



TRABAJO DE FIN DE GRADO

*Grado en ingeniería electrónica Industrial y
Automática*

*SISTEMA DE LOCALIZACIÓN EN INTERIORES PARA
ROBOTS MÓVILES
(CONTINUACIÓN)*

Autor: Camilo Ledesma García–Ramos

Tutor: Jonay Tomás Toledo Carrillo

AGRADECIMIENTOS

A todos mis familiares y amigos sin los cuales no hubiera sido capaz de mantener la constancia en el estudio de este grado y a todos los profesores y profesoras que han despertado en mí el interés por el campo de la robótica y me han brindado los conocimientos necesarios para realizar esta investigación, muchas gracias.

Índice

RESUMEN.....	1
ABSTRACT	2
CAPÍTULO 1: INTRODUCCIÓN.....	3
1.1 Introducción.....	3
1.2 Objetivos del trabajo	6
1.3 Estructura del trabajo.....	7
CAPÍTULO 2: MARCO TEÓRICO	9
2.1 Explicación sobre el trabajo de referencia	9
2.1.1 Comunicación entre emisores	9
2.1.2 Receptor	10
2.2 Fusión sensorial:.....	10
2.3 LOCALIZACIÓN Y SISTEMAS SENSORIALES:	12
2.3.1 Localización.....	12
2.3.2 Sistema de localización a emplear	12
2.3.3 Trilateración.....	13
2.3.4 Determinación de la orientación	15
CAPÍTULO 3: HARDWARE UTILIZADO Y HERRAMIENTAS SOFTWARE EMPLEADAS ...	18
3.1 ELEMENTOS HARDWARE	18
3.1 .1 Sensores de ultrasonidos:.....	18
3.1.2 Módulos de radio NRF24L01:	20
3.1.3 ARDUINO	21
3.2 HERRAMIENTAS SOFTWARE EMPLEADAS.....	24
3.2.1. ARDUINO	24
3.2.2 AUTODESK INVENTOR 2020.....	25
3.2.3 REPETIER HOST	27
3.2.4 SLIC3R	28
CAPÍTULO 4: PROBLEMAS ENCONTRADOS, SOLUCIONES PLANTEADAS Y MEJORAS EN EL PROYECTO.....	29
4.1 CAMBIOS REALIZADOS EN EL HARDWARE DEL PROYECTO ANTERIOR.....	29
4.2 CAMBIOS REALIZADOS EN EL SOFTWARE DEL PROYECTO.....	32
4.2.1 Problemas de comunicación.....	32
4.2.2 Medición de distancias	33
4.3 DISEÑO 3D:	44
CAPÍTULO 5: CONCLUSIÓN Y PROPUESTAS DE MEJORA.	47
5.1 CONCLUSIÓN Y PROPUESTAS DE MEJORA	47
5.2 CONCLUSION AND FUTURE IMPROVEMENTS	48
ANEXO 1: Tabla de resultados.....	I

ANEXO 2: PRESUPUESTO	IV
ANEXO 3: PLANOS	VI
ANEXO 4: Códigos.....	VIII
ANEXO 5: Documentación.....	XXVII
Bibliografía	LIX

RESUMEN

En este proyecto se trata de desarrollar un sistema de localización para un robot de interiores con el objetivo inicial de calcular tanto la posición(x,y) del robot como la orientación del robot. En caso de conseguirlo, el robot podría moverse de forma autónoma por el plano.

Sin embargo, en este proyecto se debería partir de un previo sistema de localización que ya fuera capaz de localizarse y orientarse, aunque esto último con menor precisión. Por el contrario, al probar el trabajo anterior, con el objetivo de realizar mejoras y modificaciones, se encuentra que no es posible realizar la localización del robot puesto que ni siquiera el cálculo de la posición se puede realizar.

Teniendo estos problemas iniciales, se utilizarán sensores de ultrasonidos, módulos de radiofrecuencia y microcontroladores integrados en las placas de Arduino utilizadas, para intentar conseguir al menos un sistema que calcule la posición del robot con cierta precisión.

ABSTRACT

In this Project, the aim is to develop a robots location system for interiors which could be able to calculate both the position (x,y) of the robot and the orientation. If this is achieved the robot, would be able to move autonomously through the plane.

However, this Project should have started from a previous location system that should be able to locate and orient itself. No matter how, when testing the previous location system, aiming for making improvements, it is found that is not possible to perform the correct location of the robot since even the calculation of the position could not be performed.

Having this initial problems, ultrasound sensors, radiofrequency modules and microcontrollers integrated in the Arduino boards used, will be combined to try to achieve a system that calculates the position of the robot with a certain precision.

CAPÍTULO 1: INTRODUCCIÓN.

1.1 Introducción

El campo de la robótica móvil es un campo de investigación relativamente reciente, que tiene sus inicios en 1968 con la aparición del robot SHAKEY creado en el Stanford Research Institute (SRI). El nombre deriva de la palabra “shake”, en español “agitar”, puesto que el robot se movía bastante al caminar. Como primer robot que pretendía ser autónomo, contaba con multitud de sensores y cámaras para poder moverse por el entorno con seguridad. Esta información sería utilizada por un ordenador que se encargaría del procesamiento de las imágenes y, paralelamente, otro ordenador distinto integrado en el robot, combinaría la información aportada por los sensores para controlar los motores del mismo.

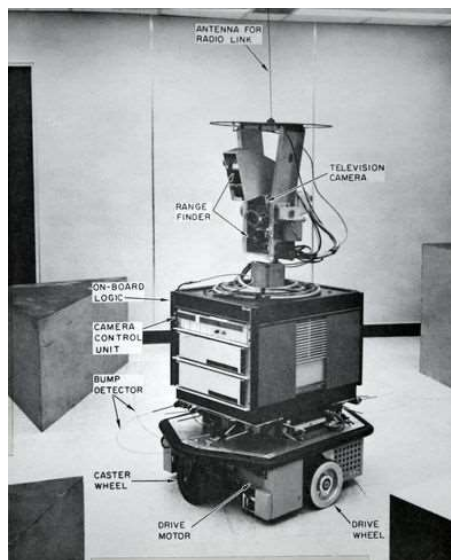


Imagen 1: Robot Shakey

En la actualidad podemos hablar de la existencia de robots humanoides que con el tiempo han ido ganando mayor autonomía y de automóviles autoguiados que no necesitan de la ayuda del ser humano para posicionarse, orientarse y moverse de manera independiente.

La existencia de estos robots y de estos avances tecnológicos sería impensable si no fuera por las investigaciones desarrolladas en torno a la estimación de la localización de robots móviles y los sistemas sensoriales.

En el campo de la robótica móvil, la localización de los robots es un problema al que se le da solución de diversas maneras. Los sistemas de localización que se utilizan en la actualidad están en su mayoría basados en la utilización de sistemas de posicionamiento global o GPS. De esta forma, teniendo una referencia global de la localización de un robot, resulta sencillo especificar una segunda posición al robot para que avance hacia ella. Sin embargo, este tipo de localización es útil únicamente en entornos abiertos o de fácil acceso para las señales utilizadas en los sistemas GPS.

Al no poder contar con la utilización de sistemas GPS, es necesario implementar sistemas de localización específicos para recintos cerrados o interiores. Puesto que no se tiene un sistema de referencia de posición como ocurre en los sistemas GPS, se debe disponer de un sistema sensorial potente para poder ubicar a un robot en este tipo de recintos.

Para posicionarse, el robot debe conocer su posición relativa al entorno en el que se esté moviendo. La posición del robot será hallada a través de un sistema de referencia externo que estará compuesto por un mínimo de tres puntos de referencia o nodos, tal y como ocurre en los sistemas GPS; pero en este caso estará compuesto por tres balizas emisoras de ultrasonidos.

El método de localización a emplear, por lo tanto, será a través de balizas que determinarán la localización absoluta del robot con una rápida frecuencia de muestreo, de tal manera que se podrá localizar al robot incluso cuando el mismo esté cambiando su localización. Las balizas empleadas serán balizas activas puesto que emitirán señales de ultrasonidos para conocer la distancia desde las mismas hasta el robot. Con lo cual, podemos especificar que las balizas empleadas serán balizas ultrasónicas.

Al utilizarse tres balizas simultáneamente, se hace necesario identificar a cada una de ellas de tal manera que el robot reconozca sin lugar a equivocaciones de donde proviene cada pulso de ultrasonidos. Para ello será utilizado un módulo de radio en cada baliza, así como en el robot, de tal manera que cada pulso vaya identificado con un mensaje de radio que indicará de donde proviene el mismo.

Las ventajas ofrecidas por este tipo de localización serán tanto conseguir un sistema de localización realmente económico como un sistema de localización sin una complejidad

computacional tan elevada como en alguno de los sistemas mencionados anteriormente.

Como desventaja, con respecto a otros sistemas de localización en interiores, puede señalarse que se hace necesario modificar el entorno para la colocación de las balizas que deben estar alimentadas en todo momento y conseguir una comunicación entre emisor y receptor directa, de tal manera que no haya ninguna interferencia que pueda influir en el cálculo de la posición.

1.1.2 Contextualización.

Actualmente, existen multitud de sistemas de localización en interiores con multitud de sensores diferentes utilizados en ellos. Se describirán una serie de estos a continuación:

Localización de robots en interiores basada en wifi: En función de la intensidad de la señal de wifi de distintos puntos de acceso distribuidos a lo largo de un recinto cerrado se podrá calcular la distancia a los mismos y tener una estimación de la localización en tiempo real.

Sistema de localización en interiores Situm RLTS: Permite estimar la posición de un smartphone en un recinto cerrado a través de conexiones tipo bluetooth, wifi o cualquier tipo de onda electromagnética que sea capaz de detectar un smartphone. Con esta técnica se consigue un margen de error de un metro, un margen prácticamente imperceptible para los humanos en recintos amplios como hospitales para los que está pensado este sistema de localización.

Sistema de localización en interiores basado en telemetría láser: Una vez hallado el tiempo de vuelo del pulso láser, será posible estimar la localización del objeto deseado o robot. Este tipo de sistema de localización está bastante extendido.

Sistemas de localización en interiores a través de cámaras omnidireccionales, como el robot Doris diseñado por la Universidad Politécnica de Madrid. Este sistema utiliza un único sensor para estimar su posición de tal manera que evita la fusión sensorial. Pese a que el proyecto se presenta como un sistema que sólo utiliza una cámara omnidireccional, también cuenta con la aportación de un sensor láser que, en caso de

tener menos ruido que la aportación recibida por la cámara, predominará como información de mayor importancia a utilizar en ese momento.

Debido a que se coordinará la información de distintos sensores, este sistema llevará asociada una cierta complejidad computacional característica de este tipo de sistemas constituidos por múltiples aportaciones de información de diversos sensores.

Teniendo contextualizado este proyecto con las tendencias de localización en la actualidad, podemos decir que el tipo de localización utilizado es una forma eficiente y económica de localizar un robot pese a necesitar de la implementación de la fusión sensorial.

Este proyecto también puede ser un proyecto útil para la educación en robótica de los sistemas sensoriales, un campo tan importante en este campo de ingeniería, ya que se podrá ver de una forma no compleja en qué consiste la fusión sensorial y cómo se puede implementar a la hora de localizar un robot móvil.

1.2 Objetivos del trabajo

En el proyecto que se trata, se proponen una serie de objetivos partiendo de un trabajo anterior realizado por D. Agustín León García, alumno de la Universidad de La Laguna al que se hace referencia en más ocasiones a lo largo de la redacción de esta memoria.

Este trabajo de referencia diseñaría un sistema de localización de robots móviles en interiores que, con la ayuda de tres emisores de ultrasónicos, conseguiría obtener la posición del robot en un plano marcado por la localización de los emisores de ultrasonidos previamente nombrados.

Como objetivos, se plantean mejoras en el sistema anterior en cuanto a la tolerancia a fallos de medida, robustez de los códigos empleados para la localización y además se añadiría la posibilidad de no solo conocer la posición en la que se encontraría el robot sino también la de conocer el ángulo con el que este se oriente para tener así un robot perfectamente localizado.

Entre los objetivos anteriormente mencionados, también se persigue obtener un sistema de localización de robots móviles en interiores de sencilla interacción con el

usuario y de un presupuesto relativamente modesto y una complejidad reducida en comparación con otros sistemas de localización que se utilizan en la actualidad.

Partiendo de un proyecto que, al retomarlo parece no funcionar, se intentará en la medida de lo posible lograr los objetivos previamente nombrados para mejorar este sistema de localización de robots móviles en interiores. Debido a los problemas de funcionamiento encontrados al retomar el trabajo anterior, el objetivo principal de este trabajo se reduce a la investigación de los problemas que no permiten que el sistema de localización funcione, así como buscar soluciones a la mayor parte de ellos.

