entre el posicionamiento del robot y la detección del oponente; en la comunicación se tratan por separado los dos sistemas trabajados, comunicación por puerto serie utilizando UART e implementación de un BUS de datos utilizando el protocolo CAN.

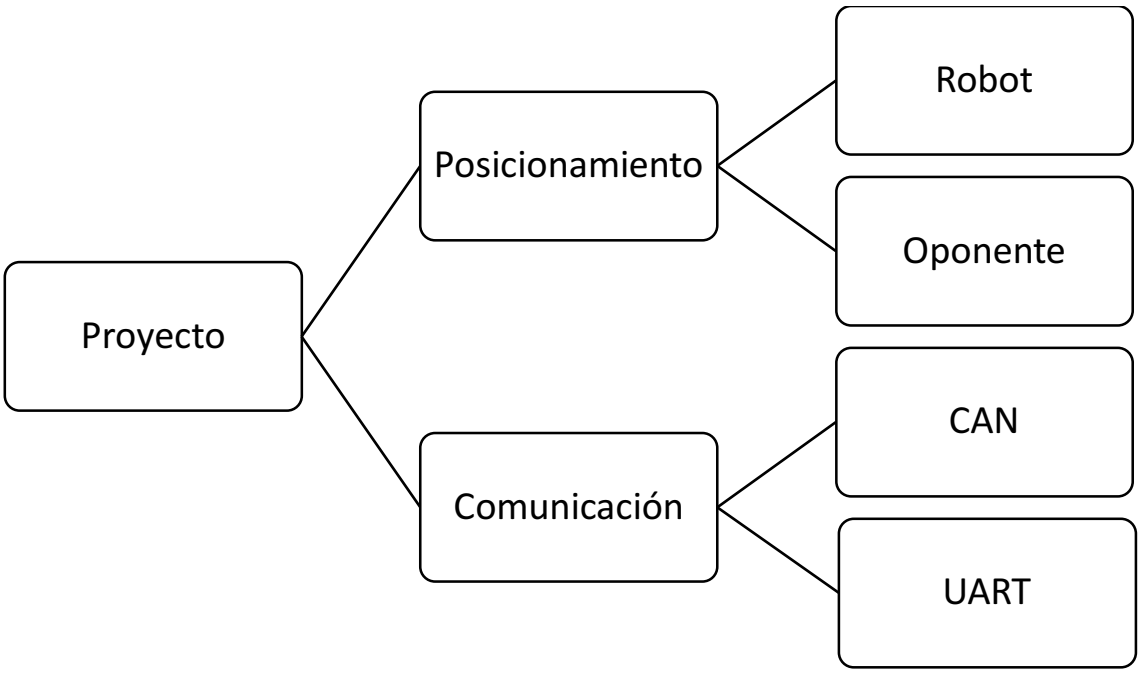


Figura 3: Distribución del proyecto.

Durante el desarrollo del proyecto se llevan a cabo las dos partes, dando prioridad a una o a otra dependiendo de las necesidades del equipo. En ambos casos, se realiza una fase previa de estudio del problema y posibles soluciones; también se contrastan diferentes fuentes de información reflejadas en la bibliografía.

A continuación, se explican brevemente cada una de las fases por las que ha pasado el proyecto durante su realización.

Sistema de balizado.

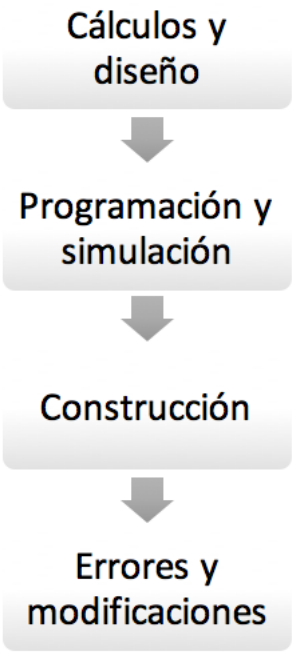
Soluciones matemáticas al problema. Diseño del circuito a utilizar. Posibles alternativas.

Simulaciones del algoritmo e implementación del código. Optimización del mismo.

Construcción de las balizas. Selección del motor y montaje de sensores en este.

Se realizan pruebas y modificaciones constantes hasta obtener un resultado válido.

Sistema de comunicación.



Estudio de distintas posibilidades. Estudio de las librerías del sistema operativo.

Desarrollo y simulación del código. Pruebas con otras placas como la MCB2300.

Comunicación entre mini-dk2 y raspberry pi. Pruebas conjuntas con el equipo.

Comparación y selección final. Por necesidades técnicas y temporales.

2 Comunicación interna del robot

2.1 Introducción

El robot consta de diferentes actuadores y sensores, cada uno de ellos debe ser controlado, para lo que se utilizan cuatro tarjetas de control. La comunicación entre éstas es necesaria para obtener un funcionamiento adecuado, en este apartado se detalla el sistema de comunicación, su evolución y el diseño final. Durante la realización del proyecto se desarrollan dos sistemas de comunicación: BUS CAN y UART; se explican los dos y ambos se han llegado a implementar, aunque el único utilizado en el robot es el de la UART.

2.2 Alternativa 1: Comunicación por BUS CAN

El protocolo CAN (Control Area Network), inicialmente, fue diseñado para la industria automovilística, pero actualmente se utiliza en multitud de sectores de la industria. Se caracteriza por la rápida transferencia de datos, su eficaz detección de errores y por alta inmunidad ante interferencias electromagnéticas [7]. Estas características son las deseadas en el diseño de cualquier robot, ya que suele estar sometido a interferencias, tanto internas como externas.

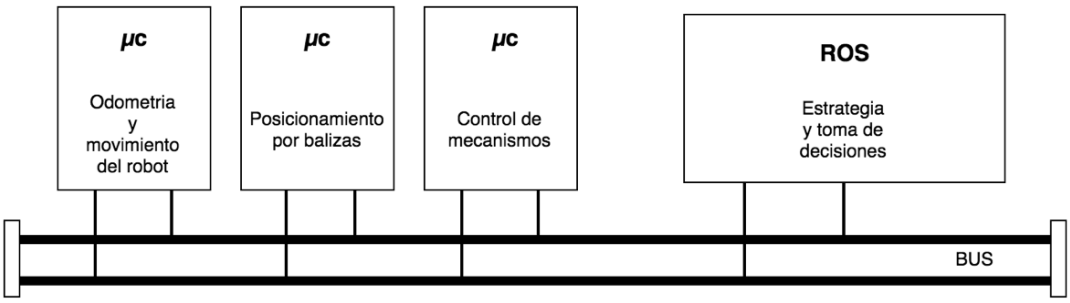


Figura 4: Esquemático de las placas comunicadas por un BUS de datos.

2.2.1 Diseño teórico

El sistema está compuesto por varios nodos, todos ellos comunicados entre sí a través de un BUS de datos. Consta de cuatro nodos, tres de ellos son microcontroladores y el cuarto una raspberry pi. Cada microcontrolador se encarga de una tarea física (movimiento, actuadores y posicionamiento), de tal forma que están continuamente enviando y recibiendo información del BUS. La raspberry pi se encarga de la estrategia, es decir, recibe esta información, procesándola y coordinando las funciones que realizan el resto partes.

Cada nodo consta de su controlador CAN, éste es capaz de enviar y recibir datos al mismo tiempo, lo que es la base para la arbitración del BUS. Para la trasmisión de datos son necesarios al menos dos cables, el BUS se puede encontrar en dos estados: cuando ambos cables están a una tensión entre 0 y Vcc el bus está en espera y se reconoce como un nivel lógico uno, estado no dominante; por otro lado, cuando la diferencia de voltaje es máxima el BUS se encuentra en estado dominante y se representa con un nivel lógico cero [10] .

La comunicación se lleva a cabo por medio de mensajes, que tienen una estructura definida. Estos pueden ser enviados al BUS o requeridos por algún nodo, pero en este sistema no se utiliza el requerimiento de mensajes. La estructura del mensaje es la siguiente:

Figura 5: Estructura de un mensaje de CAN-BUS [10].

El controlador CAN genera el mensaje, mientras que la aplicación software se encarga de los datos que forman este mensaje. Comienza con un bit dominante para

indicar el inicio del mensaje. Seguidamente, el identificador del mensaje, que puede ser hasta de 29 bits, se utiliza para definir que nodos recibirán este mensaje, ya que el mensaje es enviado por un nodo, pero puede ser leído por varios de ellos. El bit RTR se pone a cero, ya que solo se utiliza cuando se requiere un mensaje remoto. El DLC marca la longitud del mensaje que se va a enviar. A continuación, se envía la información, que ocupa de cero a ocho bytes. El mensaje acaba con un sistema de detección y corrección de errores, formado por un sistema de chequeo cíclico de 15 bit y un bit (ACK) que se utiliza para comprobar que el mensaje ha sido recibido por alguno de los nodos. Cuando el mensaje ha concluido se envía el bit de final de mensaje. Una vez que el mensaje llega al BUS de datos, si este está en espera, el mensaje será enviado a los distintos nodos, si por el contrario está en modo dominante, el mensaje esperará hasta que haya de nuevo un uno lógico. Si varios mensajes llegan se llevará a cabo la arbitración del bus, de manera que aquellos mensajes que tengan mayor prioridad serán enviados primero [10].

Para implementar este sistema se cuenta con las librerías de RL-ARM proporcionadas por KEIL. Gracias al uso de estas librerías la implementación del BUS es más sencilla y el funcionamiento del sistema ya ha sido comprobado anteriormente. Por otro lado, es necesario utilizar el sistema operativo RTX, lo que añade cierta dificultad al código de la aplicación.

2.2.2 Implementación Software

El sistema operativo (OS) trabaja con tareas. Una de las tareas se utiliza para enviar mensajes, otra de ellas para recibir información y la última constituye el hilo principal del programa, donde se incluye el código del balizado. La utilización de un OS en una aplicación de tiempo real, que necesita realizar distintas operaciones en espacios de tiempo muy cortos, puede suponer un problema. El OS tiene limitaciones temporales debido a los continuos cambios de una tarea a otra, por lo tanto, para que el programa funcione de acuerdo con los requerimientos, es necesario optimizarlo al máximo. Este es uno de los motivos por los que se desecha la utilización del BUS CAN, ya que es más sencillo y rápido el otro método. La utilización de las librerías [3] simplifica mucho el código, en el caso de esta aplicación se utilizan las funciones necesarias para una comunicación básica.

La configuración inicial se realiza utilizando tres funciones, a continuación, se explica cómo se configura el microcontrolador de la parte de balizado, los otros dos se configuran de la misma forma, variando el identificador. Primero se define la estructura del mensaje que se desea enviar. Se selecciona un identificador, seguido de ocho bytes de datos definidos por el usuario. La longitud del mensaje en este caso es de un byte y se utiliza el canal 1 para enviarlo. Por último, se define el formato estándar y el tipo de dato.

```
/* Initialize message  = { ID, {data[0] .. data[7]}, LEN, CHANNEL, FORMAT, TYPE } */
CAN_msg msg_send       = { 10, {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00},
                             1, 2, STANDARD_FORMAT, DATA_FRAME };
```

Figura 6: Código para la creación de un mensaje.

A continuación, se habilita el controlador. Para ello se selecciona uno de los controladores y la velocidad de trasmisión. Como se observa en la figura 7, se utiliza el controlador 2 y se selecciona una velocidad de 500 kbit/s.

```
CAN_init (2, 500000);                /* CAN controller 2 init, 500 kbit/s   */
```

Figura 7: Código para la inicialización del BUS.

Por último, es necesario definir el tipo de mensaje que se desea recibir. Se habilita la recepción de datos, el canal que se va a utilizar (canal 2) y el identificador (33 estándar). Solo se recibirán mensajes con este identificador.

```
CAN_rx_object (2, 2,  33, DATA_TYPE | STANDARD_TYPE); /* Enable reception    */
                                    /* of message on controller 2, channel 1 */
                                    /* standard id 33                        */
```

Figura 8: Código que define el tipo de mensaje a recibir.

Una vez que todo está configurado se inicializa el controlador 2. Se utiliza la función de la figura 9, donde únicamente hay que seleccionar el número de controlador a utilizar.

```
CAN_start (2);                       /* Start controller 2                    */
```

Figura 9:Código para la inicialización del controlador CAN.

Cada una de las partes del robot envía y recibe datos en función de sus necesidades. La parte de movimiento envía posición y velocidad, recibiendo consignas de aceleración. La parte del balizado envía posición global y la posición respecto al oponente. La parte de actuadores recibe la orden de comenzar y devuelve una señal cuando el trabajo ha concluido. En todos los casos el mensaje se envía y recibe de la misma forma, siendo cada una de las placas responsable de enviar su mensaje una vez que internamente ha calculado los parámetros pertinentes. La estrategia se encarga de coordinar todos estos mensajes.

En el caso de la mini-dk2, para enviar un mensaje se utiliza la función de la figura 9, es necesario definir el controlador utilizado.

```
CAN_send (2, &msg_send, 0x0F00);  /* Send msg_send on controller 2        */
```

Figura 10: Código para el envío de un mensaje.

Para recibir el mensaje se comprueba si algo ha llegado por el canal 2 con el identificador anteriormente definido. Se almacenan los datos recibidos en una variable global para que estos puedan ser utilizados con mayor comodidad.

```
/* When message arrives store received value to global variable val    */
if (CAN_receive (2, &msg_rece, 0x00FF) == CAN_OK)  {
  {

  val = msg_rece.data[3];
  val |= msg_rece.data[2]<<8;
  val |= msg_rece.data[1]<<16;
  val |= msg_rece.data[0]<<24;

  }
```

Figura 11: Código para la recepción de un mensaje.

En el caso de la raspberry pi, es necesario añadir una interfaz CAN. En lo referido al software, se debe habilitar la interfaz SPI e instalar las librerías de CAN. La implementación de este sistema en Linux no forma parte de este proyecto [2].

2.2.3 Implementación Hardware

Como ya se ha mencionado anteriormente, el BUS de datos consta de al menos dos cables. En este caso se utilizan tres, dos de información y el tercero para que las masas de todas las placas estén unidas. En muchos casos, en aplicaciones más críticas o dónde el cable es de mayor longitud, se emplean conectores de nueve pines, cuyo resultado es un sistema sea más fiable.

Para que el sistema funcione es imprescindible colocar dos resistencias de 120 ohm, una a cada extremo del BUS.

El microprocesador LPC1768 cuenta con la interfaz CAN, pero en la mini-dk2 es necesario utilizar un transceptor para generar las señales CAN_H y CAN_L. Se utiliza el MCP2551. Éste se alimenta a 5V directamente desde un pin de la placa. En la figura 12, se muestra un esquema del integrado utilizado. Las patillas CAN_Tx y CAN_Rx se llevan a los pines 2.7 y 2.8 de la placa. Las patillas CAN_H y CAN_L se conectan directamente al BUS.

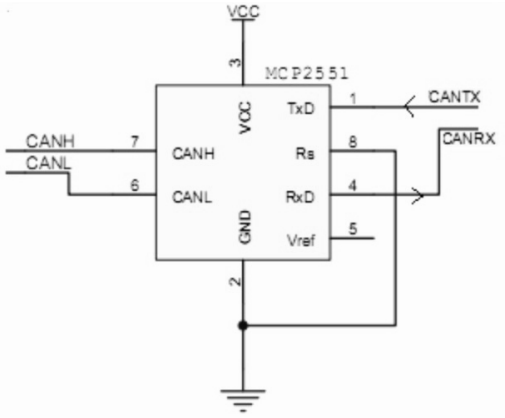


Figura 12: Esquemático de un MCP2551 [anexo 2].

Por otro lado, la raspberry pi no cuenta con interfaz CAN, por lo que se utiliza una placa que consta de un integrado MCP2515, que transforma de SPI a CAN, de un transceptor TJA1050 y de un reloj. El esquemático de la placa y el conexionado con los pines de la raspberry pi se muestra en la figura 13. Los pines de la raspberry pi no deben recibir tensiones superiores a los 3.3V, por lo que es necesario realizar una modificación del circuito. Se separan las alimentaciones de ambos integrados, dejando la del MCP2515 a 3.3V y la del TJA1050 a 5V, que es la tensión máxima del BUS.

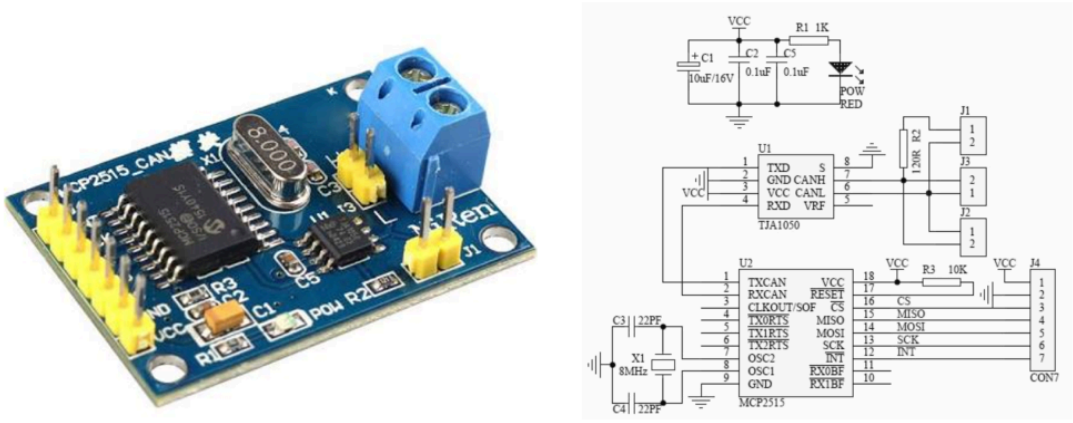


Figura 13: Esquema de conexionado y fotografía de la placa spi-can [11].

De esta forma el sistema de comunicación a través de BUS CAN estaría formado por: tres mini-dk2 y sus respectivos transceptores; la raspberry pi y la tarjeta SPI-CAN; y los cables con el par de resistencias de 120Ω.

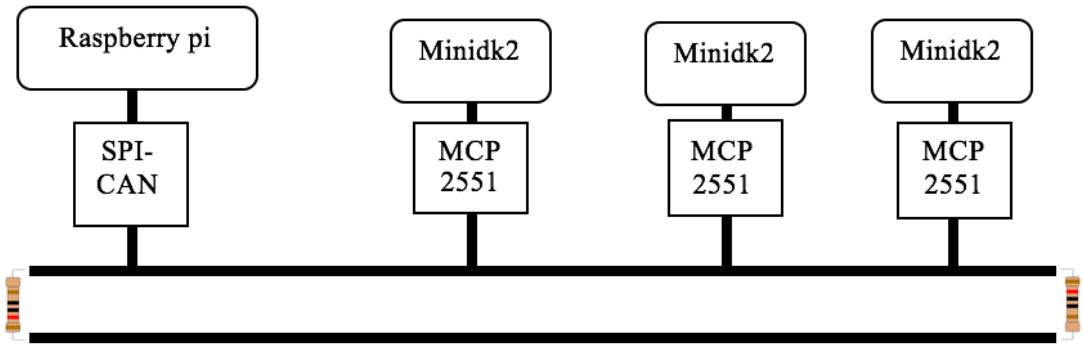


Figura 14: Esquema del sistema de comunicación por BUS CAN.

Para comprobar el funcionamiento del BUS y la correcta trasmisión de datos se realiza un montaje provisional que comunica una de las mini-dk2 con la raspberry pi. Este montaje se muestra en la figura 15.

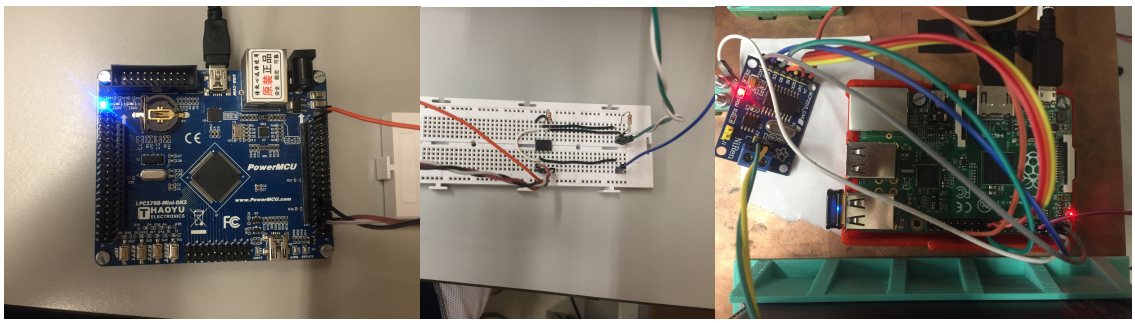


Figura 15: Imágenes del BUS implementado en una placa protoboard.

2.3 Alternativa 2: Comunicación por UART

La comunicación por medio de UART (Universal Asynchronous Receiver-Transmitter) se realiza de forma asíncrona, enviando datos en serie. El controlador se encarga de transformar la información proporcionada por un microcontrolador o un PC y enviarla de forma secuencial. Otro controlador recibe estos bits y los agrupa para obtener el mensaje. Es un sistema de comunicación simple y fácil de implementar en la mayoría de placas, ya que es usado en múltiples aplicaciones.

2.3.1 Diseño teórico

La implementación de la comunicación a través del puerto serie en este proyecto ha sido consecuencia de la participación en el concurso. Como ya se ha mencionado anteriormente, la idea inicial era usar un BUS de datos, al no haberse realizado suficientes pruebas, se ha optado por esta alternativa cuyo funcionamiento ya había sido testado.

El sistema consta de tres microcontroladores y una raspberry pi, todos ellos tienen tanto la interfaz software como hardware para implementar un sistema de comunicación a través del puerto serie. Cada uno de los microcontroladores enviará los datos a la raspberry pi cuando se hayan realizado las operaciones internas en su código y será tarea de la estrategia tener en cuenta estos datos dependiendo de la situación durante la competición.

2.3.2 Implementación Hardware

La implementación hardware no requiere ningún diseño, todas las mini-dk2 cuentan con un conversor CP2102, que trasforma los datos enviados en serie a USB. Se utiliza la salida mini-USB de este módulo para realizar la comunicación; la raspberry pi tiene cuatro entradas USB, por lo que simplemente se conecta cada una de las placas a la raspberry pi utilizando un cable USB-miniUSB estándar.

2.3.3 Implementación Software

La implementación software se realiza utilizando las librerías proporcionadas por Keil para UART. Lo relevante para la aplicación es tener una forma de comunicación sencilla y funcional, por lo que el código no es demasiado complejo.

Hay dos consideraciones importantes, el puerto que se desea utilizar y la información a enviar. En este caso se usa la UART0 y la información depende de cada placa, aunque en ningún caso supera el peso de 10 bytes.

En la función de la figura 16 se especifica el uso de UART0 y la velocidad de trasmisión. La velocidad de envío está condicionada por la aplicación más crítica, que, en el caso del robot, es la tarjeta que controla la odometría. Ésta envía y recibe cada diez milisegundos.

$$Velocidad = 10bytes/10ms \quad (1)$$

$$Velocidad = 8000\ bits/s \quad (2)$$

$$Velocidad = 8000\ baudios \quad (3)$$

Finalmente se selecciona la velocidad de 19200 baudios, ya que es una velocidad estándar y no compromete la funcionalidad del código.

```
uart0_init(19200);
```

Figura 16: Código para el inicio de la UART 0.

Para el envío de datos es necesario, primero convertir los datos a cadena y luego enviarlos utilizando la función de la figura 17. Una se produce la llamada a la función, se espera a que esta devuelva la variable "tx_completa" que indica que los datos han sido almacenados en el buffer correctamente.

```
sprintf(cadena,"%d"",""%d"",""%d""\n",xr,yr,D);
tx_cadena_UART0(cadena);
while(tx_completa==0);
tx_completa=0;
```

Figura 17: Código para el envío de datos por el puerto serie.

Para la recepción de datos se utiliza el código de la figura 18, en el hilo principal del programa se comprueba constantemente si ha llegado información al buffer, en caso de que esto ocurra, se guarda la información en la variable global "data".

```
if(rx_completa){
  rx_completa=0;
  data=buffer[6];
}
```

Figura 18: Código para la recepción de datos por el puerto serie.

2.3.4 Pruebas y resultados

Se han realizado pruebas de trasmisión de información desde la mini-dk2 hasta el ordenador a través del puerto serie. En el ordenador, se recibe la posición del robot cada segundo. En la figura 19, se muestran extractos de la ventana de la aplicación "Putty", por la que se muestran los datos recibidos.

Figura 19: Datos recibido a través del puerto serie por el ordenador.

El sistema implementado en el robot funciona exactamente igual, en vez de llevar el USB al puerto del ordenador, se conecta a la raspberry pi. Se realizan cambios en el código al recibir información de la estrategia, ya que esa información se almacena en un buffer, no siempre en la misma posición. Se han realizado pruebas con el montaje completo en el robot y el sistema funciona correctamente.

3 Sistema de posicionamiento por balizas

3.1 Posicionamiento absoluto

El sistema de posicionamiento del robot se basa en un sistema de odometría, que proporciona una información, esta información tiene un error acumulativo. Este error debe ser corregido por la información proporcionada por el sistema de balizado. Este sistema utiliza tres balizas activas que envían señales a distintas frecuencias. El receptor es girado por un motor paso a paso. Cada vez que el motor completa una vuelta se calcula la posición del robot.

3.1.1 Diseño teórico

El sistema empleado para el posicionamiento ha sufrido varios cambios a lo largo del proceso de diseño, en este apartado se detalla tanto aspectos teóricos como la evolución y el resultado final del proceso.