1.3 Estructura del trabajo.

La memoria se estructura en un total de 5 capítulos que serán descritos brevemente a continuación:

En el capítulo 1, se hará una introducción al trabajo en la cual se explicarán los objetivos del mismo y se pondrá en contexto con las investigaciones y proyectos actuales de la localización en interiores.

El marco teórico del trabajo será descrito a lo largo del capítulo 2, donde se explicará cómo realizar el sistema de localización de robots en interiores sin atender a los aspectos prácticos.

A lo largo del tercer capítulo se explicará el marco práctico del proyecto, aclarando los componentes utilizados y los aspectos prácticos del trabajo.

El capítulo 4 centrará su desarrollo en los problemas encontrados al retomar el trabajo de fin de grado previo al que se explica en esta memoria. También se explicará que soluciones se aportan a algunos de esos problemas y qué mejoras se han aportado al sistema.

Por último, las conclusiones y propuestas de mejora para futuras investigaciones serán escritas en el quinto capítulo.

CAPÍTULO 2: MARCO TEÓRICO

2.1 Explicación sobre el trabajo de referencia

Este proyecto se basa en el trabajo de fin de grado realizado por Agustín León García, titulado “Sistema de localización de robots en interiores para robots móviles” que propone un sistema de localización de robots en interiores, basado en la emisión de ultrasonidos controlados por radiofrecuencia. Además, el sistema de localización diseñado es capaz de comunicarse con el robot a través de la herramienta ROS o “Robot Operating System” mediante la cual hará moverse al robot de un punto A a un punto B en el plano.

Se puede dividir el sistema de localización en dos bloques diferenciados, pero no independientes. Estos dos bloques serán el conjunto de emisores de ultrasonidos y el robot.

2.1.1 Comunicación entre emisores

En primer lugar, se tienen tres módulos emisores de ultrasonidos que, a través de módulos de radiofrecuencia conformarán un “anillo” emisor de ultrasonidos. El hardware de este primer bloque, está compuesto por tres transceptores de ultrasonidos fácilmente implementables con Arduino denominados HC-SR04. También consta de tres módulos de radiofrecuencia que deben encargarse de controlar la emisión de ultrasonidos por parte de los transceptores mencionados anteriormente. Ambos módulos irán conectados a una placa Arduino UNO.

El ciclo de emisión de ultrasonidos se realiza de la siguiente manera:

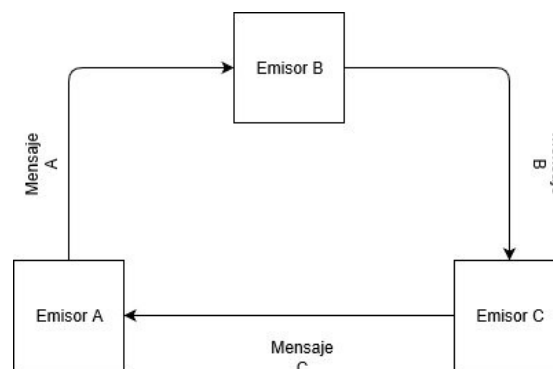


Imagen 2: Ciclo de emisión de mensajes entre emisores

El emisor A envía un mensaje "A" que debe ser recibido por el emisor B. Una vez el emisor B ha recibido el mensaje "A", éste envía un mensaje "B" que será recibido por el emisor C. Cuando este ha sido recibido, se emite el mensaje "C" para que el emisor A empiece el ciclo de nuevo. Si han pasado más de 10 segundos desde el último envío del mensaje "A" sin recibir el mensaje "C", el emisor A enviará de nuevo el mensaje "A". De esta manera, se consigue que los emisores A, B y C estén emitiendo pulsos permanentemente, comprobando que se ha recibido el mensaje de su emisor predecesor.

2.1.2 Receptor

En segundo lugar, y como elemento del sistema a localizar, tendremos al robot o receptor de ultrasonidos. El hardware empleado para la conformación de este elemento serán seis receptores de ultrasonidos, un módulo de radio y la placa Arduino MEGA 2560 a la que irán estos elementos conectados.

El robot recibirá los pulsos emitidos por los ultrasonidos situados en los emisores. A través de los pines de interrupción de la placa Arduino MEGA (2, 3, 18, 19, 20 y 21) se podrá calcular el tiempo que se tarda en recibir un pulso proveniente de los sensores de ultrasonidos situados en el receptor, con lo cual se calculará la distancia del robot a cada emisor y por consiguiente se podrá realizar la triangulación que permita hallar la posición (x,y) del robot en el plano. A través de la medida de estos tiempos también será posible calcular el ángulo de orientación del robot.

2.2 Fusión sensorial:

En este proyecto, se necesita de la aportación de hasta nueve sensores de ultrasonidos y tres módulos de radio. Esto es así puesto que hay datos que sin esta cantidad de sensores no serían recogidos y perderíamos precisión y exactitud a la hora de localizar a nuestro robot móvil, aparte de requerir un mínimo de tres sensores de ultrasonidos necesarios para la trilateración. En consecuencia, se trabajará en todo momento con una red multisensorial. A la unión de las aportaciones de los distintos sensores y el contraste de la información servida por los mismos se le denomina "fusión sensorial".

Los sensores de ultrasonidos no son dependientes entre sí directamente, simplemente combinarán su información para disponer de un entorno de trabajo sin puntos muertos,

en el cual los radios de los conos característicos de la propagación de la onda ultrasónica desde el sensor puedan coincidir en un punto, haciendo posible la trilateración.

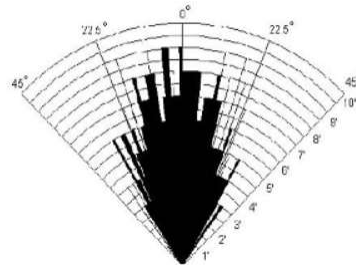


Imagen 3

En la imagen 3, perteneciente al datasheet del sensor de ultrasonidos HC-SR04 empleado en este proyecto, se muestra el ángulo de propagación de la onda ultrasónica. En el datasheet se especifica que funciona de manera más eficiente en un ángulo de propagación de 30 grados. Puesto que, por lo mencionado anteriormente, no se tendrá un ángulo de detección de 360 grados, los sensores de ultrasonidos deberán estar ubicados de la manera en la que se muestra en la imagen 3.

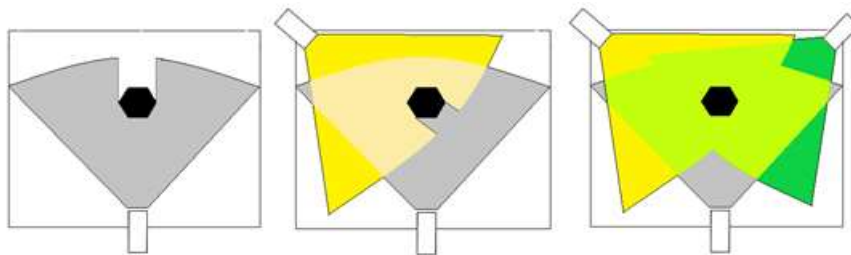


Imagen 4

En la imagen 4 también se puede observar como a través de la inclusión de distintos ultrasonidos en el sistema, se tiene una “visión” más precisa del entorno de trabajo y se tendrán más puntos en el mapa por los que el robot podrá moverse pudiendo ser percibido por los ultrasonidos en todo momento. En definitiva, al usar una red de sensores complementaria se evita la aparición de puntos muertos de percepción en el robot.

2.3 LOCALIZACIÓN Y SISTEMAS SENSORIALES:

2.3.1 Localización

Tal y como especifica la RAE, el verbo localizar significa “averiguar el lugar en que se halle alguien o algo”. Sin embargo, en robótica móvil, la localización es la disciplina que se encarga de plantear distintos modelos matemáticos o geométricos de entornos físicos para encontrar la posición y orientación de un robot móvil, con la ayuda de sensores y actuadores.

2.3.2 Sistema de localización a emplear

Como se explica en el capítulo I, este proyecto nace por la incapacidad de utilizar un GPS para localizar a un robot en interiores. El Sistema de Posicionamiento Global (GPS) utiliza un mínimo de tres satélites para obtener la posición de cualquier objeto con una precisión de centímetros.

Como alternativa a los Sistemas de Posicionamiento Global, nacen los Sistemas de Posicionamiento Local o LPS (Local Positioning System) que, como su nombre indica, están diseñados para funcionar en entornos locales. En particular, el Sistema de Posicionamiento Local empleado en este trabajo estará basado en el empleo de señales ultrasónicas y de radiofrecuencia.

Dentro de los LPS hay distintas maneras de estimar la posición de un objeto a localizar. Entre ellas podemos destacar aquellas basadas en la medida de rangos, en la medida de ángulos, en la medida de fuerza de señal... Sin embargo, en este proyecto se opta por utilizar un Sistema de Posicionamiento Local basado en la medida de rangos como es la trilateración.

Partiendo de esta base, se utilizarán tres balizas activas para localizar a un robot móvil de tracción diferencial.

2.3.3 Trilateración

Como se comenta en el apartado anterior, se contará con la presencia de tres balizas activas para poder localizar al robot por medio de la trilateración. La trilateración, como método de localización, estimará la posición del robot móvil utilizando las distancias entre las balizas y el propio robot. Al utilizar este método, resulta obvia la necesidad de disponer de un mínimo de tres balizas activas o dispositivos de emisiones ultrasónicas.

Por lo comentado anteriormente, antes de empezar con la utilización de las ecuaciones de la trilateración propiamente dichas, es necesario conocer las distancias entre los emisores y el objeto a posicionar, que en este caso es un robot. Puesto que no es posible determinar las distancias entre las balizas y el robot de una manera directa, se calcularán por medio de la única magnitud física observable que posee el sistema utilizado en este proyecto que es el tiempo de vuelo o ToA (Time of Arrival). Partiendo de esta magnitud, y tomando una serie de medidas, se podrá establecer una relación entre el tiempo de vuelo y la distancia baliza – robot. Por ello, la ecuación que relacionaría el tiempo de vuelo con la distancia entre la baliza y el robot, tendrá la siguiente forma:

$$z = h(x) + e$$

- z: Magnitud observable (ToA)
- h: Relación entre la posición del robot móvil y la magnitud observable
- x: Distancia robot - baliza
- e: Errores de medida

Una vez obtenidas las tres distancias entre los emisores de ultrasonidos y el robot, se procede a la aplicación de las fórmulas de la trilateración.

Existen dos tipos de trilateración: esférica e hiperbólica. Sin embargo, en este proyecto se va a utilizar exclusivamente la trilateración esférica debido a que el cálculo de las distancias se realizará por medio de los tiempos de vuelo absolutos y no por la diferencia de tiempos de vuelo como hace la trilateración hiperbólica. En consecuencia,

la trilateración partirá del desarrollo de las ecuaciones de la circunferencia puesto que la localización será en el plano bidimensional XY.

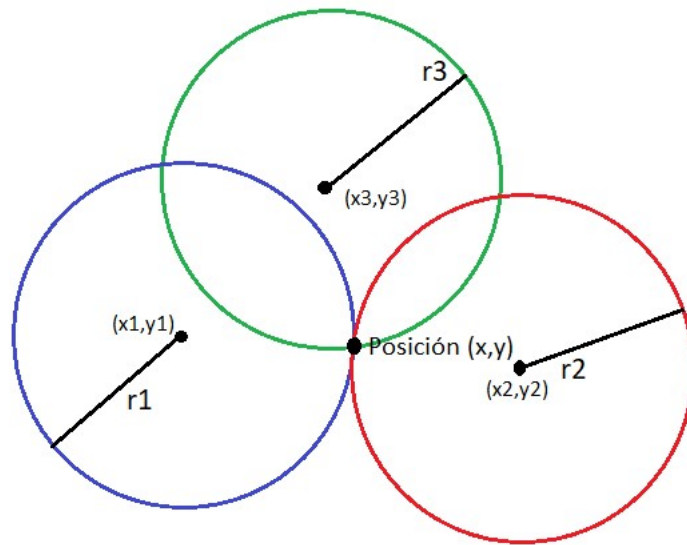


Imagen 5

Asumiendo que cada uno de los centros de las circunferencias mostradas en la imagen 5 representa a cada uno de los emisores, el punto de intersección de las tres circunferencias se correspondería con la posición en el plano del robot.