Este sistema de posicionamiento está diseñado siguiendo las especificaciones del concurso. El campo de juego es un tablero rectangular de dos metros de ancho y tres metros de largo. Las balizas se sitúan en las plataformas exteriores, bien rojas o azules, que se pueden ver en la figura 18. Conocer la distancia entre las balizas es fundamental para que el sistema sea fiable, en caso de variar estas distancias, se tendrían que variar ciertos valores asignados a variables del código.

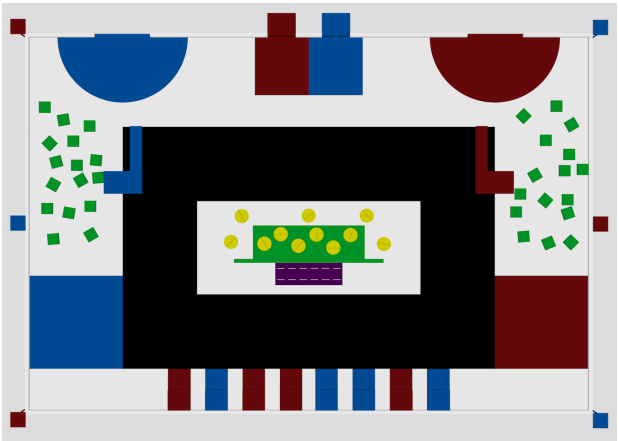


Figura 20: Campo de juego.

El cálculo de las coordenadas de robot dentro del campo se realiza por medio de trilateración. Para esto es necesario conocer la distancia de cada una de las balizas al robot. Estas distancias se determinan a partir los ángulos que se forman entre las líneas imaginarias que unen al robot con cada una de las balizas. De esta forma se divide en dos pasos la obtención de la posición del robot: distancia a cada una de las balizas y cálculo de coordenadas.

Para obtener la distancia de las balizas al robot se utiliza el teorema del coseno, de tal forma que se obtiene un sistema de tres ecuaciones con tres incógnitas. Se forman tres triángulos utilizando las rectas entre balizas y las rectas entre robot y balizas, se pueden ver en la figura 19. De esta forma, en cada triángulo se aplica la ecuación del teorema conociendo las distancias entre balizas y los ángulos alpha, betha y theta.

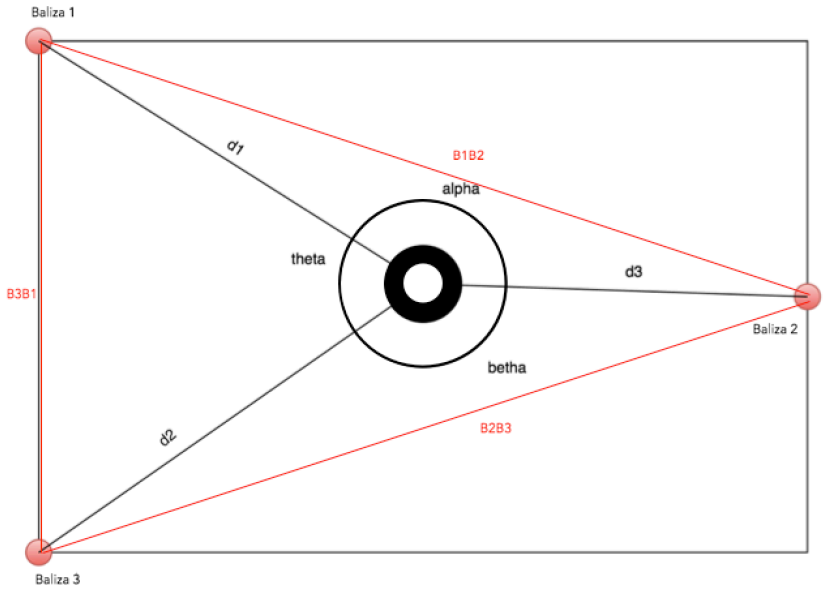


Figura 21: Esquemático del robot en el campo. Distancias y ángulos a tener en cuenta.

$$\overline{B1B3} = 2\ metros \tag{4}$$

$$\overline{B2B3} = \sqrt{10}\ metros \tag{5}$$

$$\overline{B1B2} = \sqrt{10}\ metros \tag{6}$$

$$\overline{B1B2}^2 = d1^2 + d3^2 - 2 * d1 * d3 * \cos(alpha) \tag{7}$$

$$\overline{B2B3}^2 = d2^2 + d3^2 - 2 * d2 * d3 * \cos(betha) \tag{8}$$

$$\overline{B1B3}^2 = d1^2 + d2^2 - 2 * d1 * d2 * \cos(theta) \tag{9}$$

A la hora de resolver este sistema surgen problemas, experimentalmente se determina que no tiene solución analítica. Se emplean métodos numéricos, de forma que se incrementa el valor de una de las distancias desde su valor mínimo hasta el máximo (0.1m - 3.1m). Para una distancia d3 determinada, el sistema converge y las otras dos distancias toman el valor adecuado. Al igual que cualquier sistema resuelto con métodos numéricos, la solución no es exacta, pero sí lo suficientemente precisa para la aplicación. La precisión de esta solución se puede ajustar, teniendo en cuenta que cuanto más precisa sea, más tiempo de cálculo requiere. Se necesita una precisión que no afecte de ninguna manera al cálculo de la posición y no requiera un tiempo de cálculo mayor a 5ms; la posición del robot debe ser enviada a la estrategia lo antes posible, si se supera el tiempo establecido, esta información dejaría de ser útil. El algoritmo primeramente es desarrollado en Matlab (figura 22) y más tarde adaptado a c.

```
for d3=0:0.0001:3.1

if cos(alpha)>0
d1=d3*cos(alpha) - (d3^2*cos(alpha)^2 - d3^2 + 10)^(1/2);
if d1<0
    d1=d3*cos(alpha) + (d3^2*cos(alpha)^2 - d3^2 + 10)^(1/2);
end
else
d1=d3*cos(alpha) + (d3^2*cos(alpha)^2 - d3^2 + 10)^(1/2);
if d1<0
    d1=d3*cos(alpha) - (d3^2*cos(alpha)^2 - d3^2 + 10)^(1/2);
end
end

if cos(betha)>0
d2=d3*cos(betha) - (d3^2*cos(betha)^2 - d3^2 + 10)^(1/2);
if d2<0
   d2=d3*cos(betha) + (d3^2*cos(betha)^2 - d3^2 + 10)^(1/2);
end
else
d2=d3*cos(betha) + (d3^2*cos(betha)^2 - d3^2 + 10)^(1/2);
if d2<0
   d2=d3*cos(betha) - (d3^2*cos(betha)^2 - d3^2 + 10)^(1/2);
end
end

theta_big=acos((4-d1^2-d2^2)/(-2*abs(d1)*abs(d2)))+0.0001;
theta_small=acos((4-d1^2-d2^2)/(-2*abs(d1)*abs(d2)));

if ((theta<theta_big)&&(theta>theta_small))
    break;
end
end
```

Figura 22: Código en Matlab para resolver el sistema de ecuaciones.

Este algoritmo se prueba para diferentes escenarios. Se introducen ángulos correspondientes a puntos conocidos en el campo y se comprueba que funciona correctamente. Se tiene un error inferior al centímetro para posiciones cercanas al centro del campo y éste va aumentando según se acerca a zonas fuera de los triángulos de la figura 21.

Matlab tiene una alta capacidad computacional, por lo que realizar estos cálculos supone un tiempo despreciable, mientras que el microprocesador utilizado no es capaz de realizarlos en un tiempo suficientemente pequeño. Por ello, una vez cargado el programa en la mini-dk2, es necesario reducir el número de iteraciones del bucle, así como mejorar el algoritmo de búsqueda. Éste, en vez de avanzar de uno a uno, va acotando el rango de valores en el que se encuentra el numero hasta alcanzar el valor deseado. El código empleado en el programa se encuentra al final del documento

El cálculo de la posición en el campo de juego una vez que se tienen las tres distancias ser realiza por trilateración. Es decir, el punto que se busca será la intersección de las tres esferas formadas a partir de cada baliza como centro y cada distancia correspondiente como radio.

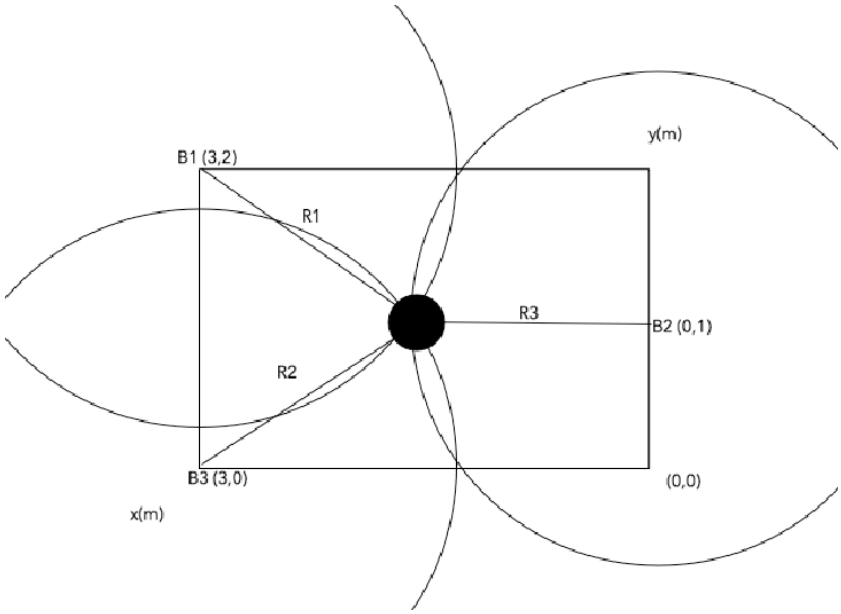


Figura 23: Esquemático del robot en el campo. Explicación de trilateración.

Tomando como origen de coordenadas la esquina inferior derecha del campo tenemos las coordenadas de las balizas.

$$B1\ (x_1, y_1) = (3,2) \quad (10)$$

$$B2\ (x_2, y_2) = (0,1) \quad (11)$$

$$B3\ (x_3, y_3) = (3,0) \quad (12)$$

A partir de la ecuación de la circunferencia, se definen las tres circunferencias, teniendo en cuenta los valores de las distancias obtenidas (d1, d2, d3), que pasan a ser radios (R1, R2, R3).

$$(x-3)^2+(y-2)^2=R1^2 \quad (13)$$

$$(x-0)^2+(y-1)^2=R2^2 \quad (14)$$

$$(x-3)^2+(y-0)^2=R3^2 \quad (15)$$

Operando el sistema se obtienen las ecuaciones implementadas en el código.

$$y=\frac{4+R3^2-R1^2}{4} \quad (16)$$

$$x=\sqrt{R2^2-(y-1)^2} \quad (17)$$

En lo referente al sistema de envío de información, cabe discutir dos cuestiones principales, el medio que se va a utilizar para enviarla y cómo se va a codificar. En todo momento se ha decidido el uso de balizas activas y un receptor pasivo. Esta decisión se toma partiendo de la idea de crear una señal diferente en cada baliza, además de que la señal debe recorrer la mitad de distancia que una señal que rebota. Para diferenciar las balizas entre si se ha optado por enviar señales a diferentes frecuencias. A partir de estas frecuencias y conociendo la posición del motor es posible calcular los ángulos que se forman entre las balizas. Con estos datos se pueden aplicar los métodos matemáticos explicados anteriormente.

La idea inicial fue utilizar un fototransistor para recibir señales de frecuencias entre 100Hz y 1000Hz, estas señales serían generadas por un integrado 555 en cada una de las balizas. Esta idea se desechó debido a la dificultad de utilizar un filtro a la salida del fototransistor que acondicionase la señal para ser procesada por el microcontrolador. En lugar del fototransistor como tal se ha utilizado un receptor que lleva integrado el sistema de filtrado y amplificación. Se trata del TSOP4838, un receptor de señales infrarrojas que solo detecta señales de 38kHz. Para poder utilizar este receptor las balizas deben emitir a esta frecuencia, por lo que se han utilizados dos integrados 555, uno de

ellos, común para las tres balizas, que genera la frecuencia de 38kHz y otro que genera las frecuencias que debe diferenciar posteriormente el microcontrolador.

Es necesario procesar la información que recibe el sensor, para ello, se conecta la salida de éste a una entrada del micro, que está continuamente capturando la señal. Cada vez que el motor se mueve un paso, esta información se almacena en un array junto con el paso en el que se encuentra el motor, de manera que cuando gire una vuelta se tiene asociado a cada paso la información que obtuvo el sensor en ese momento. Una vez que se realiza una vuelta, con la información almacenada se calculan los tres ángulos formados entre balizas y seguidamente la posición del robot.

3.1.2 Implementación Hardware

La parte hardware de este apartado se divide en las balizas emisoras y el sistema de recepción. Se han diseñado y construido tres balizas iguales. Utilizando potenciómetros se consiguen las diferentes frecuencias deseadas. Por otro lado, el sistema de recepción consta de un motor paso a paso, con su controlador, y un receptor de infrarrojos.

Balizas emisoras

Las balizas emisoras tienen se alimentan a 8V, tensión generada por dos pilas de litio. La tensión generada por éstas se reduce con el paso del tiempo, por lo que el circuito que genera las señales está alimentado a 6V. Para ello se utiliza un regulador de tensión, el seleccionado en este caso es el L7806. Esta medida evita problemas por regulación de línea que fueron detectados durante la fase experimental.

El circuito se puede dividir en tres partes: La primera está formada por un circuito integrado 555 y componentes pasivos, generan la señal de información. Se utiliza el modo de operación multivibrador astable, de forma que son necesarias dos resistencias y un condensador. En el diseño se utilizan potenciómetros en vez de resistencias de valor fijo. A la hora de hacer pruebas es mejor poder variar las frecuencias sin tener que realizar cambios en la placa. Con este modo de funcionamiento se obtiene una señal cuadrada de

periodo constante, con la particularidad de que el tiempo en alto y el tiempo en bajo de la señal no coincide. El esquema es tal y como se muestra en la figura 24, se utiliza este modo de funcionamiento ya que no requiere una señal de disparo.

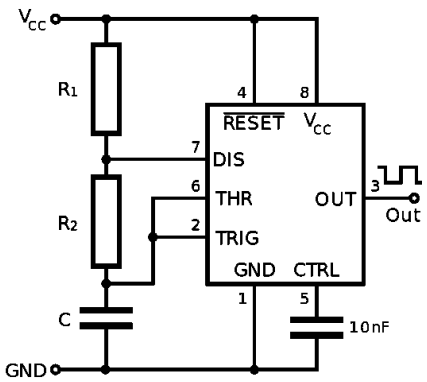


Figura 24: Circuito integrado 555 en modo de funcionamiento multivibrador astable [5].

La señal generada a la salida del primer integrado (figura 25) se conecta con el terminal "RESET" del segundo integrado, éste tiene la misma configuración que el anterior.

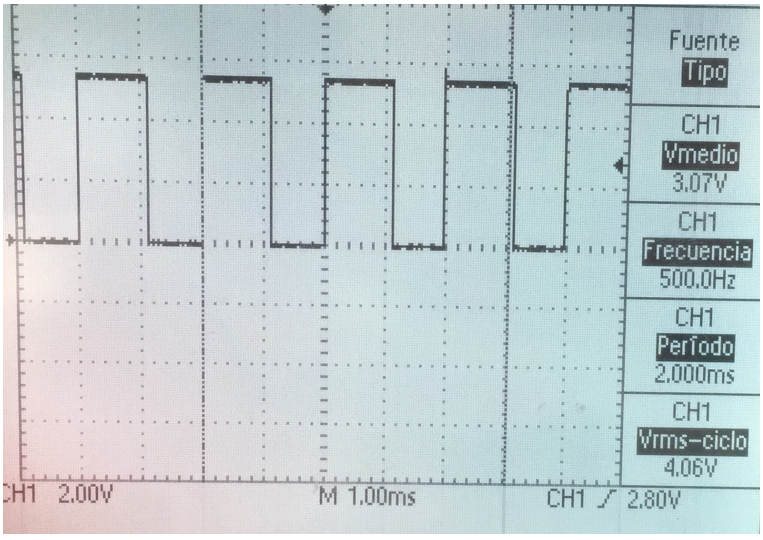


Figura 25: Señal generada a la salida del primer 555. OUT 1.

Se utilizan elementos pasivos de otros valores de forma que se genera la señal de 38kHz. Se determina el uso de condensadores de 100uF para obtener las frecuencias deseadas con resistencias y potenciómetros fáciles de encontrar en el mercado. Los valores ideales de las resistencias para cada uno de los montajes se determinan a partir de las ecuaciones [18], [19] y [20].

$$f = \frac{1}{\ln(2) * C * (R1 + 2R2)} \quad (18)$$

$$t_{alto} = \ln(2) * (R1 + R2) * C \quad (19)$$

$$t_{bajo} = \ln(2) * R2 * C \quad (20)$$

La frecuencia producida por el primer 555 se comprende entre 300Hz y 500Hz; se desea que la señal esté aproximadamente un tiempo similar a nivel alto y a nivel bajo, para este requerimiento se debe cumplir que $R2 \gg R1$ en las ecuaciones [15], [16] y [17]. Así, obtenemos que los valores de R2=[14-24k]kΩ, manteniendo fijo R1=100kΩ. Se utiliza un potenciómetro de 25kΩ. En el caso del segundo integrado se desea una señal de 38kHz, siendo este el único requerimiento y usando condensadores de 100uF en todos los casos, se determinan los valores de R1=100Ω y R2=140Ω.

A la salida de este integrado tenemos una señal como la de la figura 24. Se trata de la señal de 38kHz, en pulsos de 500Hz.

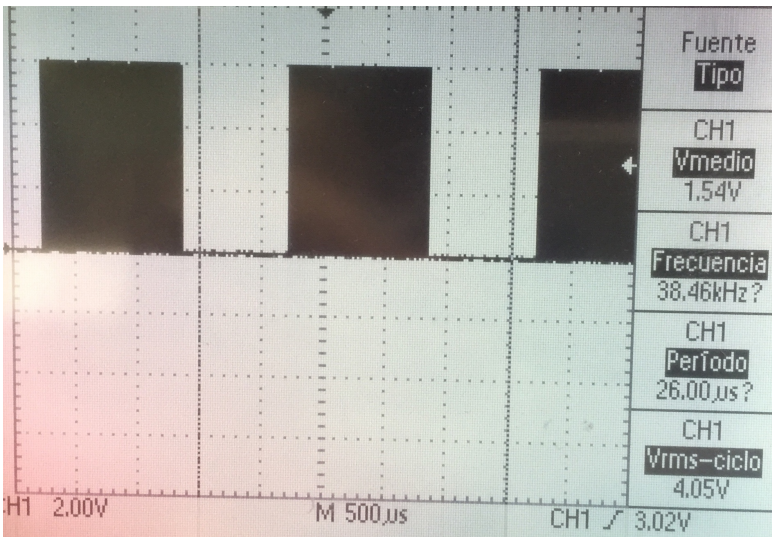


Figura 26: Señal generada a la salida del segundo 555. OUT 2.

Se centra la imagen en los pulsos de nivel alto y se ajustan los segundos por división en el osciloscopio, obtenemos la señal de la figura 27.

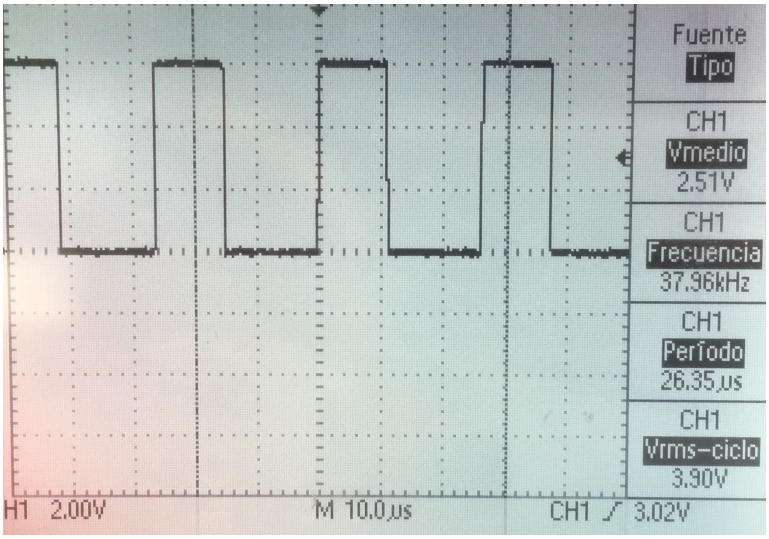


Figura 27: Señal de la Figura 26 ampliada.

Esta señal alimenta un transistor de potencia IRF530N que controla los leds. Éstos conmutan muy rápidamente, por lo que, reciben picos de corrientes de forma intermitente, así se evita que puedan llegar a quemarse. El modelo de led elegido es el TSAL6200, ya que es el más adecuado para trabajar con el receptor según las especificaciones definidas en las hojas de características del anexo 3. Como se observa en la figura 28, tienen un ángulo de emisión próximo a los 20º para distancias cercanas.

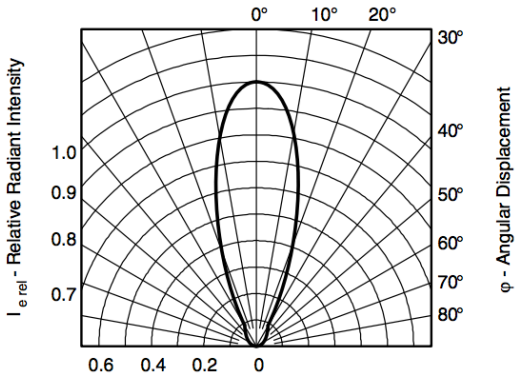


Figura 28: Ángulo de emisión de los leds en función de la intensidad (anexo 4).

Para cubrir los 90º de las balizas basta con utilizar 5 leds, pero en el caso de una de las balizas es necesario abarcar 160º, para cumplir con este requisito, se utilizan ocho diodos en cada baliza. Se colocan cuatro pares de leds en serie, de forma que la corriente se distribuye en dos ramas, pero la tensión total se divide entre cuatro, alimentando cada led a 1.5V.

En las hojas de características (anexo 3) se observa que con una intensidad de 200mA se alcanzan distancias típicas de 45 metros; se decide utilizar este parámetro de corriente ya que es un estado de funcionamiento recomendado por el fabricante. No se utilizan resistencias para limitar la corriente que pasa por los leds ya que con la resistencia del transistor y la fuente es suficiente.

$$Ron = 0.064\Omega; Rcc \approx 2\Omega \tag{21}$$

$$Vcc = 6V; Vled = 1.4V \tag{22}$$

$$I = \frac{6 - 1.4 * 4}{2 + 0.064} = 0.19\ A \tag{23}$$

El esquema completo de la baliza se muestra en la figura 29. Además de todos los elementos mencionados anteriormente se añaden condensadores de 15 nF en cada patilla de control de los integrados y otro de 100uF para el desacoplo entre alimentaciones.

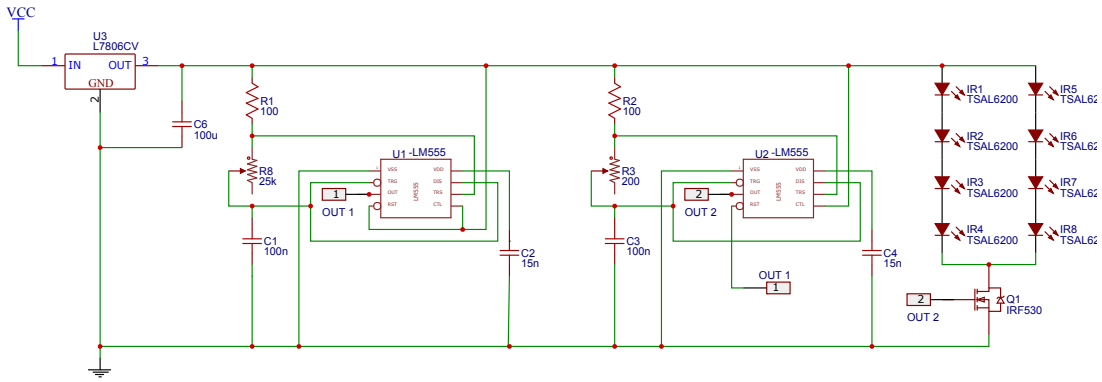


Figura 29: Esquema electrónico de las balizas.

Se muestra una de las tres balizas construidas en la figura 30.

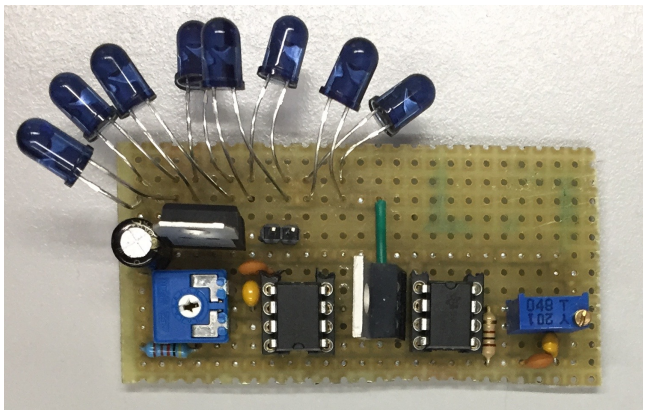


Figura 30: Imagen de la baliza construida.

Sistema de giro.

Se utiliza un motor paso a paso de 12V, sacado de una impresora 3d; se controla con un A4988 que está conectado a la mini-dk2 y a su vez a la alimentación general del robot, ya que el motor necesita 12 voltios para moverse. Una de las ventajas de este controlador es que se puede regular la corriente consumida, debido al ligero peso de los sensores, basta con 50mA para girarlos. El esquema de conexión del controlador es el de la figura 31.