Se plantean las ecuaciones de las circunferencias:

$$1) r_1^2 = (x - x_1)^2 + (y - y_1)^2$$

$$2) r_2^2 = (x - x_2)^2 + (y - y_2)^2$$

$$3) r_3^2 = (x - x_3)^2 + (y - y_3)^2$$

Restando la ecuación 3 a la ecuación 2 y restándole a la ecuación 1, la ecuación 2, se plantea un sistema de ecuaciones con dos incógnitas de la siguiente forma:

$$Ax + By = C$$

$$Dx + Ey = F$$

- Siendo:

$$-A = -2x_1 + 2x_2$$

$$-B = -2y_1 + 2y_2$$

$$-C = r_1^2 - r_2^2 - x_1^2 - x_2^2 - y_1^2 - y_2^2$$

$$-D = -2x_2 + 2x_3$$

$$-E = -2y_2 + 2y_3$$

$$-F = r_2^2 - r_3^2 - x_2^2 - x_3^2 - y_2^2 - y_3^2$$

Por tanto, conociendo los valores de A, B, C, D, E y F, se calculan los valores de las coordenadas del robot en el plano x e y en función de estas variables conocidas quedando despejadas de la siguiente manera:

$$x = \frac{CE - BF}{AE - BD}$$

$$y = \frac{AF - CD}{AE - BD}$$

Operando estas dos ecuaciones obtendremos la posición x e y del robot, habiendo terminado con el proceso de posicionamiento del robot.

2.3.4 Determinación de la orientación

Para terminar de localizar al robot, es necesario conocer su orientación. Esta sería calculada de forma experimental. Sin embargo, puesto que no se ha avanzado en este aspecto, se repetiría el cálculo realizado por D. Agustín León García en el trabajo anterior.

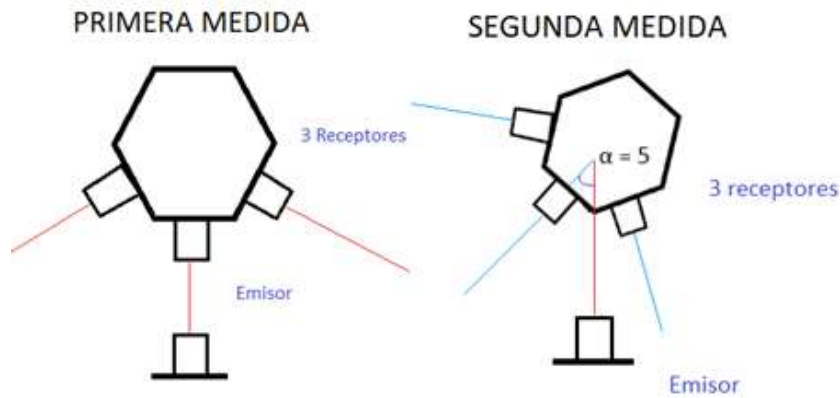


Imagen 6

Para el cálculo de la orientación, se utilizarán únicamente tres sensores de ultrasonidos en configuración receptora en el receptor y un emisor. En primer lugar, se colocarán los tres receptores tal y como se muestra en la parte izquierda de la imagen 6, de tal manera que el emisor forme un ángulo de cero grados con el receptor central. Con este montaje, se tomarán los valores de tiempo por interrupción empleados por cada receptor de ultrasonidos y serán asociados a un ángulo 0.

En segundo lugar, se girará en sentido horario (considerando este sentido positivo), el soporte de los sensores de ultrasonidos, de tal forma que todos estos giren un ángulo de 5 grados. Una vez se gire dicho ángulo, se anotarían los valores de tiempo por interrupción correspondientes a cada receptor. Se repetiría el procedimiento hasta girar un total de 50 grados, donde el emisor situado a la derecha del receptor central, quedará con un ángulo de 0 grados con el emisor.

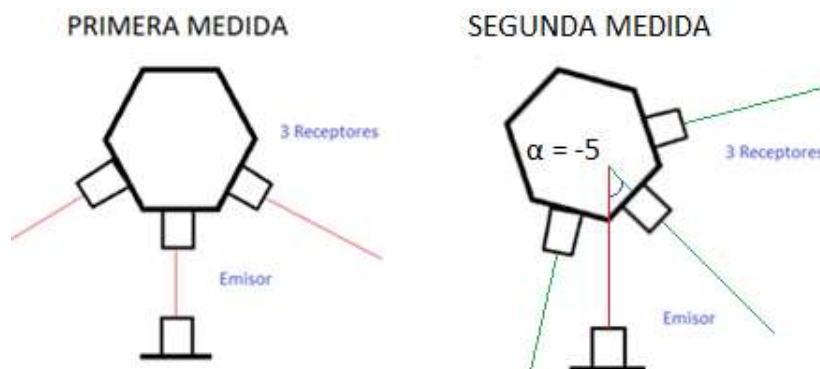


Imagen 7

Una vez se ha completado el giro horario de 50 grados, se procede a realizar el mismo procedimiento pero en sentido antihorario. Habiendo tomado datos de todos los receptores en ambos sentidos de giro se tendrían dos tablas: una con los valores registrados al girar en sentido horario, y otra con los valores con el sentido de giro contrario.

Con estos valores registrados se calculará la diferencia de tiempos registrados por los emisores laterales con el emisor central, quedando dos diferencias por cada medida, en lugar de tres datos por medida. Con estos valores de diferencias de tiempos y los ángulos asociados a estas diferencias, se harán un total de cuatro rectas a partir de las cuales se calculará el ángulo de orientación del robot en un rango de 100 grados con respecto al emisor.

CAPÍTULO 3: HARDWARE UTILIZADO Y HERRAMIENTAS SOFTWARE EMPLEADAS

3.1 ELEMENTOS HARDWARE

3.1 .1 Sensores de ultrasonidos:

Serán la base de funcionamiento de este sistema de localización y estarán implementados tanto en el hardware utilizado para la localización como en el propio robot a localizar.

Los ultrasonidos utilizados en el sistema de localización tendrán como única función emitir ondas de ultrasonidos cuando sea así indicado por el microcontrolador. Por otra parte, los ultrasonidos localizados en el robot tendrán como único objetivo recibir los pulsos emitidos por los emisores previamente nombrados.

Las ondas de ultrasonido son fundamentalmente iguales a las ondas sonoras que se pueden percibir por el sistema de audición humano. A diferencia de estas, las ondas del ultrasonido presentan una mayor frecuencia, transmitiendo su vibración a una velocidad aproximada de 340 m/s.

En el campo de la robótica, los sensores de ultrasonidos son utilizados fundamentalmente para medir distancias. Este tipo de sensores contienen un emisor y un receptor. Una vez el emisor emite una onda, se inicia una cuenta, se espera a que esa misma onda sea recibida por el mismo sensor y en función del tiempo que tarde en volver la onda tras toparse con algún objeto, se podrá calcular a la distancia a la que se encuentra a través de la fórmula que relaciona la velocidad y distancia:

$$d = v \cdot \frac{t}{2} \rightarrow d = 340 \frac{m}{s} \cdot \frac{t}{2}$$

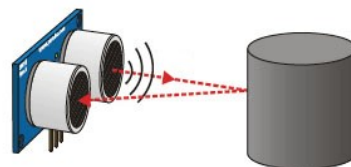


Imagen 8

El tiempo de espera desde el envío de la onda ultrasónica hasta su recepción se suele denominar tiempo de eco o tiempo de conmutación. Este sistema de cálculo de distancias es fundamentalmente utilizado para distancias pequeñas, debido a que a mayores distancias, el error incrementa. Los sensores utilizados en este trabajo poseen una distancia máxima de detección de 4 m teóricamente.

Dependiendo del tipo de sensor, existirá una limitación de distancia mínima detectada, a partir de la cual el sensor no podría detectarla. Este fenómeno se produce porque los ultrasonidos, tras enviar un pulso, necesitan un tiempo de espera para estabilizarse y posteriormente recibir el envío. Este tiempo mínimo de interrupción es traducido a una distancia mínima de detección.

Aparte de esta última limitación mencionada anteriormente, los sensores de ultrasonidos presentan una serie de inconvenientes. Como se muestra en la imagen 3, el área de detección del sensor posee una forma cónica, por lo que solo se podrán detectar objetos dentro de la misma. El ángulo con el que la onda incida en un obstáculo será el mismo que el ángulo de reflexión de la misma. Por lo tanto, si se supera un cierto ángulo crítico de incidencia, la onda reflejada podría no ser detectada.

También podría ocurrir que la onda, en vez de regresar directamente al sensor tras haberse topado con un obstáculo, tuviera múltiples reflexiones antes de llegar al mismo. Esto haría que el tiempo de conmutación fuera mayor y provocaría una medida de distancia errónea. Pese a todos estos posibles inconvenientes, estos sensores serán utilizados debido a su tiempo de viaje ya que permite utilizar electrónica convencional para realizar medidas del tiempo de vuelo

Por este tipo de limitaciones se decide montar ultrasonidos tanto en los emisores como en los receptores colocados en el robot de tal manera que no haga falta el empleo de la onda de rebote del ultrasonido.

3.1.1.1 Pineado del sensor de ultrasonidos.



Imagen 9

En particular, se emplearán para este proyecto los ultrasonidos HC-SR04 como el que se muestra en la imagen 9. Cada uno de estos sensores, consta de un módulo transmisor señalizado con una T en el lado izquierdo del componente y un módulo receptor señalizado con una R en lado derecho del mismo. Por otra parte, también se compone de cuatro pines que serán descritos a continuación:

- Vcc: Pin de alimentación del componente.
- Trig: Pin de disparo (trigger) del ultrasonido. Emitirá un pulso de 5 voltios cuando indique el microcontrolador que lo maneja,
- Echo: Pin de recepción de ondas que se mantendrá a 5 voltios cuando reciba un pulso de ultrasonidos el tiempo que tarde en recibir la onda que el mismo sensor ha emitido. Como únicamente se utilizará en los sensores empleados como receptores, veremos una señal conformada por picos continuados de 5 voltios.
- Gnd: Pin de tierra del componente.

3.1.2 Módulos de radio NRF24L01:

En este proyecto serán utilizados cuatro módulos de radio (uno por cada emisor y uno en el receptor) que se encargarán de coordinar la emisión de ultrasonidos así como de

ordenar los pulsos recibidos por el robot, categorizando en cada caso la procedencia de los pulsos distinguiendo si ha sido recibido por el emisor A, B o C en cada caso.

Estos módulos de radio son módulos transceptores, lo que quiere decir que serán capaces de emitir información y de recibirla. Funcionan con una tensión de alimentación de 3.3V fácilmente obtenible en microcontroladores como la placa Arduino Uno o la placa Arduino MEGA que serán utilizadas en este proyecto. Para la comunicación, emplean el protocolo de comunicación SPI.



Imagen 10

Estos módulos de radio son muy utilizados en proyectos de Arduino debido a que el control por bus SPI es sencillo de realizar con los microcontroladores de Arduino. Además, serán utilizados en este proyecto principalmente por su bajo precio y la aportación de una comunicación robusta entre los emisores y en las comunicaciones emisor-receptor sin la necesidad de implementar un código muy extenso o complicado.

3.1.3 ARDUINO

3.1.3.1 Placa Arduino UNO:

Esta placa es la más común de la marca Arduino, llevando ya nueve años en el mercado, existen numerosos proyectos que la utilizan. Contiene un microcontrolador llamado ATmega328 perteneciente a la empresa AVR que tiene una memoria de 32 KB, una RAM de 2KB y una EEPROM de 1 KB. Así como otras placas posteriores más potentes, esta posee un reloj u oscilador de 16 MHz de frecuencia.

Para la realización del proyecto que se defiende, serán utilizadas tres placas Arduino UNO R3, una asociada a cada emisor. Estas placas tendrán la función de controlar tanto los sensores de ultrasonidos encargados de emitir los pulsos de ultrasonidos, así como

los módulos de radio encargados de la emisión y la recepción de los mensajes A, B o C en cada caso.



Imagen 11

En la imagen 11 se muestra la placa Arduino Uno R3 que será la utilizada en este proyecto. En cada baliza ultrasónica, se emplearán un total de 11 pines.

Los pines 5 y 6 serán empleados como pines Trigger y Echo respectivamente. Como se explica anteriormente, estos pines corresponden al sensor de ultrasonidos.

Los pines 9, 10, 11, 12 y 13 de la placa serán utilizados por el módulo de radio NRF24L01. El pin 9 irá conectado al pin 3 del módulo de radio (CE) así como el pin 10 irá conectado al pin CSN del módulo NRF24L01. Los pines 11, 12 y 13 están conectados a los pines SCK, MISO y MOSI respectivamente.