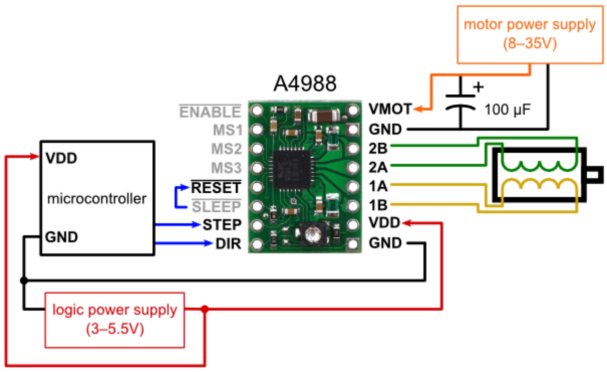


Figura 31: Esquemático y conexionado del controlador A4988.

Para sujetar la estructura del motor al robot se diseña una pieza en 3D, al igual que para sujetar los sensores de detección. Éstos, por necesidades del concurso se fijan a la plataforma que gira con termofusible.

Se utiliza un circuito detector de mando a distancia TSOP4838, su esquema interno se muestra en la figura 32. La conexión es sencilla, la alimentación se realiza directamente con la mini-dk2 y la salida se lleva a un pin en modo captura.

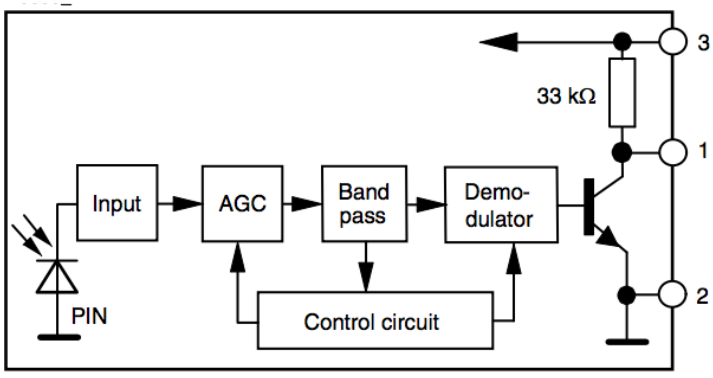


Figura 32: Esquemático del sensor TSOP4838.

Consta de un sistema interno de acondicionamiento de la señal. Este recibe exclusivamente señales de 38kHz, por lo que a la salida se observa una señal cuadrada de la frecuencia modulada de la baliza emisora correspondiente. En la figura 33, se muestra la señal obtenida cuando el sensor apunta hacia la baliza que emite una señal de 500Hz.

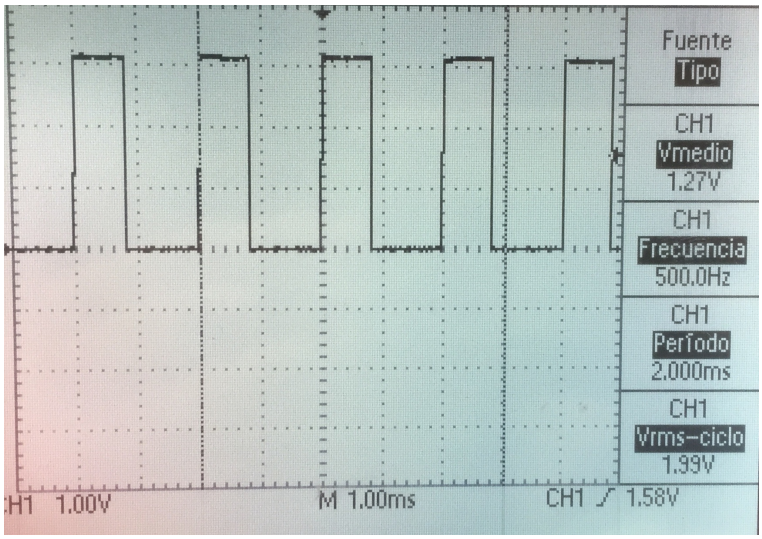


Figura 33: Señal recibida por el sensor TSOP4838.

3.1.3 Implementación Software

El control del movimiento del motor es sencillo debido al controlador utilizado. Se utilizan dos pines para realizar el control, en uno de ellos se envía un pulso cada vez que se desea avanzar un paso y en el otro se selecciona la dirección de giro. Se requiere que el motor dé una vuelta por segundo y el método seleccionado para este movimiento es el de medio paso. Para generar los pulsos, se modifica la salida del pin 1.14. Para el cambio de dirección, una vez que se ha completado la vuelta, se cambia el valor del pin 1.15. Cada vez que se envía un pulso al controlador, significa que el motor se ha movido 0.90 grados, entonces, se almacena en un array la información que proporciona el sensor asociado a la posición del motor en ese instante.

```
if(step){
   LPC_GPIO1->FIOSET|=(1<<(14));
   step=0;}
else{
   LPC_GPIO1->FIOCLR|=(1<<(14));
   step=1;}
```

```
if(direction)
   LPC_GPIO1->FIOSET|=(1<<(15));
else
   LPC_GPIO1->FIOCLR|=(1<<(15));

steps_left--;
if(step){
pasos++;
datos[pasos][0]=abs(N);
datos[pasos][1]=pasos;
```

Figura 34: Código utilizado para el movimiento del motor.

La variable “step” cambia para la generación del pulso, cada vez que se entra en la interrupción se varía el nivel del pin, por lo que es necesario interrumpir dos veces por cada paso del motor. Esto supone interrumpir cada 1.25ms. La variable “dirección” determina el sentido de giro. En el array “datos” se almacena tanto posición del motor, como la información contenida en “N”. Cada baliza tiene una “N” asociada, que se corresponde con una frecuencia de parpadeo.

Cada vez que se completa una vuelta, se realiza, en la función principal del código, el cálculo de la posición del robot. Lo primero es determinar los ángulos ente las balizas, a partir de la información contenida en “datos”. Entonces, se utiliza un algoritmo para conocer las distancias de cada una de las balizas al centro del robot. Conocidas estas distancias, por medio de trilateración, se determinan las coordenadas x e y de la posición del robot dentro del campo. Estos cálculos de posición se realizan en una función llamada “calculo_posicion” que se encuentra en el código al final del programa, recibe como argumentos los tres ángulos y la dirección de las variables “x” e “y” que están definidas como punteros. Una vez que los valores de las coordenadas están calculados se envían a través del puerto serie.

3.1.4 Pruebas y resultados

Durante la realización del proyecto se realizan diferentes pruebas en diferentes etapas.

Las primeras son durante la construcción de las balizas, hasta llegar al resultado final se produjeron cambios en el diseño, como controlar el segundo integrado con la entrada RESET, introducir un regulador de tensión en el sistema, añadir un condensador entre las alimentaciones y la colocación de los leds en paralelo.

Una vez creadas las balizas comienzan las pruebas del sensor. Se determina que, tratándose de distancias inferiores a los tres metros, las señales que llegan al sensor son exactamente iguales, tanto cerca como lejos.

El movimiento del sistema, inicialmente, se realizaba con un motor que tenía un paso mucho mayor, al realizar pruebas con el sistema, se determina que el error introducido por un paso mayor a un grado, no es aceptable, por lo que se cambia de modelo.

Por último, se realizan pruebas del sistema completo, situándolo en un punto y tomando la posición y los ángulos que obtiene. El código es modificado para que el resultado sea correcto en cualquier posición dentro del campo, se tiene en cuenta el sentido de giro, la posición de cada una de las balizas y el error de los tres ángulos es repartido de forma que el algoritmo funcione. En el centro del campo se obtienen las medidas representadas en la tabla 1.

Coordenada	Muestra 1	Muestra 2	Muestra 3	Muestra 4	Muestra 5
x	1.85	2.11	1.93	1.75	1.88
y	0.91	0.82	0.78	0.89	0.87

Tabla 1: Resultados para el robot situado en el centro del campo.

Se toman muestras de más puntos en el campo, cercanos a un lado o al otro, los resultados son similares, cuando está situado más cerca del origen se obtiene lo mostrado en la tabla 2.

Coordenada	Muestra 1	Muestra 2	Muestra 3	Muestra 4	Muestra 5
x	0.51	0.49	0.63	0.65	0.48
y	1.14	0.89	1.22	0.98	0.96

Tabla 2: Resultados para el robot situado al final del campo.

Así como en la Tabla 3, se muestran los datos cuando el robot se encuentra más alejado del origen y situado cercano al eje x.

Coordenada	Muestra 1	Muestra 2	Muestra 3	Muestra 4	Muestra 5
x	2.51	2.43	2.55	2.43	2.48
y	0.29	0.32	0.38	0.39	0.37

Tabla 3: Resultados para el robot situado al comienzo del campo.

3.1.5 Error del sistema

Se determina que la precisión del sistema no es suficiente para aportar una información fiable a la odometría, pero sí es útil para actualizar la posición del campo en la que se encuentra.

El sistema tiene un error debido al ángulo de giro del motor. Cada paso que se mueve el motor es de 0.9º, esto genera un error intrínseco en el sistema que no es despreciable. Para disminuir este efecto, el error se distribuye entre los tres ángulos obtenidos, de forma que siempre sumen 360º.

Además, en algunas medidas, hay fallos en la recepción de señal, por lo que se obtienen datos erróneos. Para solucionar estos errores se podría mejorar la lógica del código, así como implementar un sistema de sincronización de las balizas de forma que no emitiesen al mismo tiempo.

3.2 Posición oponente

Conocer la posición del robot oponente es crucial en la competición, ya que uno de los principales requerimientos es que no se produzca ninguna colisión. Para evitar que esto suceda existen dos mecanismos: Sensores de proximidad de unos 40cm y el sistema de baliza de tipo faro; el primero es necesario para evitar colisiones inminentes y el segundo proporciona a la estrategia una información, actualizada cada segundo, sobre la posición exacta del centro del otro robot.

3.2.1 Diseño teórico

El sistema está basado en un sensor que gira 360 grados, primero en un sentido y luego en otro. Emite una luz que se refleja en una baliza pasiva cuando se encuentra apuntando hacia esta. Si la baliza está lejos, el tiempo que la luz está incidiendo en ella es menor que cuando está cerca. A partir de este principio es posible calcular la distancia a la que se encuentra la baliza, ya que se conoce su diámetro. El sensor proporciona un

nivel alto cuando no recibe señal, y un nivel bajo cuando está detectando algo. Midiendo los pulsos que genera este sensor se determina la distancia.

El ángulo correspondiente con el tiempo en el que la baliza está siendo detectada se define como ángulo de detección. Éste es conocido, ya que el motor se mueve por pasos, cada paso se corresponde con un ángulo. A partir de este ángulo se determina la distancia al sensor siguiendo la expresión de la ecuación [24], donde "D" es el diámetro de la baliza y "α" el ángulo relativo.

$$d = \frac{D/2}{\tan\left(\frac{\alpha}{2}\right)} \tag{24}$$

Como ya se ha mencionado, para conocer el ángulo que ha girado el motor se utilizan los pasos del mismo. Esto supone una ventaja de los motores paso a paso frente a los DC, por este motivo se ha decidido la utilización de este motor, ya que evita la necesidad de usar un encoder.

3.2.2 Implementación Hardware

Se utiliza un sensor fotoeléctrico con sistema difuso, el elegido por cuestiones de presupuesto y diseño es el 42JS-PNP. Este se alimenta a 12V al igual que el controlador del motor. La señal que genera se lleva a un pin de la placa en modo interrupción, es necesario reducir la tensión de esta señal de salida, de 12V a 3.3V, que es la tensión de entrada requerida de la mini-dk2, para esto se utiliza un divisor de tensión. El sensor proyecta un haz de luz con un ángulo de apertura demasiado grande, por lo que es necesario reducirlo. Para ello, se utiliza una pieza impresa en 3D, así se consigue que el ancho de la luz reflejada en el catadióptrico sea menor y esto implica una mayor precisión del sistema.

3.2.3 Implementación Software

Se habilita una interrupción por flanco de bajada, una vez llega este flanco, generado por la detección de la baliza pasiva, se cambia a interrupción por flanco de

subida y se toma el valor de los pasos en ese momento. El flanco de subida, generado por el sensor cuando se deja de detectar la baliza, interrumpe de nuevo y se toma de nuevo el valor de los pasos. La resta de estos valores proporciona el número de pasos relacionado con el ángulo de detección de la baliza. Antes de salir de la interrupción se selecciona de nuevo flanco de bajada. Al final de cada vuelta, en la función principal del programa se calcula la distancia a partir de los parámetros ya conocidos.

3.2.4 Pruebas y resultados

Primeramente, se comprueba el funcionamiento del sensor; para ello se mide con el osciloscopio la señal que se obtiene cuando el sensor detecta un objeto, ésta es de nivel bajo. Por otro lado, cuando no se detecta ningún objeto se obtiene un nivel alto.

La siguiente prueba es de distancia, se determina que la distancia de detección de objetos es suficiente para nuestro sistema, de más de dos metros.