Por último, se utilizarán los pines de alimentación (POWER) para suministrar 3.3 V al módulo de radio y 5 V al módulo emisor de ultrasonidos.

3.1.3.2 Placa Arduino MEGA 2560:

La placa Arduino MEGA pertenece a la marca Arduino y es una versión más potente y con más pines que la placa Arduino UNO. Posee una tensión de alimentación de 5 V y un límite de entrada de entre 6-20 V. Consta de un total de 54 pines digitales entre los cuales 14 serán PWM. Además, contiene 16 pines analógicos, pero estos no serán utilizados en este proyecto.

Esta placa será alimentada a través de la entrada USB que posee por la que también serán cargados los programas. Además, cuenta con una entrada Jack de 2.5 mm con la que también podrá ser alimentada.

En cuanto a la memoria de la placa debemos decir que contiene una memoria flash de 256 KB, una memoria SRAM de 8 KB y una EEPROM de 4 KB. Otro dato de interés puede ser la frecuencia de reloj de este dispositivo que es de 16 MHz.

En este proyecto será incluida una placa Arduino Mega 2560 en el robot. Se encargará de controlar el módulo receptor de radio perteneciente al robot y los seis sensores de ultrasonidos.



Imagen 12

Se empleará esta placa principalmente porque tiene seis pines de interrupción incluidos que serán utilizados por los seis ultrasonidos situados en el robot móvil. Estos pines serán los pines 2, 3, 18, 19, 20 y 21 e irán conectados a los pines echo de los distintos ultrasonidos utilizados en el diseño de este proyecto.

Se utilizará el pin de alimentación de 5 V junto con uno de los pines de tierra para suministrar el voltaje a todos los ultrasonidos del robot. Pese a solo haber un pin de 5 V, se ramificará esa salida para que haya tanto 6 cables de 5V como 6 cables de tierra (GND).

Por otro lado, la alimentación del módulo receptor de radio del robot se realizará por medio del pin de 3.3 V situado en la esquina inferior izquierda de la imagen 12 que muestra los pines de alimentación (POWER).

Para terminar de conectar el módulo de radio serán escogidos los pines 9 y 10 correspondientes a los pines PWM para unirlos a los pines CE y CSN respectivamente. Por último, se utilizarán los puertos digitales 52, 48 y 49 para los pines SCK, MOSI y MISO.

3.2 HERRAMIENTAS SOFTWARE EMPLEADAS

3.2.1. ARDUINO

El proyecto o la marca Arduino nace en 2005 como un proyecto destinado a estudiantes que utilizaban un microcontrolador denominado BASIC Stamp con un precio de 100 dólares. Siendo este un precio elevado, se decide realizar un proyecto en el que se creen unas herramientas simples y económicas para crear proyectos digitales. Con esta idea y basándose en el proyecto Wiring, Massimo Banzi y David Mellis crean Arduino.

3.2.1.1 ARDUINO IDE

El programa Arduino IDE (Integrated development environment) es una aplicación disponible para Windows MacOs y Linux escrito en Java. Se utiliza normalmente para cargar los programas realizados en placas compatibles con esta aplicación. Consta de dos funciones básicas que serán empleadas en todos los programas pero además se podrán incluir otras funciones. Las dos funciones básicas serán la void setup() donde se especificarán el modo de los pines de las placas a utilizar entre otras especificaciones y se pondrán las instrucciones que solo se quieran realizar una vez. Por otro lado, en la función void loop() se escribirán aquellas instrucciones que se quieran repetir en bucle.

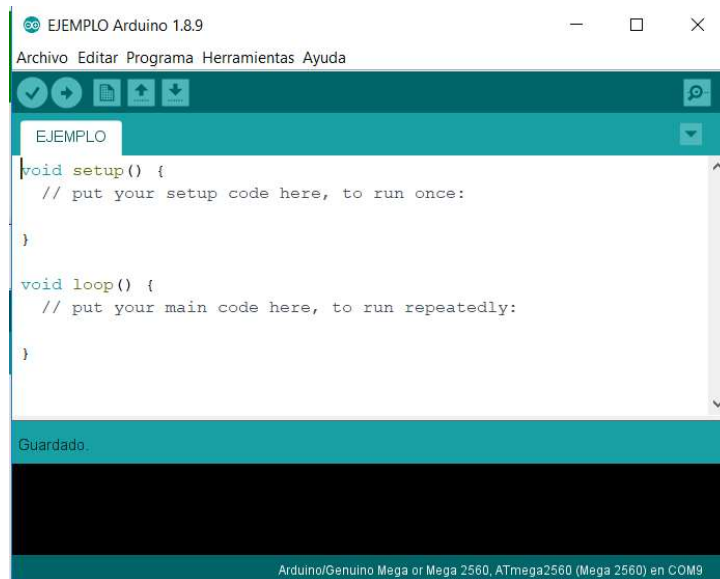


Imagen 13

En la imagen 13, se muestra un ejemplo de un programa vacío de Arduino en el que se pueden apreciar las dos funciones principales que componen: `void setup()` y `void loop()`. Además, en la esquina inferior derecha de la ventana del programa se mostrará en que placa de la familia Arduino ha sido cargado el programa y en que puerto del ordenador se halla conectada esa placa.

Los programas creados en Arduino IDE serán cargados en las placas compatibles con Arduino por medio de una comunicación serial, que comunicará la placa utilizada con el ordenador en el que haya sido creado el programa.

3.2.2 AUTODESK INVENTOR 2020

Autodesk Inventor es un programa de diseño de objetos sólidos en 3D muy usado en el ámbito de la ingeniería y la tecnología. Es un software de CAD perteneciente a la empresa Autodesk.

Mientras se cambiaba el hardware de algunos emisores, los antiguos soportes impresos en 3D se rompieron y por ello, se trató de encontrar los diseños realizados para volver a imprimirlos. Puesto que estos no se encontraron, se decidió realizar un nuevo diseño tanto para los soportes empleados por los emisores como para el soporte que iría encima del robot.

Todos estos diseños han sido realizados a través del software mencionado sin excesiva dificultad para posteriormente ser imprimidos en 3D.

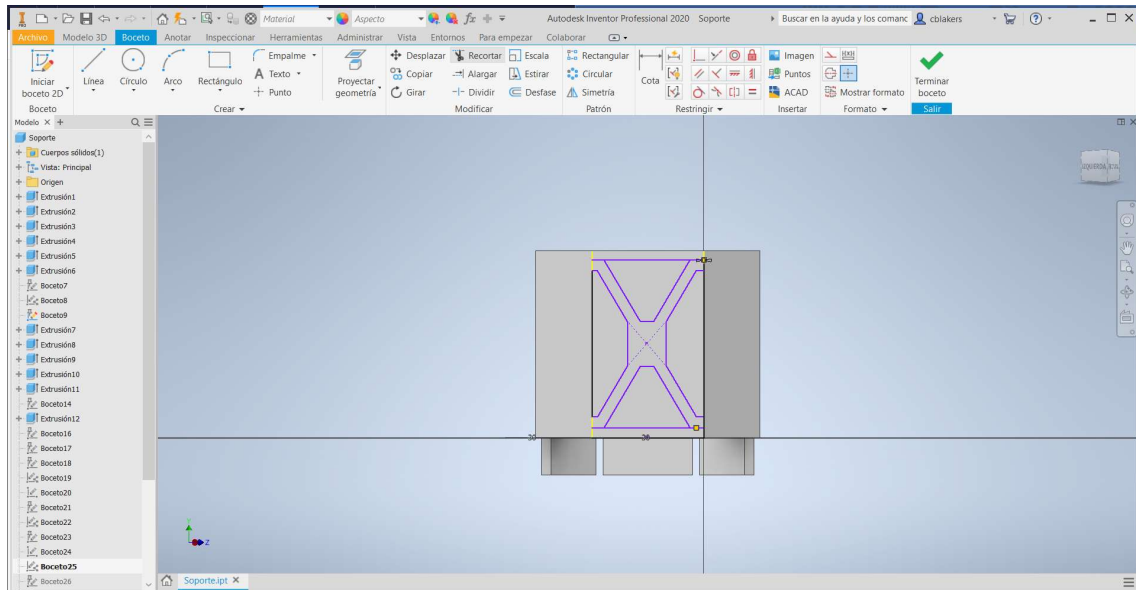


Imagen 14

En la imagen 14 se muestra la interfaz del programa Autodesk Inventor 2020, en la cual podemos ver la pestaña de edición de un boceto. Se trata de una de las piezas que se fabrican en este proyecto. En particular, corresponde al soporte donde irán colocados los ultrasonidos receptores pertenecientes al robot móvil.

Debemos indicar que se ha utilizado la licencia de estudiante para utilizar este programa como se muestra en la imagen que sigue (imagen 15):



Imagen 15

3.2.3 REPETIER HOST

En el trabajo de fin de grado que se describe desarrolla en esta memoria, se utiliza el software Repetier Host. Se trata de un software cuya función es la configuración de archivos para que estos puedan ser impresos en 3D. Es necesario generar un archivo de extensión “.gcode” que contenga las especificaciones necesarias para imprimir un archivo de pieza diseñada en 3D, generalmente de extensión “.stl.”

Por medio de la utilización de este software, también será posible ajustar la velocidad de impresión a tiempo real, subir o bajar la temperatura de fundición del filamento utilizado en la impresión y ajustar la ventilación de la impresora en función de lo que necesite.

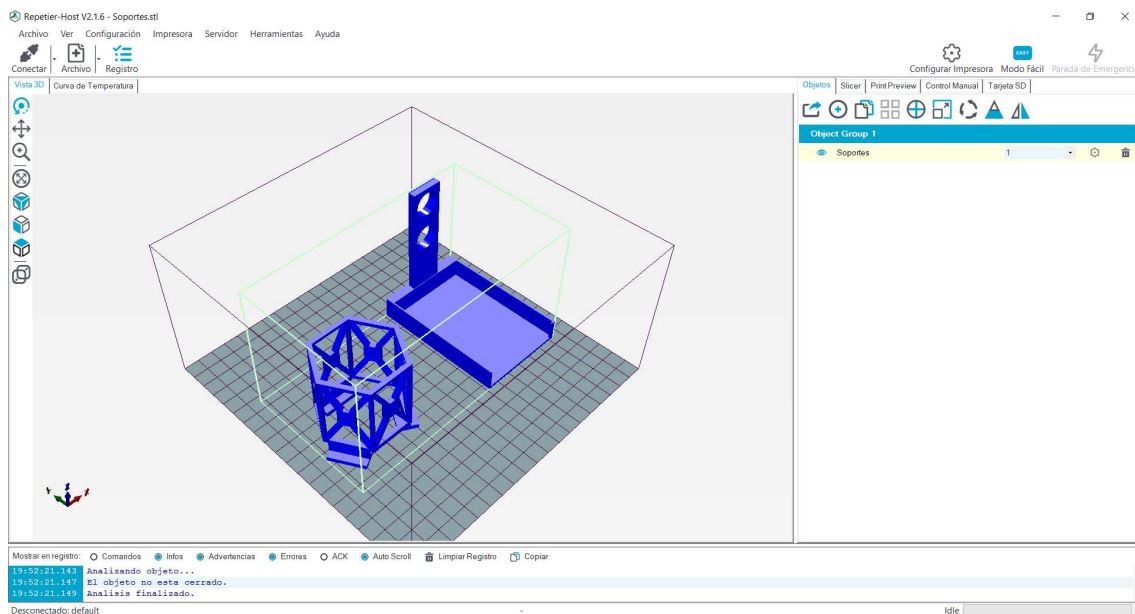


Imagen 16

En la imagen 16, se observa la interfaz del programa Repetier host, en la cual aparecen representadas dos de las cuatro piezas a imprimir en este proyecto dispuestas sobre un plano representativo de la mesa calefactora de la impresora. La disposición de las piezas que se haga en este programa será la disposición en la cual la impresora imprima las mismas.

3.2.4 SLIC3R

Este programa o software, está integrado en el Repetier Host, explicado anteriormente. Su función es incluso más importante que la del propio Repetier puesto que SLIC3R es el encargado de cortar la pieza en rodajas horizontales en de tal manera que la impresión se vaya realizando capa a capa. Este mismo programa será el encargado de la generación de las trayectorias a recorrer por la impresora a lo largo de la mesa calefactora. Además, dispondrá de otra función muy importante de cara a la impresión que será el cálculo del material a extruir que, en conjunto con las trayectorias calculadas para la impresión, proporcionará al usuario el tiempo restante de impresión en cada momento.