Por último, se realizan pruebas con el sensor ya incorporado en el robot. Se mide con el osciloscopio la salida. Se obtienen gráficas que verifican los estudios teóricos. En la figura 35, se muestra la señal recibida por el sensor cuando la baliza se encuentra alejada, por otro lado, en la figura 36, se muestra la señal cuando la baliza se encuentra cercana. Estas medidas se realizan mientras el sensor está girando.

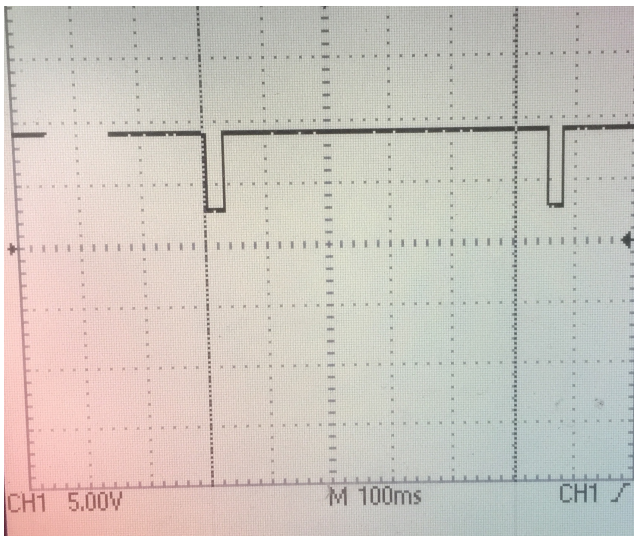


Figura 35: Señal recibida por el sensor 42JS-PNP con la baliza lejana.

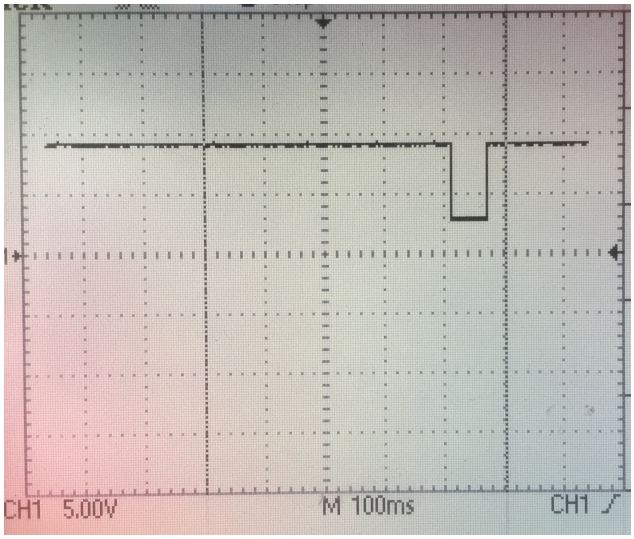


Figura 36: Señal recibida por el sensor 42JS-PNP con la baliza cercana.

3.2.5 Error del sistema

El sistema empleado ha sido probado en trabajos de años anteriores, por lo que en un primer momento se tomó como una elección segura para conocer la posición del oponente. Debido a la no disponibilidad del sensor utilizado en trabajos previos, se adquirió un sensor similar, pero con limitaciones de distancia y precisión.

El primer paso para la implementación es comprobar la detección de la baliza. Esta baliza fue creada en plástico, con la impresora 3D y cubierta por un papel catadióptrico adhesivo. Una vez se acondiciona la señal, obtenemos salidas de 3.3V y 0V.

El siguiente paso es comprobar en qué punto el sensor comienza a detectar la baliza. El principal problema es el haz de luz que genera el sensor, es demasiado ancho e incide luz en la baliza que no debería ser tenida en cuenta. Aun incluyendo una pieza que reduce el ángulo de apertura, no se soluciona completamente el problema.

Finalmente se consigue un sistema que a partir de una función lineal experimental nos devuelve una distancia aproximada de la posición del oponente, de forma que podemos saber su posición con un error de ±10cm. Esto es aceptable en nuestro caso ya que en el caso se encuentren ambos robots demasiado juntos, actuarán los sensores Sharp de proximidad.

4 Estructura mecánica

El sistema de posicionamiento es el único que necesita de la construcción una estructura mecánica. Se encuentra situado en la parte superior del robot, sobre una plataforma sujetada por cuatro varillas de acero.

En la figura 37, se muestra la colocación de los elementos que componen el sistema giratorio. El motor va sujeto a la plataforma con una pieza impresa en 3D; en su eje se ajusta otra pieza impresa en 3D, ésta es la plataforma giratoria en la que están anclados los dos sensores. El sensor de posición soldado a una placa y atornillado a la plataforma y el sensor del oponente pegado con termofusible.

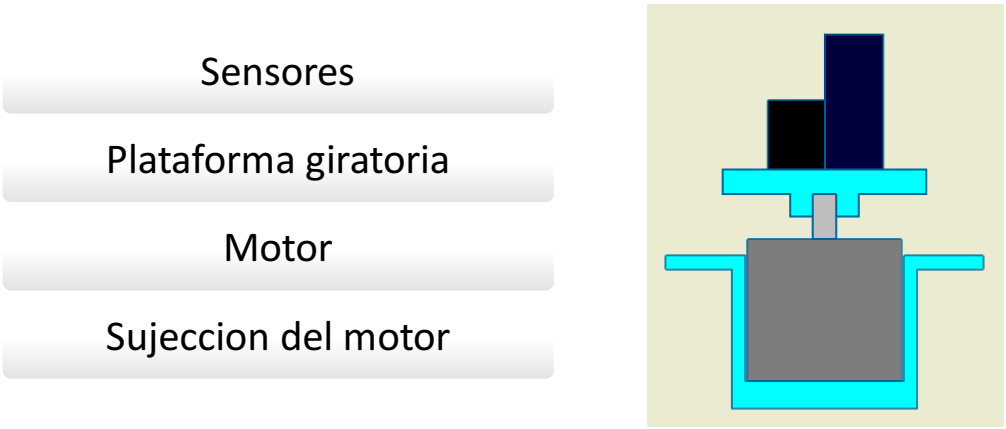


Figura 37: Esquema de la estructura mecánica.

La parte superior del robot es una plataforma metálica, sobre esta se atornilla la tarjeta de desarrollo. Las conexiones de la placa con el resto de elementos del sistema no necesitan una estructura, simplemente se llevan por medio de cables. Las pilas que alimentan la placa también cuentan con una pieza impresa en 3D, cuya función es mantenerlas sujetas al robot.

Se han diseñado dos piezas para los sensores para reducir el ángulo de detección del sensor. En el caso del receptor de infrarrojos, además es necesario utilizar cinta aislante negra para evitar reflejos de la señal. En la figura 38, la pieza del sensor de infrarrojos está a la izquierda y la pieza del sensor difuso está a la derecha.

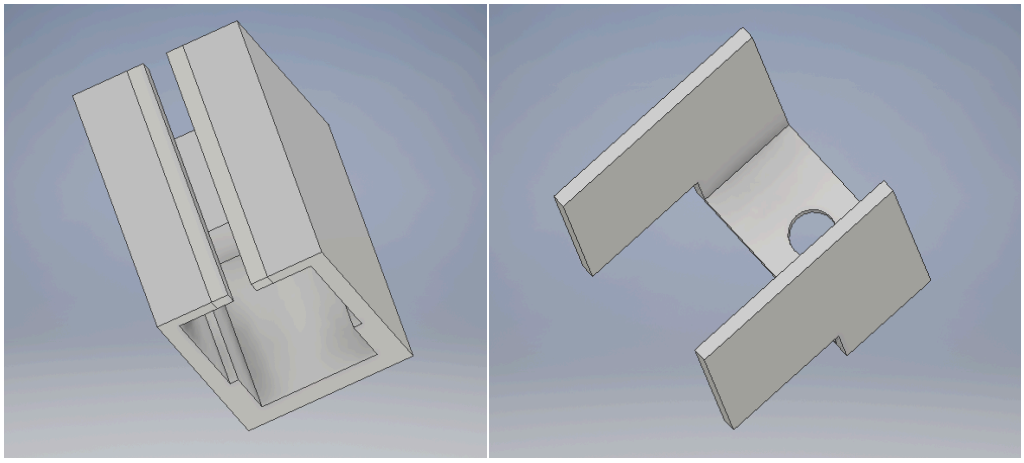


Figura 38: Piezas que reducen el ángulo de recepción de los sensores.

5 Conclusiones

Con la realización de este proyecto, el autor ha adquirido una serie de conocimientos que complementan la educación impartida durante el grado. Embarcarse en un proyecto anual, como es el de participar en un concurso de robótica, enriquece intelectualmente al alumno de ingeniería. La planificación de un gran proyecto, el trabajo en equipo y el uso de los conocimientos teóricos para realizar un sistema real son los principales retos que este desafío supone; estos aspectos proporcionan una gran experiencia al estudiante, que en un futuro será extrapolable al mundo laboral.

La localización de robots autónomos es uno de los principales problemas que se presentan en este tipo de competiciones. Siendo tan popular, ya se han desarrollado muchas soluciones al mismo. El reto es diseñar e implementar un sistema diferente a los demás utilizando los materiales a los que se tiene acceso. Los resultados no son del todo satisfactorios, ya que no se consigue la precisión esperada desde el principio.

En lo respectivo a la comunicación interna del robot, se ha conseguido el objetivo; un sistema funcional que permita comunicar las diferentes partes. Además del correcto funcionamiento, este documento sirve de guía para la implementación del sistema en otras aplicaciones futuras.

Presupuesto

Introducción

Este presupuesto engloba los costes de todos los materiales utilizados y el tiempo de trabajo dedicado. Las licencias de las aplicaciones utilizadas son proporcionadas por la UAH. No se incluye el tiempo dedicado a desarrollar el documento.

Coste de mano de obra

El tiempo de trabajo dedicado al diseño y construcción del sistema. No se tiene en cuenta la etapa previa de estudio del problema.

Trabajador	Sueldo/h	Número de horas	Coste
Ingeniero Electrónico Industrial	35€/h	160 h	5600€

Tabla 4: Coste de mano de obra.

Presupuesto material

Los materiales necesarios para la realización del proyecto se encuentran en la Tabla 5. Estos costes incluyen cualquier gasto de envío.

Artículo	Proveedor	€/unidad	Unidades	Coste
Potenciómetros	Electrónica Alcalá	0,25€	10	2,5€
Papel catadióptrico	Aliexpress	3€	1	3€
Emisor-Receptor luz difusa	RS-componets	42,5€	1	42,5€
Fototransistor TSOP4838	Electrónica Alcalá	1,3€	3	3,9€
Diodos IR TSAL6200	Amidata	0,1€	50	5€
Placa de soldadura	Electrónica Alcalá	2,5€	2	5€
Elementos pasivos	Electrónica Alcalá	10€	1	10€
Integrado 555	Electrónica Alcalá	0,5€	8	4€

Transistor mosfet IRF530	Electrónica Alcalá	1€	5	5€
Motor paso a paso	Electrónica Alcalá	12,5€	1	12,5€
Controlador A4988	Aliexpress	4,5€	2	9€
Baterías	Hobby Kings	5€	3	15€
Cable	Electrónica Alcalá	5€	1	5€
PLA	Amazon	10€	1	10€
Conversor DC-DC L7806	Electrónica Alcalá	2€	4	8€
Mini-dk2	Aliexpress	44€	1	44€
Transceptor 2551	RS-Components	3,5€	3	10,5€
Placa spi-can	DX-Componets	5€	1	5€
Total				159,9€

Tabla 5: Coste de materiales.

Coste global

Se muestra la distribución del coste global del proyecto en la figura 37. El coste de mano de obra supone el 97% del coste total del proyecto.

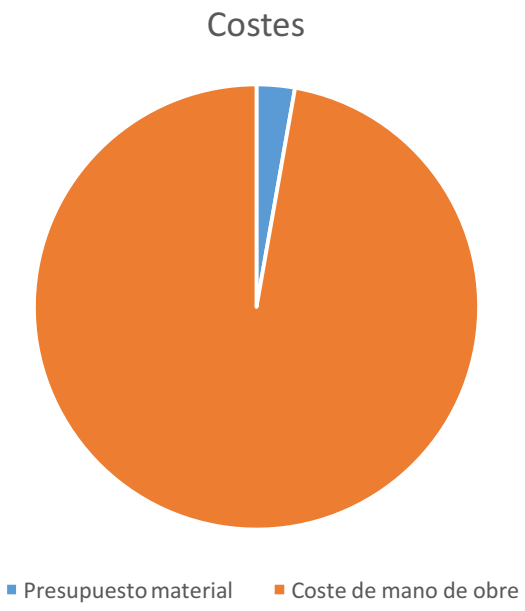


Figura 39: Gráfico de costes.

El coste total del proyecto es de 5759,9€.