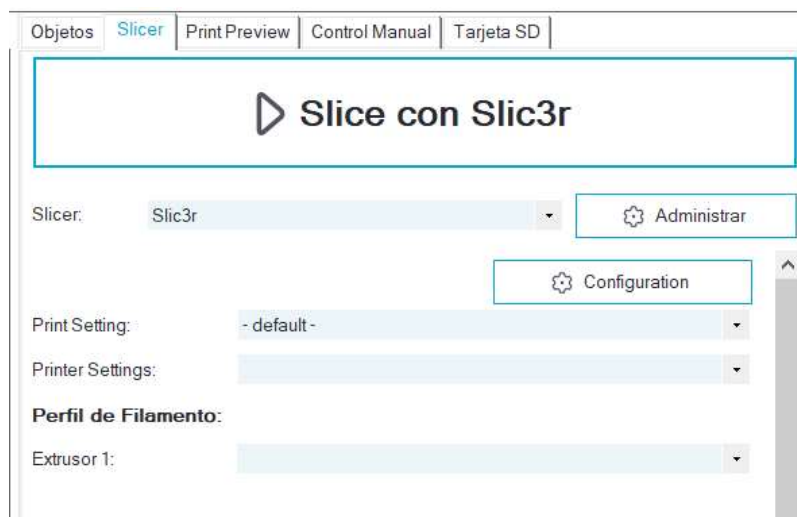


Imagen 17

La imagen 17 muestra un fragmento de pantalla de la interfaz del programa Repetier Host. Esta pestaña muestra la configuración del slicer a utilizar, el tipo de slicer y las configuraciones de impresión. En la ventana aparecen las configuraciones por defecto. Sin embargo, la configuración utilizada no será esa ya que se necesita un espesor de relleno específico en las piezas realizadas en este proyecto para que sean resistentes.

CAPÍTULO 4: PROBLEMAS ENCONTRADOS, SOLUCIONES PLANTEADAS Y MEJORAS EN EL PROYECTO.

4.1 CAMBIOS REALIZADOS EN EL HARDWARE DEL PROYECTO ANTERIOR

En el comienzo de este proyecto, en el que se partía de un trabajo anterior, el hardware estaba compuesto por nueve transceptores de ultrasonidos HC-SR04, cuatro módulos de radio NRF4L01, tres placas Arduino UNO, una placa Arduino MEGA 2560 y un set de Dexter Industries Gopigo, que es un robot en configuración diferencial (dos ruedas y un pivote o bola) con una placa raspberry pi integrada.

Además, había un total de 5 piezas impresas en 3D que son las siguientes:

- 3 soportes para las balizas ultrasónicas.
- 1 soporte para los seis ultrasonidos colocados en el robot móvil.
- 1 caja para la placa Arduino MEGA 2560 dispuesta en el robot móvil.

En primer lugar, se realiza un chequeo de los sensores de ultrasonidos, que conllevará los primeros cambios en el hardware. Este chequeo se realiza puesto que tras probar en varias ocasiones los códigos llevados a cabo por D. Agustín León García, no se consiguen resultados lógicos de medición de distancias entre baliza y robot. Se verifica el correcto funcionamiento de los sensores de ultrasonidos tomando un osciloscopio como referencia.

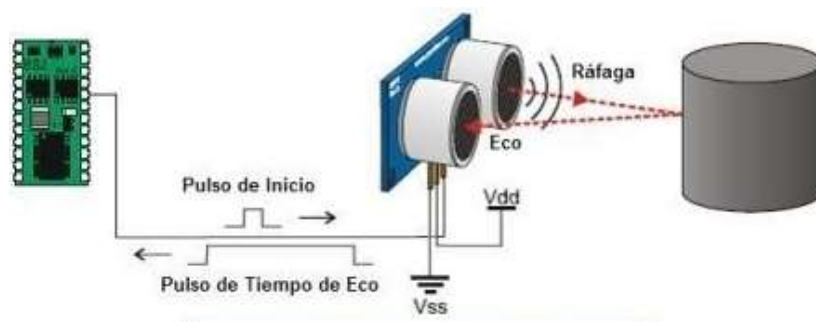


Imagen 18

Para poder comprobar el buen estado de los sensores, estos fueron extraídos de la estructura 3D diseñada por Agustín León García, situada encima del montaje gopigo de Dexter Industries. Fue necesario despegarlos debido a que al estar adheridos a la

estructura 3D era complicado acceder al pin echo, necesario para comprobar que se reciben bien los pulsos enviados por las balizas ultrasónicas.

Una vez desmontados los ultrasonidos receptores, se procede a evaluarlos individualmente con la ayuda de un osciloscopio, en el cual se comparará la señal de disparo de uno de los emisores y la señal del pin echo de cada ultrasonido. De esta manera, cada vez que se reciba un pulso digital de 5V en el canal 1, tanto la imagen del canal 1 como la del canal 2 permanecerán en la pantalla hasta que se reciba un nuevo pulso.

En la comprobación de estos ultrasonidos, fueron retirados dos ultrasonidos debido a que no mostraban respuesta al recibir señales.

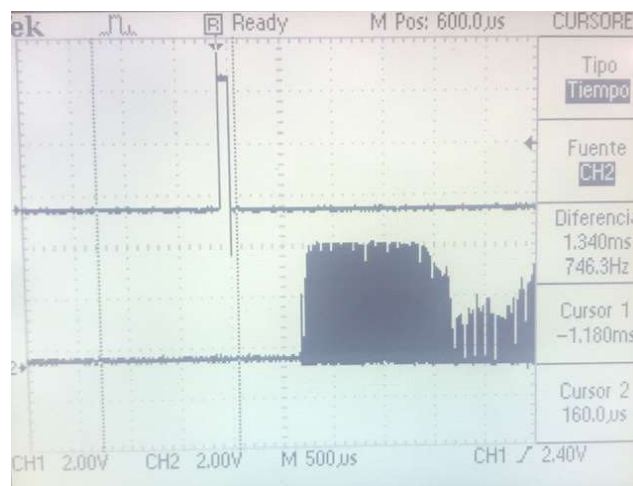


Imagen 19

Como se puede ver en la imagen 19, captada por el osciloscopio, el canal 1 se ha conectado al pin de emisión (Trigger) del emisor A que emite un pulso de cinco voltios cada ciclo de emisión. El canal 2 se ha conectado al pin echo de uno de los ultrasonidos de tal manera que se muestre la señal por el mismo. Tal y como está construido este ultrasonido y como indica su datasheet, el pin echo debería mantenerse a 5 voltios un tiempo equivalente al tiempo que tarde en ir y volver la onda emitida. Sin embargo, como el ultrasonido conectado a la placa Arduino MEGA no emite ningún pulso, sino que el pulso recibido proviene de una de las balizas, obtenemos este tipo de señal (canal 2 del osciloscopio) en la cual el pin echo se pone a 5V durante breves periodos de tiempo mientras va disminuyendo la señal.

Se tomará como referencia el primer pico de 5V para determinar el tiempo de conmutación de nuestro grupo emisor-receptor. Para verificar que estos sensores funcionan correctamente, se irá aumentando la distancia entre el sensor de ultrasonidos en configuración emisora y el sensor de ultrasonidos en configuración receptora y se verifica si el tiempo de conmutación aumenta proporcionalmente.

En segundo lugar, a medida que se van haciendo pruebas con distintos códigos, se realiza un nuevo chequeo de los ultrasonidos puesto que, en ocasiones, no se obtienen medidas de los tiempos de interrupción asociados a algunos sensores de ultrasonidos. Para determinar el porqué de este fenómeno se acude al datasheet de este sensor HC-SR04.

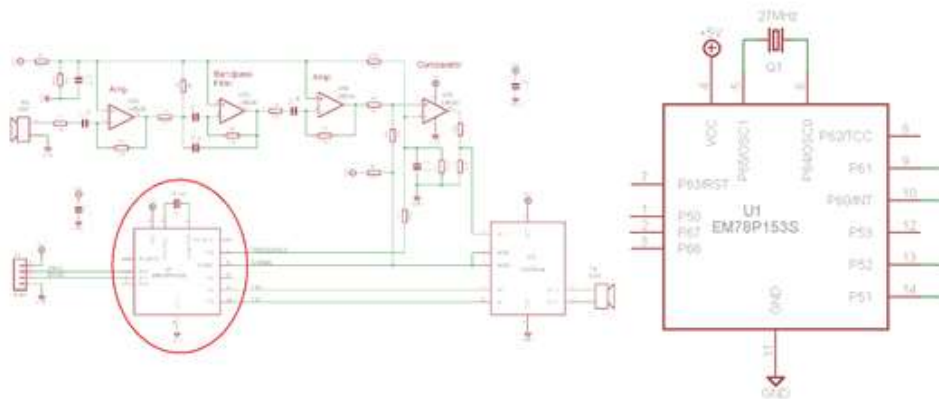


Imagen 20

En el diseño esquemático del sensor presente en el datasheet del sensor HC-SR04 se observa que hay un microcontrolador que establece un umbral que no permite que se dispare el pin echo del sensor a menos que se supere ese umbral establecido. Por ello, se decide quitar este componente del sensor de ultrasonidos, para que la señal echo no esté controlada por una fuente externa y pueda ser recibida en todo momento.

Por lo tanto, todos los ultrasonidos empleados en el robot, han sufrido esta modificación, actuando ahora como transceptores pasivos que únicamente amplificarían la señal ultrasónica recibida.



Imagen 21

En las fotos mostradas (imagen 21), se aprecia la modificación descrita anteriormente. A partir de este momento, la señal echo será extraída directamente por el pin anteriormente ocupado por el microcontrolador EM78P153S.

Como última modificación realizada en el hardware de este proyecto, se han añadido condensadores de 33 uF entre los pines Vcc y GND de los sensores de ultrasonidos y de los módulos de radiofrecuencia. De esta forma, se filtrarán posibles ruidos a la entrada de estos componentes.

4.2 CAMBIOS REALIZADOS EN EL SOFTWARE DEL PROYECTO

4.2.1 Problemas de comunicación

En cuanto a la revisión ejecutada en el software del proyecto, se comienza ejecutando los códigos previos para de esa manera comprobar la correcta localización del robot. Sin embargo, al probar el código del receptor, se comprueba que no se comunican bien los emisores con el receptor y que no se obtiene ninguna información por el monitor serie.

Esto es un problema grave dado que no se puede verificar si el receptor está recibiendo los mensajes "A", "B" y "C" en orden y, en consecuencia, si está funcionando correctamente el ciclo de emisión de ultrasonidos en el cual se basa este sistema de localización en tiempo real.

Por ello, la primera solución y cambio que se realiza en este proyecto es la sincronización de los tres emisores de ultrasonidos con el receptor y la puesta en marcha del ciclo de emisión. Para tal fin, se analizan los códigos de los emisores en comparación con el código del receptor y se comprueba que la velocidad de transmisión del receptor está a 57600 bps mientras que la de los emisores está a 9600 bps; esta diferencia inhabilitaba la comunicación entre ambos puesto que la

comunicación es serial asíncrona en estos módulos de radio y es necesario fijar la velocidad ya que no se usa el reloj como referencia.

Una vez realizados los distintos cambios en los códigos, se consigue establecer comunicación entre los emisores y el receptor, indicando este último cuando se reciben pulsos, especificando además qué mensaje se recibe en el receptor si bien es A, B o C.

4.2.2 Medición de distancias

Una vez solucionado el problema descrito anteriormente, se procede a comprobar el correcto funcionamiento de la medición de distancias. Este proceso de medición de distancias es realizado en el código implementado en la placa Arduino MEGA situada en el receptor del robot.

El código implementado es un código basado en interrupciones de tal manera que, cada vez que se reciba un pulso digital en cualquiera de los seis pines de interrupción de la placa Arduino MEGA, habrá una función que recogerá el tiempo que se ha tardado en recibir esa interrupción desde que se recibe la señal de radio. Con este tiempo, se calcula la distancia desde el emisor del que provenga la señal de radio hasta el receptor.

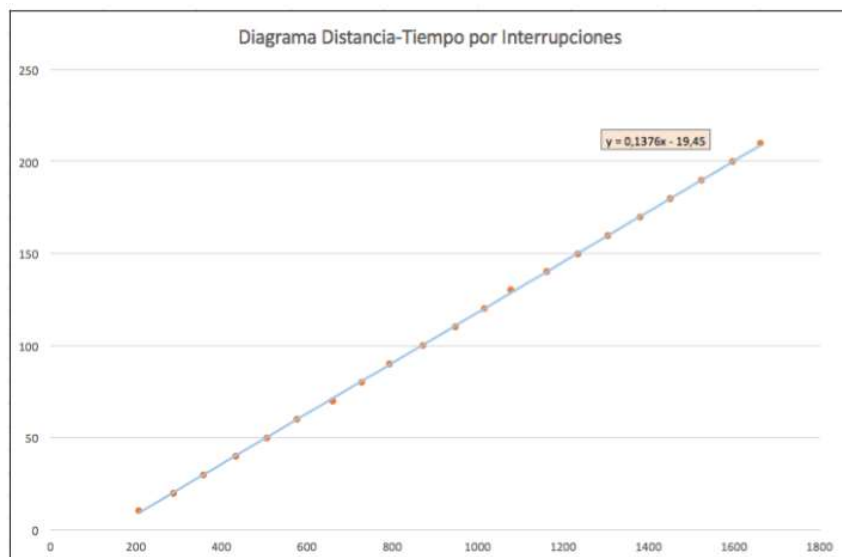


Ilustración 1

Siguiendo la recta mostrada en la ilustración 1 que calcula D. Agustín León García en el trabajo de fin de grado que se pretende continuar, los datos obtenidos por el receptor en tiempo por interrupciones dan valores inexactos con respecto a la distancia real de

referencia entre los emisores y el receptor. Lo que se obtiene en el monitor serie de la placa Arduino MEGA son las distancias entre los emisores y los seis ultrasonidos del receptor, quedando un mensaje de la siguiente forma:

Interrupción: (nº de interrupción: 1-6)

Distancia al emisor X: (distancia) cm

La placa Arduino Mega consta de seis pines de interrupción tal y como se comenta en el apartado 3.1.3.2 de esta memoria por lo que habrá seis interrupciones presentes en el código del receptor. Como se describe anteriormente, cada vez que se activa una de estas interrupciones se calcula la distancia entre el emisor del que se haya recibido el mensaje y el receptor y estas medidas, sin haberse movido el robot, variaban en todo momento lo cual hacía imposible una localización precisa del robot.

Careciendo el sistema propuesto tanto de exactitud como de precisión, es necesario rediseñar el software del sistema por lo que se comienzan a programar nuevos códigos tanto para los emisores como para el receptor.

4.2.2.1 Código 1 emisor - 1 receptor

Antes de la ejecución de este código se debe organizar el hardware de la siguiente manera:

- Pin 2 del emisor A conectado al pin 21 del receptor.
- 1 sensor de ultrasonidos conectado al pin 3 de la placa Arduino MEGA 2560.
- Emisores B y C desconectados.

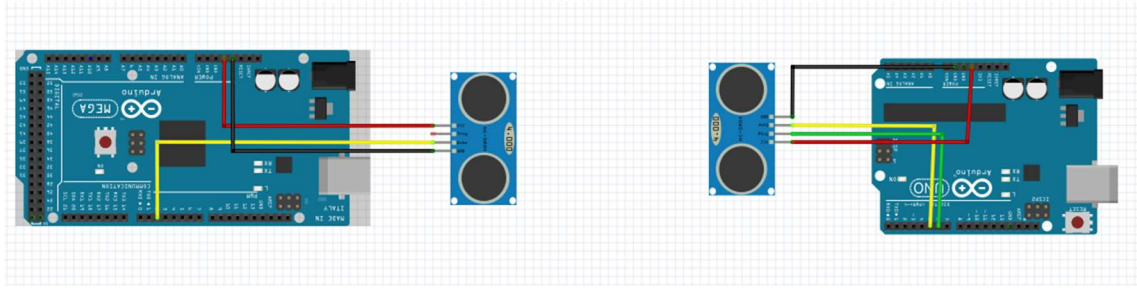


Imagen 22: (Se ha ignorado la presencia de los módulos de radio nrf24l01 por mejorar la comprensión de los montajes)

Una vez preparado el hardware, se carga en la placa Arduino Mega un código que sólo registrará las interrupciones que sean producidas por un pulso emitido por uno de los emisores, en este caso será el emisor A.

Al probar este código se observa que hay determinados valores de los tiempos de interrupción que no siguen una proporcionalidad en relación a la distancia a la que se encuentran del emisor por lo que se decide hacer un filtrado de estas interrupciones. En consecuencia, únicamente serán tenidas en cuenta aquellas interrupciones que hubieran sido producidas dentro de un periodo de 25 ms (estos 25 ms hacen referencia a la frecuencia de emisión de ultrasonidos que son 40 kHz). Para ello, se calculará la diferencia entre cada interrupción y la siguiente dejando un margen de 4 ms, teniendo en cuenta únicamente aquellas interrupciones que hubieran sido producidas entre 24 ms y 27 ms de periodo.

```

for (int j=0; j < NUM_INT-1; j++)
  difs[j] = tiempoInt1[j+1] - tiempoInt1[j];

```

En esta parte del código perteneciente al receptor, se hallarán las diferencias entre los valores de tiempo por interrupción. En primer lugar, se definirá una variable NUM_INT que valdrá 9 para sacar por pantalla y analizar un total de 9 interrupciones producidas en un mismo pin.

```
int oks[NUM_INT];
int primerOk = -1;
for (int j=0; j < NUM_INT -1; j++) {
    oks[j] = 0;
    if ((difs[j] < 27) && (difs[j] > 23)) {
        oks[j] = 1;
        if (primerOk == -1)
            primerOk = j;
    }
}
```

Si las diferencias de tiempo no entran dentro del margen de entre 23 y 27ms, serán descartadas y no serán analizadas por nuestro código.

Tras realizarse este filtrado, se habrá conseguido una proporcionalidad entre los tiempos de interrupción y la distancia entre el dispositivo emisor y el receptor con lo cual se podrá obtener una relación entre los tiempos de interrupción y la distancia al receptor.

4.2.2.2 Código 3 emisores - 1 receptor

Disposición del hardware:

- Pin 2 del emisor A conectado al pin 21 del receptor (pin de interrupción).
- 1 sensor de ultrasonidos conectado al pin 3 de la placa Arduino MEGA 2560.
- Emisores B y C conectados.

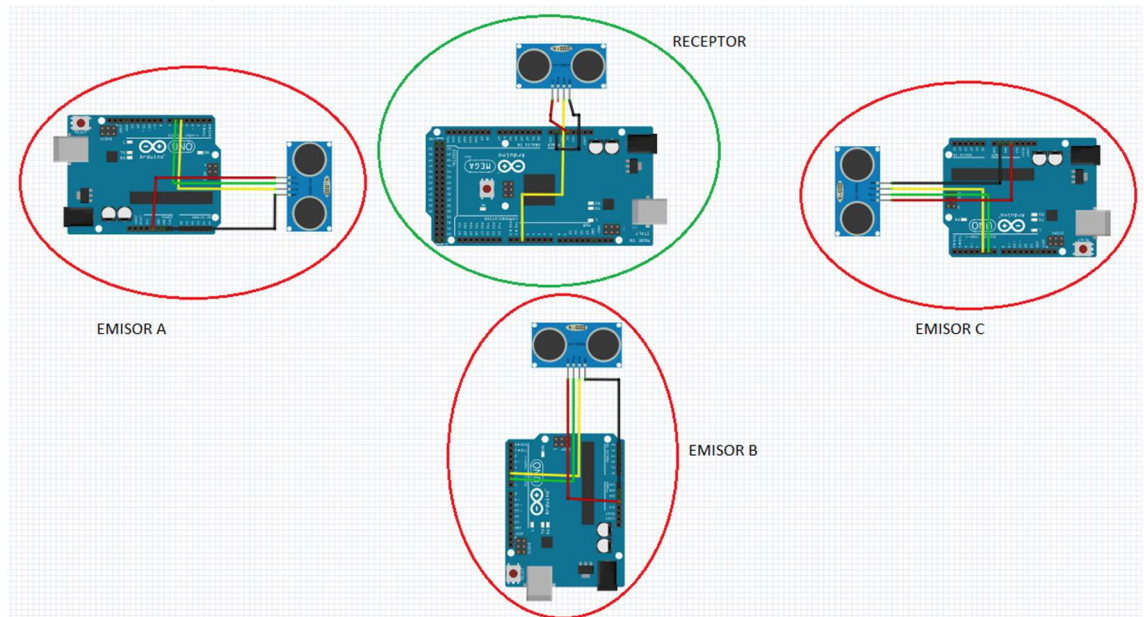


Imagen 23: (Se ha ignorado la presencia de los módulos de radio nrf24l01 por mejorar la comprensión de los montajes)

Con este código se intenta comprobar que la proporcionalidad anteriormente conseguida se mantenga, atendiendo solo a los pulsos emitidos por el emisor A pese a las posibles perturbaciones causadas por los emisores B y C no utilizados previamente. Sin embargo, estas perturbaciones son bastante influyentes puesto que, pese a seguir recibiendo los pulsos adecuadamente, los valores de los tiempos de interrupción varían aleatoriamente haciendo inservibles estos resultados obtenidos.

Puesto que con el código descrito previamente no fue posible obtener una linealidad entre la distancia y los tiempos por interrupción, se intentó estabilizar los valores de tiempos de interrupción para conseguirlo. Para ello, se propone esperar 1 milisegundo justo después de haber emitido un pulso a través de cable al pin de prueba (pin 21 de la placa Arduino Mega 2560) para que tenga ese tiempo para ser procesado y posteriormente enviar el pulso a través del ultrasonido.

Tras hacer esta modificación, se obtiene bastante estabilidad en los resultados. Sin embargo, se siguen produciendo diferencias entre medidas de hasta 10000 unidades de tiempo, inaceptables en un sistema de localización.

Se adjunta una tabla con los resultados obtenidos al haber probado los nuevos códigos en la que se mostrarán los valores de tiempo por interrupción asociados a cada emisor en el Anexo 1.

En esta tabla se observa que en los tiempos por interrupciones asociadas al emisor A, hay 6 medidas erróneas de 33, lo que supone un 18% de las medidas.

En los tiempos por interrupciones asociadas al emisor B se producen 15 medidas erróneas de 29 medidas tomadas, lo que supone un 52% de las medidas.

Por último, en el emisor C se obtienen 16 medidas erróneas de 30 medidas tomadas, lo que equivale al 53% de las medidas.

El número total de medidas tomadas han sido 33. Sin embargo, hay medidas en las que no se ha captado información por parte del ultrasonido y es por ello que faltan valores provenientes tanto del emisor B como del emisor C. Con estos resultados, resulta imposible tener un sistema de localización fiable puesto que más de la mitad de las medidas tomadas son erróneas.

4.2.2.3 2º Código 3 emisores - 1 receptor

En esta nueva configuración el emisor A se encargará de enviar dos pulsos: uno a través del ultrasonido como normalmente venía haciendo hasta ahora; y otro pulso que será enviado por cable a través del pin 2 del emisor A. Este pulso nos servirá de referencia para comparar el tiempo que se tarda en recibir este pulso enviado por cable, con lo que tarda en llegar a través de ultrasonidos y en comparación con el tiempo en el que se activa la radio.

Por lo tanto, habrá un total de tres cuentas en el código:

Cuenta 1: Empezará la cuenta cuando llegue la señal del pulso digital y empezará a contar en caso de que se lea el mensaje A en el receptor. En caso de leerse el mensaje B o el mensaje C, esta cuenta comenzará cuando se compruebe que la señal de radio está disponible.

```
if (digitalRead(pinMedidaEmisorOn) == 1){
  // Es el A y ha recibido la señal física encendida por el emisor A
  cuenta1 = micros(); // cuenta 1 es cuando se emite la señal inicial sincronizada
```

Imagen 24: Cuenta 1 para emisor A

```
} else { // No se ha encendido pin de prueba es B o C
  // Es B o C y hemos recibido la señal inalámbrica
  cuenta1 = micros();
```

Imagen 25: Cuenta 1 para emisores B y C

Cuenta 2: Comenzará a contar en el momento en que se detecte la señal de radio más un tiempo de 10 ms que es añadido en el emisor A justo después de enviar el pulso digital a través del pin de prueba por medio de un cable.

```
while (!radio.available());
cuenta2 = micros();
```

Imagen 26: Cuenta 2

```
delayMicroseconds(100);
digitalWrite(pin_prueba, HIGH);
delayMicroseconds(100);
digitalWrite(pin_prueba, LOW);

delay(10);

radio.stopListening();
delayMicroseconds(10);
radio.write( &dataToSend, 1 );
```

Imagen 27: Fragmento de código en el que se activa el pin de prueba (PinMedidaEmisorOn en el receptor) y donde se produce el delay de 10 ms antes de mandar el mensaje de radio

Cuenta 3: Guardará el valor de tiempo que se haya tardado en recibir el pulso emitido por el emisor de ultrasonidos siempre que el tiempo que se haya tardado en recibir ese tiempo no sea mayor que el tiempo de reinicio fijado.

```
while ((!digitalRead(pinEntrada1) == 1)&& (micros() - TiempoInicial < TIEMPO_REINICIO)) ;
cuenta3 = micros();
```

Imagen 28

Con estas tres cuentas, obtendremos tres tiempos que serán fundamentales para el estudio del funcionamiento de los sensores de ultrasonidos y sus tiempos.