Pliego de condiciones

Este pliego de condiciones contiene el conjunto de características que deben tener los elementos y materiales necesarios para la realización de este proyecto. Se realizan especificaciones para cuatro apartados:

Tarjetas de desarrollo

- Tarjeta de desarrollo que cuente con la interfaz de puerto serie integrada, debe tener al menos tres timers y diez pines entrada/salida disponibles. En este proyecto se utiliza la mini-dk2. La utilización de raspberry pi viene dada por requisitos del equipo.

Balizas

- Diodos led de un alcance mayor de 5 metros. En este proyecto se han utilizado los diodos TSAL6200.
- Receptor de un alcance mayor de 5 metros. En este proyecto se ha utilizado un receptor de mando a distancia TSOP4838.
- Sensor de distancia en el que se pueda implementar un sistema de detección por luz difusa, la distancia mínima de detección debe ser de 5 metros. En este proyecto se ha utilizado un sensor 42JS-PNP.
- Motor paso a paso que pueda configurarse de tal manera que cada paso corresponda con menos de 1º de giro. Motor de corriente continua con un encoder que mida la velocidad de giro. En este proyecto se ha utilizado un motor paso a paso NEMA-17.

Comunicación

- Sistema de comunicación que pueda trasmitir 1byte/ms, robusto ante interferencias y compatible con la tarjeta raspberry pi. En este proyecto se utiliza comunicación a través del puerto serie usando el protocolo UART.

Alimentaciones

- Alimentación de 12 voltios para alimentar al motor y a uno de los sensores. En este proyecto se utiliza una batería de Li-Po.
- Alimentación de 3.3 a 5 voltios para la tarjeta mini-dk2. En este proyecto se utilizan cuatro pilas alcalinas.
- Alimentación de 8 voltios para las balizas activas. En este proyecto se utilizan dos pilas de litio.

Planos

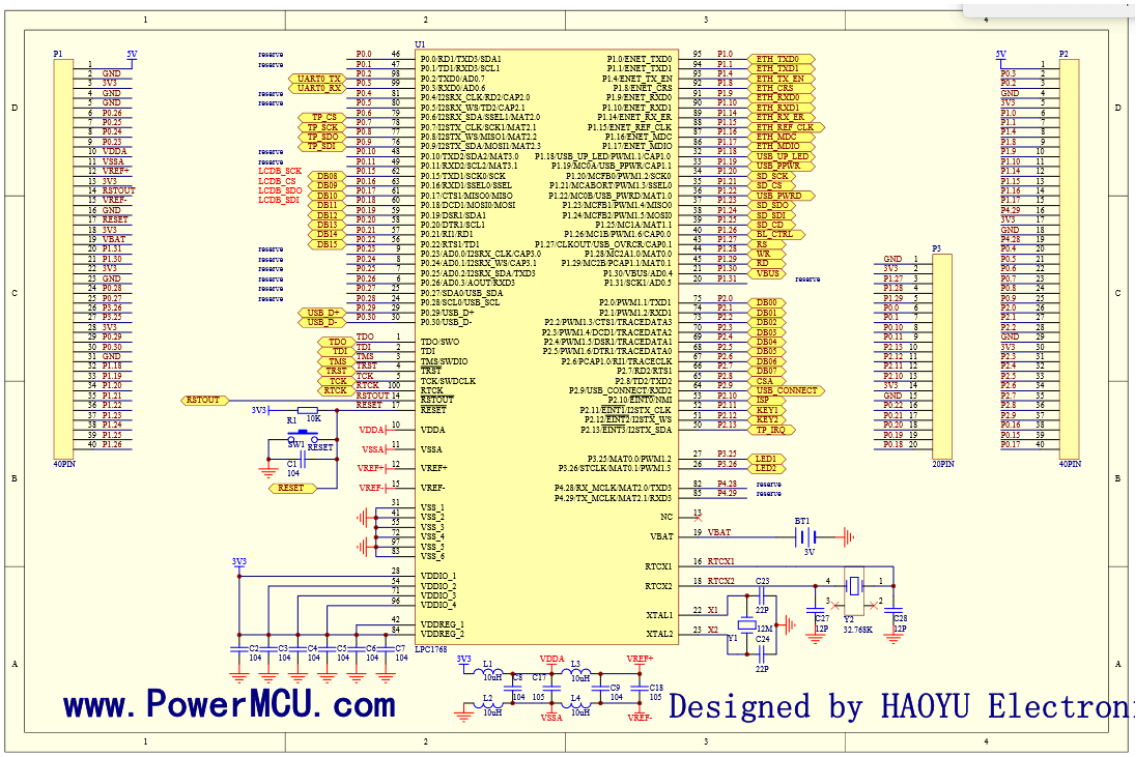


Figura 40: Esquema electrónico mini-dk2 [3].

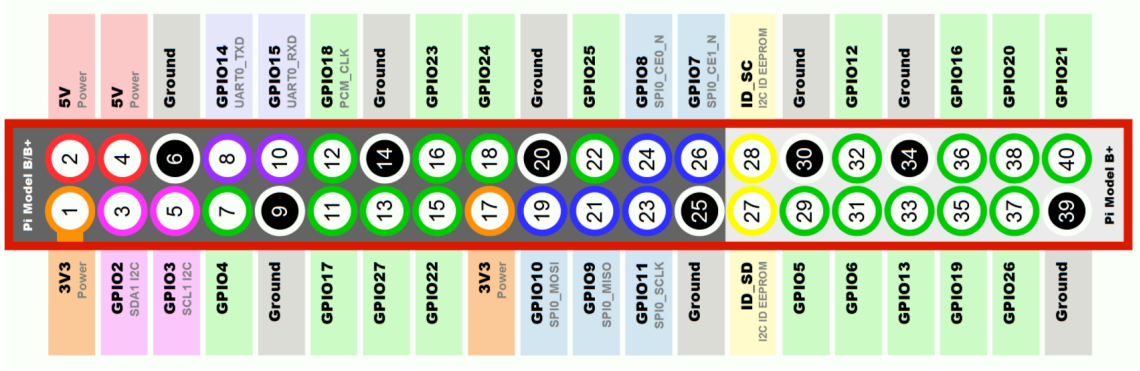


Figura 41: Esquema electrónico raspberry pi [12].

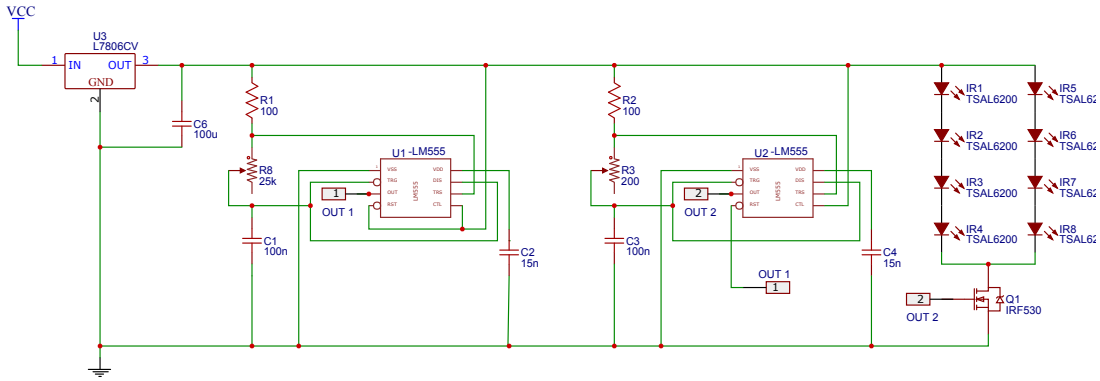


Figura 42: Esquema electrónico de las balizas.

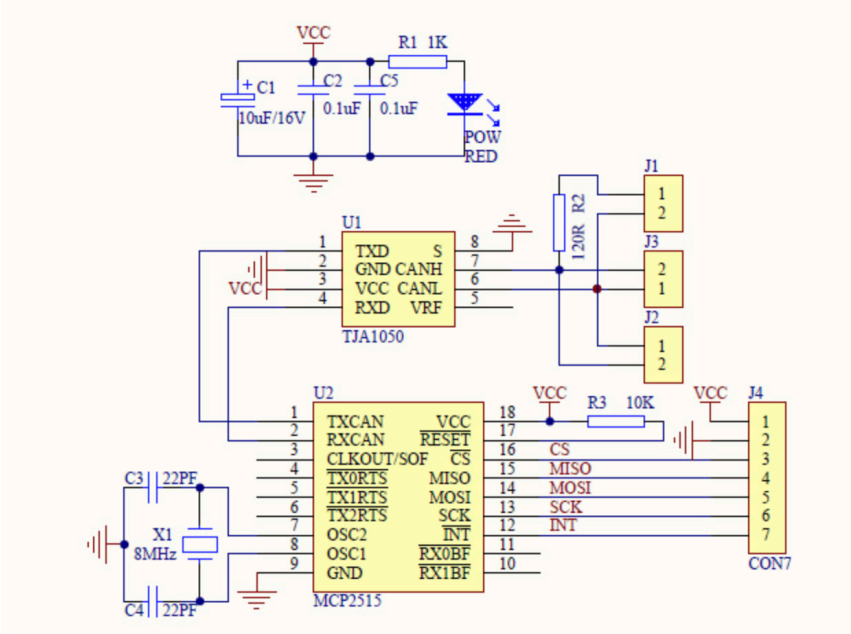


Figura 43: Esquema electrónico de la tarjeta spi-can.

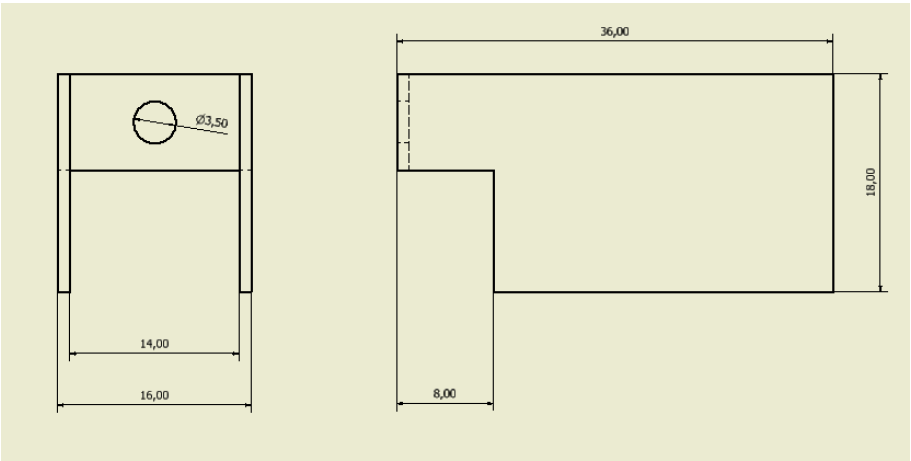


Figura 44: Pieza para el sensor difuso.

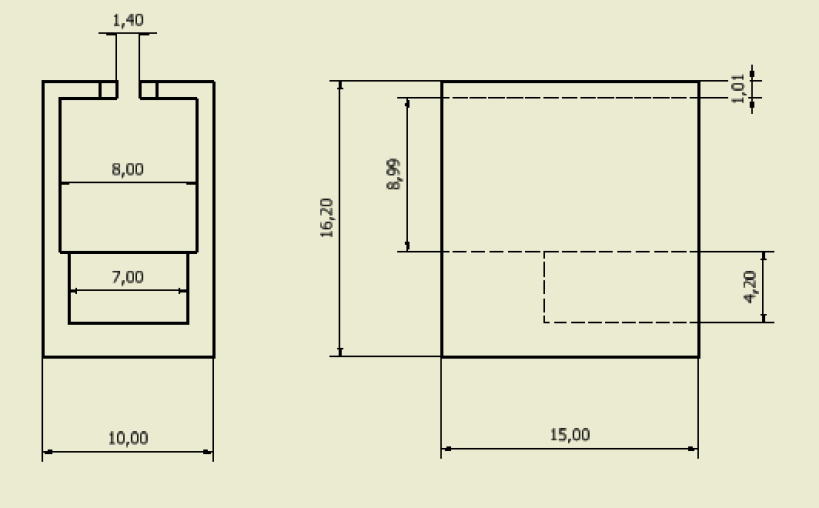


Figura 45: Pieza para el sensor de infrarrojos.