- TUltraWifi: Se calculará como la diferencia entre la cuenta 3 y la cuenta 2, con lo que hallará el tiempo entre haber recibido el pulso emitido por el sensor de ultrasonidos y haber recibido la señal de radio.
- TUltraPin: Es calculado como la diferencia entre el tiempo tardado en recibir el pulso ultrasónico y el tiempo tardado en recibir el pulso que ha sido emitido por cable a través del pin de prueba. Este tiempo será comparado con la señal obtenida en el osciloscopio.

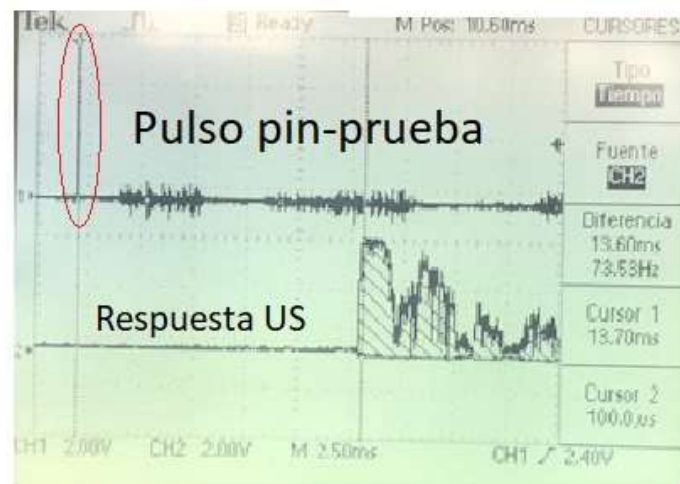


Imagen 29

DATO A TUltraWifi 1648 TUltraPin 13572 TWifi 11924 ints 3136 3183 3231 3279 3328 3377 3426 3477 3528

Imagen 30

En la imagen 29, se ha utilizado una función del osciloscopio llamada “cursores”. Con esta herramienta podremos medir la diferencia de tiempos que se produce en la realidad entre la emisión del pulso del pin de prueba y la llegada del pulso ultrasónico al receptor. Esta diferencia sería equivalente al tiempo TUltraPin anteriormente explicado. En el margen derecho de la imagen 29 se puede apreciar un recuadro que marca la

diferencia de tiempo entre los cursores y que indica que TUltraPin vale 13.6 ms. Si bien ahora se observa la imagen 25, correspondiente a la salida por el monitor serie del código del receptor, se puede comprobar como ese tiempo es prácticamente el mismo. Con lo cual, esta medida es fiable.

- TWifi: Se obtiene con la resta de tiempos entre la cuenta 2 y la cuenta 1. Será el tiempo transcurrido entre haber recibido la señal del pin de prueba y la señal de radio. Este tiempo será muy largo en el caso del emisor A y prácticamente nulo en los emisores B y C. Deberá mantenerse relativamente constante a lo largo del tiempo para que los datos que se lean por el monitor serie sean fiables.

Con este código y esta configuración hardware, se obtienen unas interrupciones constantes por parte del emisor A. Sin embargo, las interrupciones correspondientes a los emisores B y C son inestables e incoherentes.

A continuación, se procede a comprobar si los pulsos de prueba y los pulsos de disparo del ultrasonido no tienen más retraso entre ellos que el programado. Inicialmente, se enviaba el pulso de prueba, a continuación, el mensaje "A" por radio y por último se enviaba el pulso para generar el ultrasonido como se muestra en la imagen 31 (salvo el envío del pulso ultrasónico). Debido a que el tiempo de emisión y recepción de mensajes por medio del módulo NRF24L01 no es constante, se decide enviar el mensaje de radio antes de los dos pulsos de tal manera que el desfase entre los pulsos no sea mayor que el programado, quedando entonces el orden de instrucciones en la forma que se muestra en la imagen siguiente:

```
// Enviar el mensaje A
radio.stopListening();
delayMicroseconds(10);
radio.write( &dataToSend, 1);

delay(5);

delayMicroseconds(100);
digitalWrite(pin_prueba, HIGH);
delayMicroseconds(100);
digitalWrite(pin_prueba, LOW);
Serial.print("\n");
delay(5);

// Emitir el Ultrasonido
digitalWrite(pin_SR04_Trigger, HIGH);
delayMicroseconds(100);
digitalWrite(pin_SR04_Trigger, LOW);
delayMicroseconds(100);
```

Imagen 31

Habiendo ya conseguido un tiempo entre pulsos constantes, siguen sin alcanzarse tiempos estables de interrupciones debido a que no es un problema con la emisión de pulsos ultrasónicos si no que es un problema relacionado con el envío de mensajes por radio, por lo que se realizan nuevas modificaciones para ver con mayor claridad que ocurre con los tiempos de recepción de mensajes de radio en el receptor en comparación con los tiempos de recepción de pulsos ultrasónicos.

Para ello, las cuentas 1, 2 y 3 serán iniciadas en distintos lugares del código y servirán ahora para comparar los tiempos mencionados anteriormente.

```
while (!radio.available());

cuenta1 = micros();
```

Imagen 32: Valor de la cuenta 1

```
if (dataReceived[0] == 'A'){
  while (digitalRead(pinMedidaEmisorOn) == 0);
}
cuenta2 = micros();|
while ((digitalRead(pinEntrada1) == 0)&& (micros() - TiempoInicial < TIEMPO_REINICIO))
.
cuenta3 = micros();
```

Imagen 33: Valores de las cuentas 2 y 3

Como podemos ver en las imágenes 32 y 33, pertenecientes al código del receptor, la cuenta 1 contabilizará el tiempo que tarde en recibirse la señal inalámbrica de radio. LA cuenta 2, registrará el tiempo que se haya tardado en recibir el pulso de referencia por cable y la cuenta 3 guardará el tiempo transcurrido en activarse el pin de recepción de ultrasonidos.

Por lo tanto, con estas modificaciones, el cálculo de TUltraPin y TUltraWifi será distinto:

- TUltraPin: Se calculará como la resta de la cuenta 3 menos la cuenta 2, siendo este el tiempo transcurrido entre haber recibido el pulso ultrasónico y el tiempo pasado en activarse el pin de prueba activado por cable.
- TUltraWifi: Se calculará como la diferencia entre la cuenta 3 y la cuenta 1, siendo este entonces el tiempo transcurrido entre haberse recibido el pulso ultrasónico y haberse detectado la señal de radio.

Con los cambios que se habían realizado en los emisores, mencionados anteriormente, se consiguen tiempos TUltraPin estables y proporcionales a la distancia entre receptor y emisor, con ello se descarta que las variaciones temporales provengan de la emisión y recepción de pulsos. Sin embargo, es imprescindible conseguir estos mismos resultados para TUltraWifi ya que es proporcional al valor de tiempo por interrupciones, por lo que si TUltraWifi es inestable y no proporcional a la distancia emisor-receptor, con las interrupciones ocurrirá lo mismo.

Una vez acotado el principal problema de este proyecto, siendo la inestabilidad de los tiempos de emisión y recepción de los mensajes de radio, se comenzarán a realizar modificaciones en la función `setup()` de todos los emisores y del receptor en lo referente a los comandos que configuran la radio. Algunos de estos comandos configuran el número de reintentos de emisión y recepción de los mensajes de radio y el delay entre estos reintentos. Otros cambios como la velocidad de transmisión también han sido probados pero tras realizar estos cambios se pierde la comunicación entre los emisores A, B y C con lo que se para aquí la investigación del problema y se plantea como continuación de este trabajo.

4.3 DISEÑO 3D:

En este trabajo, se han desarrollado un total de cuatro piezas 3D. La necesidad del desarrollo de las mismas nace por haber quedado las anteriores piezas incapacitadas de ser reutilizables.

En el chequeo de los sensores de ultrasonidos descrito anteriormente en la memoria, el soporte de los sensores de ultrasonidos del receptor se parte. Este se rompe puesto que los sensores de ultrasonidos están sujetos con silicona a una estructura un tanto débil, que al hacer fuerza para retirar la silicona se quiebra. Además, el soporte de los sensores de ultrasonidos receptores posee una estructura difícil de anclar al receptor puesto que no tiene una superficie plana lo suficientemente grande como para pegarse con seguridad.

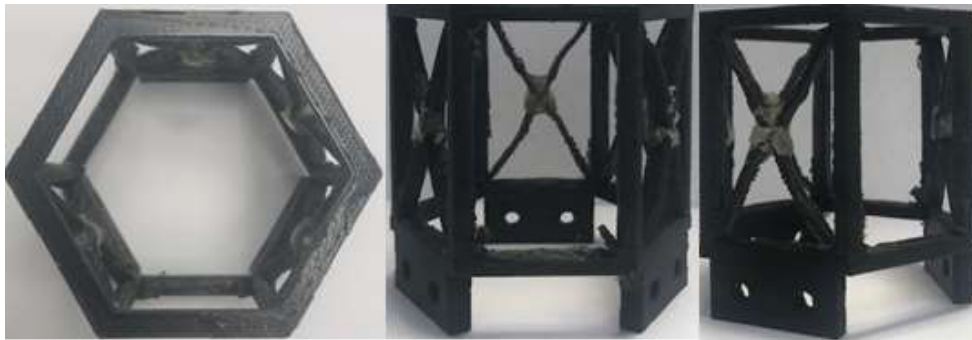


Imagen 34

En la imagen 34 se puede apreciar las roturas producidas en donde irían colocados los sensores de ultrasonidos, que hicieron necesario el rediseño de la pieza.

En el caso del soporte de los ultrasonidos emisores ocurre algo parecido al cambiar uno de los módulos de radio, con lo que se decide cambiar los tres soportes.



Imagen 35

En la imagen 35 se muestra el diseño anterior.

Partiendo de este problema, se utilizará Autodesk Inventor 2020 para rediseñar ambos soportes puesto que tampoco se dispone los archivos “.stl”. Tomando como referencia los soportes realizados por D. Agustín León García, los diseños nuevos toman la siguiente forma:

- El nuevo soporte de los receptores ultrasónicos ahora cuenta con tres plataformas de mayor superficie con la finalidad de poder adherirse bien al robot y unas zonas específicas en los laterales donde se puedan pegar de manera segura los sensores a la estructura.

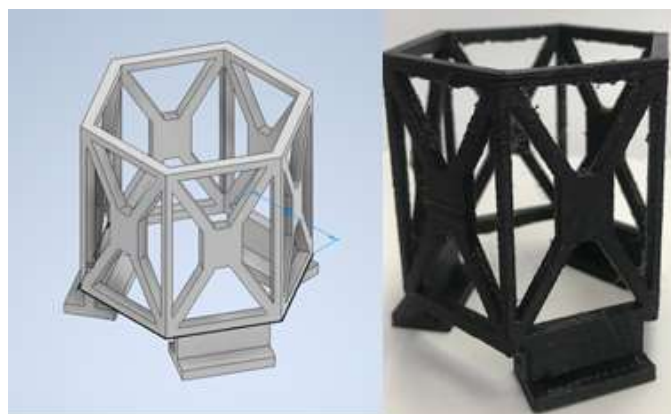


Imagen 36: Boceto 3D vs Impresión 3D I

- Para la estructura diseñada con la finalidad de soportar los ultrasonidos emisores, los módulos de radiofrecuencia y las placas de Arduino que conforman las balizas, se realiza un nuevo diseño más resistente y con una zona

específica para pegar el módulo de radiofrecuencia con seguridad como no ocurría anteriormente.