Código

Programa principal

```
#include <LPC17xx.H>                /* LPC17xx definitions                */
#include <Math.h>
#include <Stdlib.h>
#include <Stdio.h>
#include "Calculos.h"
#include "uart.h"

#define PCLK 25e6

int step=0;
int arm;
int datos [400][2];
int N=0, Ni=0, N1=82500,N2=50300,N3=62500,NX=1000,b1=0,b11=0,b1i=0,b2=0,b22=0,b2i=0, b3=0,
b33=0,b3i=0;
int i,hello;
int pasos,flanco=0,p1,pasos_aux,M;
int xr,yr,D,D_ans,D_aux,A,fin;

float steps_left=800,total=800;
float pulse=0;
uint8_t direction = 0;
float alpha,betha,theta,x,y,distancia,angle;

uint32_t flanc=0;
uint8_t isrmask=0;

char cadena[34];
char buffer[30];
char *ptr_rx;
char rx_completa;
char *ptr_tx;
char tx_completa;


// /*---------------------------------------------------------------------------
// *   Interrupts config
// *---------------------------------------------------------------------------*/
  void inic_EINTs(void)
 {

          LPC_PINCON->PINSEL4 |= (1<<22);       //Habilitamos P2.11 como EINT1

          LPC_PINCON->PINSEL4 &= ~(1<<23);

          LPC_SC->EXTMODE |= (1<<1);            //EINT1 por flanco

          LPC_SC->EXTPOLAR &= ~ (1<<1);         //EINT1 por flanco de bajada

          NVIC_SetPriority(EINT1_IRQn,1);       //EINT1 prioridad 2

          NVIC->ISER[0]=(1<<19);                //EINT1 interrupcion habilitada
 }
```

```
/*-------------------------------------------------------------------------
*   External 1 Interrupt Service Routine
*-------------------------------------------------------------------------*/
void EINT1_IRQHandler()

{
        LPC_SC->EXTINT |= (1<<1);
        if(flanco==0){

                LPC_SC->EXTPOLAR |=  (1<<1);          //EINT1 por flanco de subida
                p1=pasos;
                flanco = 1;
        }

        else{
                LPC_SC->EXTPOLAR &= ~ (1<<1);         //EINT1 por flanco de bajada
                flanco = 0;
                M=(pasos-p1);
                p1=0;
        }
}

/*-------------------------------------------------------------------------
*   Timer 0 Config
*-------------------------------------------------------------------------*/
void inic_TIMER0(void)

{
        LPC_SC->PCONP|=(1<<1);                        //Power ON
        LPC_PINCON->PINSEL3|= (0x03<<20);             //Habilitado P1.26 como CAP0.0
        LPC_TIM0->PR=0;                               //Preescaler siempre desborda
        LPC_TIM0->CCR|=(0x06<<0);                     //CAP0.0 Enable and interrupt falling edge
        NVIC_EnableIRQ(TIMER0_IRQn);                  //Habilitar interrupciÛn
        NVIC_SetPriority(TIMER0_IRQn,1);              //Prioridad 3
        LPC_TIM0->TCR=0x01;                           //Habilitar contadores
}

/*-------------------------------------------------------------------------
*   Interrupt Timer 0
*-------------------------------------------------------------------------*/
 void TIMER0_IRQHandler()
                        //Leer ADC, cambiar frec_LED
{
         isrmask=LPC_TIM0->IR;
         LPC_TIM0->IR|=isrmask;                       //Borramos el flag

        N=(flanc-LPC_TIM0->CR0);
        flanc=LPC_TIM0->CR0;

}


/*-------------------------------------------------------------------------
*   Timer 1 Config
*-------------------------------------------------------------------------*/
void inic_TIMER1(void)                                //Motor
{
        LPC_SC->PCONP|=(1<<2);                        //Power ON
        LPC_PINCON->PINSEL3|= (0x03<<12);             //Habilitado P1.22 como MAT1.0
        LPC_TIM1->PR=0;                               //Preescaler siempre desborda
```

```
	LPC_TIM1->MR0=(PCLK/800)-1;		//frecuencia 100hz
	LPC_TIM1->MCR|=(1<<0);			//Interrupcion tras MAT1.0
	LPC_TIM1->MCR|=(1<<1);			//Reset contador tras MAT1.0
	LPC_TIM1->EMR|=(1<<0);			//Pin MAT1.0 puesto a uno inicialmente
	LPC_TIM1->EMR|=(0x03<<4);		//Pin MAT1.0 a toggle
	NVIC_EnableIRQ(TIMER1_IRQn);		//Habilitar interrupcion
	NVIC_SetPriority(TIMER1_IRQn,1);	//Prioridad 1
	LPC_TIM1->TCR|=(1<<0);			//Habilitar contadores
}

/*----------------------------------------------------------------------------
*  Interrupt Timer 1
*----------------------------------------------------------------------------*/
void TIMER1_IRQHandler()
			//Variar ciclo de trabajo del motor
{

	LPC_TIM1->IR|= (1<<1);			//Borramos el flag


	if(steps_left>0){

		if(step){
			LPC_GPIO1->FIOSET|=(1<<(14));
			step=0;}
		else{
			LPC_GPIO1->FIOCLR|=(1<<(14));
			step=1;}

		if(direction)
			LPC_GPIO1->FIOSET|=(1<<(15));
		else
			LPC_GPIO1->FIOCLR|=(1<<(15));

		steps_left--;

		if(step){
			pasos++;
			datos[pasos][0]=abs(N);
			datos[pasos][1]=pasos;
			N=0;}
			}
		else {
			LPC_TIM0->TCR|=(1<<1);		//Reset
			LPC_TIM0->TCR&= ~(1<<1);		//Replay
			direction=!direction;
			steps_left=800;
			pasos=0;
			fin=1;}


/*----------------------------------------------------------------------------
 *      Main: Initialize and start RTX Kernel
 *----------------------------------------------------------------------------*/
int main (void)  {			/* Program execution starts here	*/

	ptr_rx=buffer;
	LPC_GPIO1->FIODIR|=(0xF<<14);
	LPC_GPIO1->FIOSET|=(1<<(16));
```

```
	inic_TIMER0();
	inic_TIMER1();
	inic_TIMER2();
	inic_EINTs();
	inic_PWM();
	uart0_init(19200);

while(1) {

	if((flanco==0)&&(M<70)&&(M>30)){
	distancia=2104.56-(36.575*M);                    //Calculo de distancia
	angle=3.1415*(pasos-((M)/2))*0.45/180;
	D_ans=D;
	D=(int)(distancia);
	if(abs(D-D_ans)<80)
	D=D_ans;


	A=(int)(angle);
	M=0;

	}

	if(fin==1){

		fin=0;

		for(Ni=0;Ni<400;Ni++)

			{

			if((datos[Ni][0]<(N1+NX))&&(datos[Ni][0]>(N1-NX)))
			{
				if(b1i==0){
					b11=datos[Ni][1];
					b1i=1;}
				else
					b1=datos[Ni][1];
			}

			if((datos[Ni][0]<(N2+NX))&&(datos[Ni][0]>(N2-NX)))
			{
				if(b2i==0){
					b22=datos[Ni][1];
					b2i=1;}
				else
					b2=datos[Ni][1];
			}

			if((datos[Ni][0]<(N3+NX))&&(datos[Ni][0]>(N3-NX)))
			{
				if(b3i==0){
					b33=datos[Ni][1];
					b3i=1;}
				else
					b3=datos[Ni][1];
			}

			}
```

```
if((abs(b1-b11))<20)
        b1=(b11+b1)/2;
if((abs(b2-b22))<20)
        b2=(b22+b2)/2;
if((abs(b3-b33))<20)
        b3=(b33+b3)/2;

b1i=b2i=b3i=0;

if(direction==0)                                    //Dependiendo de la dirección se
suman o se restan unos angulos a otros

        {
        if((b2-b1)>0)
        alpha=(b2-b1)*0.45;
        else
        alpha=(b2+(total-b1))*0.45;

        if((b3-b2)>0)
        betha=(b3-b2)*0.45;
        else
        betha=(b3+(total-b2))*0.45;

        if((b1-b3)>0)
        theta=(b1-b3)*0.45;
        else
        theta=(b1+(total-b3))*0.45;
        }
else
        {
        if((b1-b2)>0)
        alpha=(b1-b2)*0.45;
        else
        alpha=(b1+(total-b2))*0.45;

        if((b2-b3)>0)
        betha=(b2-b3)*0.45;
        else
        betha=(b2+(total-b3))*0.45;

        if((b3-b1)>0)
        theta=(b3-b1)*0.45;
        else
        theta=(b3+(total-b1))*0.45;

        }


calculo_posicion(alpha,betha,theta,&x,&y);          //Calcula la posición a partir de los
angulos, actualiza el valor de las coordenadas, que están definidas como punteros

        xr=(int)(x*1000);
        yr=(int)(y*1000);

if(D==0)
D=10000;

sprintf(cadena,"%d"","%d""\n",xr,yr);               //imprime las variables como
cadenas
tx_cadena_UART0(cadena);
```

```
                    while(tx_completa==0);
                    tx_completa=0;

                    }
          }
}

/*-------------------------------------------------------------------------
 * end of file
 *-------------------------------------------------------------------------*/
```

Funciones de cálculo

```
#include <LPC17xx.H>                /* LPC17xx definitions                */
#include "Calculos.h"

#define PCLK 25e6

float grados_right,grados_left;

void calculo_posicion(float alpha, float betha, float theta, float *x, float *y)
 {
          int aux_exit;
          float pi=3.14159265358979323846264338327;
          float theta_big;
          float theta_small;
          float a,i;
          float theta_ans;
          float theta_new;
          float d3_ans;
          float d1, d2, d3;
          float error;

          d1=d2=d3=0.1;
          i=3;
          alpha=alpha*pi/180;
          betha=betha*pi/180;
          theta=theta*pi/180;

if (alpha>pi)
{
   alpha=2*pi-alpha;
}
if (betha>pi)
{
   betha=2*pi-betha;
}

if (theta>pi)
{
   theta=2*pi-theta;
}
                    error=(2*pi)-(alpha+betha+theta);
                    alpha=alpha+(error/3);
                    betha=betha+(error/3);
                    theta=theta+(error/3);
```

```
while(d3<=3.1)
{

	theta_ans=theta_new;

	if (cos(alpha)>0)
{
d1=d3*cos(alpha) - sqrt(d3*d3 *cos(alpha)*cos(alpha) - d3*d3 + 10);
		if (d1<0)
			d1=d3*cos(alpha) + sqrt(d3*d3*cos(alpha)*cos(alpha) - d3*d3 + 10);
	}
	else
{
d1=d3*cos(alpha) + sqrt(d3*d3*cos(alpha)*cos(alpha) - d3*d3 + 10);
		if (d1<0)
			d1=d3*cos(alpha) - sqrt(d3*d3*cos(alpha)*cos(alpha) - d3*d3 + 10);
	}


	if (cos(betha)>0)
{
d2=d3*cos(betha) - sqrt(d3*d3*cos(betha)*cos(betha) - d3*d3 + 10);
		if (d2<0)
			d2=d3*cos(betha) + sqrt(d3*d3*cos(betha)*cos(betha) - d3*d3 + 10);
	}
	else
{
d2=d3*cos(betha) + sqrt(d3*d3*cos(betha)*cos(betha) - d3*d3 + 10);
		if (d2<0)
			d2=d3*cos(betha) - sqrt(d3*d3*cos(betha)*cos(betha) - d3*d3 + 10);
	}


		a=(4-(d1*d1)-(d2*d2))/(-2*(d1)*(d2));

 theta_new=acos(a);


if ((theta<theta_new)&&(theta>theta_ans))
{
	theta_big=acos(a)+0.001;
	theta_small=acos(a)-0.001;

	if ((theta<theta_big)&&(theta>theta_small))
	{
		*y=(4+(d1*d1)-(d2*d2))/4;
		*x=sqrt((d3*d3)-((*y-1)*(*y-1)));
		aux_exit=1;
	}
	theta_ans=theta_new=0;
	i=i/4;
	d3=d3_ans;;

 }
	if (aux_exit==1)
	{break;}
	d3_ans=d3;
	d3=d3+i/4;
}
```

Bibliografía

[1] http://www.microcontroller.com/ (Última vez consultada en 20/07/2017).

[2] https://www.raspberrypi.org/forums/viewtopic.php?f=44&t=141052 (Última vez consultada en 22/09/2017).

[3] https://www.keil.com/ (Última vez consultada en 03/06/2017).

[4] http://www.roboticaeducativa.org /(Última vez consultada en 20/06/2017).

[5] http://es.rs-online.com/web/ (Última vez consultada en 19/06/2017).

[6] http://roboticday.org/ (Última vez consultada en 14/06/2017).

[7] http://www.areatecnologia.com/ (Última vez consultada en 05/08/2017).

[8] García Oses, Alejandro (2015) (TFG) Diseño de una red CAN BUS con Arduino.

[9] Baliñas Santos, Javier (2016) (TFC) Manual de referencia para el desarrollo de robots de Eurobot.

[10] KEIL. Getting started. Building applications with rl-arm. (2009).

[11] http://www.dx.com/es/p/mcp2515-can-bus-module-tja1050-receiver-spi-module-blue434988?tc=EUR&gclid=CjwKCAjw3_HOBRBaEiwAvLBbor7lXFIwoQRjHwj0xTTPBn4oZtGL-4f2Ha4VOGdl6N4IQ8zGWibnSxoCadAQAvD_BwE#.Wd0G_BN-pAY (Última vez visitada 5/10/2017).

[12] https://www.raspberrypi-spy.co.uk/2012/06/simple-guide-to-the-rpi-gpio-header-and-pins/ (Última vez visitada 3/10/2017).

Universidad de Alcalá

Escuela Politécnica Superior

ESCUELA POLITECNICA SUPERIOR

Universidad de Alcalá