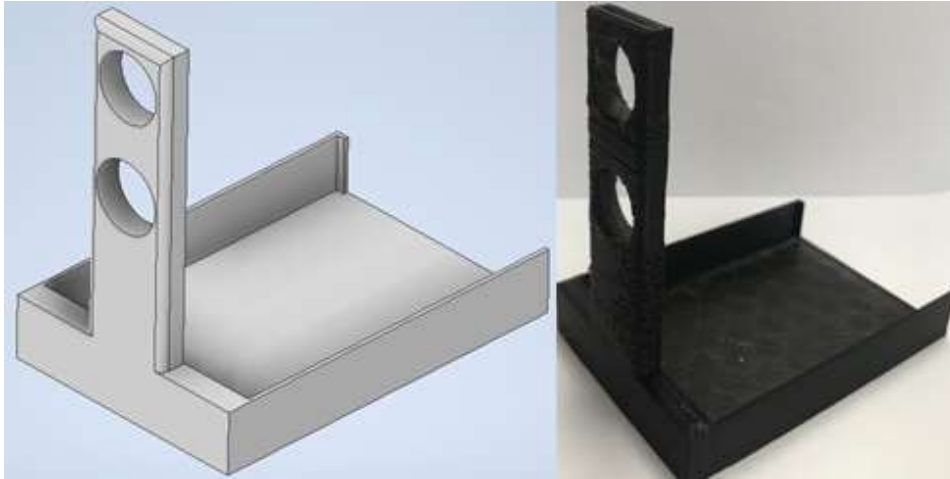


Imagen 37: Boceto 3D vs Impresión 3D II

El diseño 3D con los componentes montados adecuadamente tendrá aspecto que muestra la 37. Los planos de alzado planta y perfil de las figuras se encontrarán en los anexos de esta memoria.

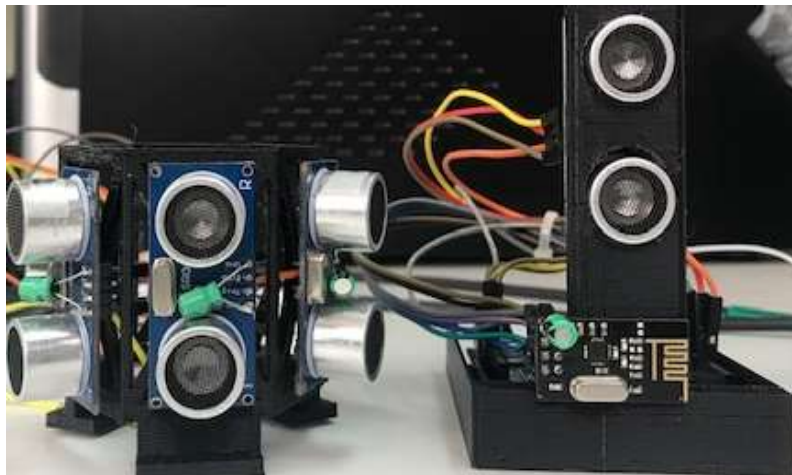


Imagen 38

CAPÍTULO 5: CONCLUSIÓN Y PROPUESTAS DE MEJORA.

5.1 CONCLUSIÓN Y PROPUESTAS DE MEJORA

Tras la utilización de muchos códigos, distintos componentes y muchas horas de trabajo, se ha intentado obtener un sistema de localización de interiores preciso. Tras tener problemas con la comunicación por motivo de los módulos de radio empleados, no se consigue realizar un sistema de localización eficiente pese a sí haber obtenido un sistema que al menos sería capaz de obtener tiempos de interrupción estables y proporcionales a la distancia entre un ultrasonido emisor y uno receptor, siempre que estuviesen apagadas el resto de balizas emisoras.

El conseguir unos tiempos de interrupción constantes para poder comenzar a realizar el cálculo de las distancias ha sido el principal problema hallado en este trabajo y es en ello en lo que más esfuerzo y tiempo se ha empleado, siendo así el problema de mayor relevancia. Con la consecución de la estabilidad de estos tiempos se consideraría que se hubiera conseguido realizar un sistema de localización capaz de conocer el posicionamiento del robot con fiabilidad en los resultados. Sin embargo, multitud de problemas fueron apareciendo a lo largo del desarrollo del sistema de localización como fallos en los sensores de ultrasonidos, tiempos de emisión y recepción variantes con respecto a los módulos de comunicación por radiofrecuencia.

Como principal propuesta de mejora, se propone establecer un sistema capaz de conseguir tiempos estables de interrupciones que permitan el cálculo de distancias, y que pueda controlar el tiempo empleado por el módulo de radiofrecuencias en enviar y recibir mensajes que es lo que debilita el sistema actual.

Otra propuesta de mejora puede ser el establecimiento de un código más sencillo que llame a funciones para calcular la distancia entre baliza y robot así como la posición final(x,y) del robot en el plano.

El cálculo del ángulo de orientación también se propone como mejora a este sistema de localización para que el robot pueda estar correctamente localizado.

La comunicación del sistema de localización con el robot de tracción diferencial a través de ROS (Robot Operating System) era uno de los objetivos fijados por D. Agustín León García en el trabajo de referencia y en lo que se pretendía centrar el desarrollo de este proyecto entre otras cosas. Puesto que no se ha avanzado en ese aspecto, queda también como posible mejora a este trabajo.

5.2 CONCLUSION AND FUTURE IMPROVEMENTS

After the use of many codes, different components and many hours of work, an attempt has been made to obtain a precise interior location system. After having problems with communication due to the radio modules used, an efficient localization system is not achieved, despite having obtained a system that would at least be able to obtain stable interruption times proportional to the distance between an ultrasound emitter and an ultrasound receiver, provided that the rest of the emitting beacons were off.

Achieving constant interruption times in order to start calculating distances has been the main problem found in this work and it is in this that the most time and effort has been employed, thus being the most relevant problem. With the achievement of the stability of these times it would be considered that a location system capable of knowing the positioning of the robot with reliability in the results had been achieved.

However, a multitude of problems appeared during the development of the localization system, such as failures in the ultrasonic sensors, different transmission and reception times with respect to the radiofrequency communication modules.

As the main improvement proposal, it is proposed to establish a system capable of achieving stable times of interruptions that allow the calculation of distances, and that could control the time used by the radio frequency module in sending and receiving messages, which is what weakens the current system .

Another improvement proposal could be the establishment of a simpler code that calls functions to calculate the distance between the beacon and robot as well as the final position (x, y) of the robot in the plane.

The calculation of the orientation angle is also proposed as an improvement to this location system so that the robot can be correctly located.

The communication of the location system with the robot of differential traction through ROS (Robot Operating System) was one of the objectives set by D. Agustín León García in the reference work and in what was intended to focus the development of this project among other things. Since no progress has been made in this regard, it remains as a possible improvement to this work.

ANEXOS

ANEXO 1: Tabla de resultados

Tabla 1: Tiempo por interrupción asociado a cada emisor

Distancia	Tiempo/ interrupción (EMISOR A)	Tiempo/ interrupción (EMISOR B)	Tiempo/ interrupción (EMISOR C)
20 cm aprox.	3066 3119 3166 3214	29055 29106 29155	3309 3359 3407 3454
	3263 3310 3360 3409	29203 29251 29299	3502 3550 3598 3646
	3458	29348 29396 29446	3694
20 cm aprox.	3084 3137 3183 3231	29089 29140 29188	8151 8202 8249 8295
	3279 3328 3377 3426	29237 29285 29333	8343 8391 8439 8487
	3475	29381 29429 29478	8535
20 cm aprox.	3213 3260 3307 3355	13425 13476 13525	3315 3368 3414 3461
	3404 3453 3505 3556	13573 13621 13669	3509 3557 3605 3653
	3606	13718 13766 13815	3702
20 cm aprox.	3184 3233 3280 3329	13345 13398 13445	3272 3324 3373 3421
	3377 3426 3476 3527	13493 13541 13590	3469 3517 3565 3613
	3578	13639 13689 13736	3662
20 cm aprox.	3210 3259 3307 3355	8393 8448 8495 8544	3363 3413 3463 3511
	3404 3453 3503 3554	8592 8639 8688 8737	3559 3607 3655 3703
	3605	8785	3752
20 cm aprox.	3258 3307 3355 3403	29094 29146 29193	3321 3374 3420 3468
	3451 3500 3550 3601	29241 29290 29338	3516 3564 3612 3660
	3658	29388 29435 29485	3709
20 cm aprox.	3261 3309 3356 3404	29038 29088 29136	3395 3448 3496 3544
	3453 3502 3552 3603	29184 29232 29281	3591 3640 3687 3736
	3653	29330 29378 29427	3784
20 cm aprox.	3236 3285 3332 3379	29035 29090 29138	3369 3420 3469 3517
	3429 3477 3527 3578	29187 29235 29284	3565 3613 3661 3709
	3629	29334 29381 29430	3758
20 cm aprox.	3211 3258 3305 3353	3538 3590 3637 3685	3371 3421 3472 3519
	3402 3452 3501 3553	3733 3782 3831 3879	3567 3615 3663 3712
	3603	3928	3760
20 cm aprox.	3239 3289 3336 3384	3519 3571 3619 3668	3363 3414 3463 3510
	3433 3482 3532 3584	3716 3765 3813 3862	3558 3606 3654 3703
	3633	3911	3751
20 cm aprox.	3240 3289 3336 3384	3950 4003 4051 4100	3393 3446 3493 3540
	3433 3482 3532 3584	4149 4196 4246 4294	3588 3636 3684 3732
	3633	4343	3781
20 cm aprox.	3228 3277 3324 3372	4352 4402 4449 4497	3407 3459 3510 3557
	3420 3470 3519 3571	4545 4594 4642 4691	3604 3653 3701 3749
	3621	4739	3798
20 cm aprox.	3230 3279 3327 3375	4321 4371 4419 4467	4453 4503 4553 4601
	3424 3473 3522 3574	4515 4564 4613 4661	4651 4700 4749 4799
	3624	4710	4849

20 cm aprox.	3412 3462 3509 3557 3606 3655 3704 3756 3806	4398 4447 4495 4543 4591 4641 4689 4737 4786	4389 4439 4490 4539 4588 4636 4685 4733 4781
20 cm aprox.	4369 4422 4469 4517 4566 4615 4664 4714 4762	4334 4385 4433 4481 4529 4578 4627 4676 4725	4380 4433 4479 4527 4575 4623 4672 4720 4769
20 cm aprox.	9145 9198 9244 9292 9340 9389 9438 9487 9538	4287 4345 4388 4435 4484 4532 4581 4631 4679	29293 29343 29390 29437 29485 29533 29581 29630 29678
20 cm aprox.	29677 29728 29775 29822 29870 29920 29968 30017 30068	9200 9249 9297 9345 9394 9443 9492 9540 9589	8746 8798 8844 8891 8939 8987 9035 9083 9131
40 cm aprox.	4431 4485 4532 4580 4629 4677 4726 4775 4826	9231 9280 9328 9376 9425 9473 9523 9571 9620	23407 23462 23508 23555 23603 23651 23699 23747 23795
40 cm aprox.	4213 4267 4313 4363 4410 4458 4507 4557 4606	4348 4401 4449 4496 4545 4593 4642 4691 4740	29355 29404 29450 29499 29546 29593 29643 29691 29737
40 cm aprox.	4245 4295 4342 4390 4438 4487 4536 4585 4634	4522 4574 4626 4676 4724 4773 4822 4871 4920	23387 23438 23485 23532 23579 23627 23675 23723 23772
40 cm aprox.	4215 4267 4314 4362 4411 4460 4509 4558 4607	4720 4773 4822 4870 4919 4968 5018 5066 5115	29347 29399 29447 29495 29541 29589 29639 29686 29735
40 cm aprox.	4219 4270 4318 4365 4415 4463 4512 4561 4610	4612 4663 4714 4762 4810 4858 4906 4953 5001	23701 23752 23802 23849 23896 23944 23993 24041 24089
40 cm aprox.	4245 4296 4343 4391 4439 4490 4537 4587 4636	9548 9600 9648 9696 9744 9799 9841 9889 9938	29731 29783 29831 29877 29925 29972 30020 30068 30116
40 cm aprox.	4190 4244 4290 4338 4387 4436 4485 4534 4583	4645 4698 4747 4794 4842 4890 4939 4987 5036	29607 29658 29706 29753 29801 29849 29897 29945 29994
40 cm aprox.	9460 9512 9562 9609 9657 9705 9753 9802 9850	14388 14438 14486 14532 14580 14630 14677 14726 14774	28587 28638 28687 28734 28783 28830 28879 28927 28976
60 cm aprox.	4561 4611 4659 4709 4758 4807 4856 4906 4955	18951 19001 19048 19097 19145 19194 19245 19292 19341	29634 29685 29734 29781 29828 29876 29925 29973 30021
60 cm aprox.	4590 4638 4685 4734 4783 4832 4881 4930 4979	4515 4564 4611 4659 4707 4756 4804 4853 4901	9091 9142 9189 9236 9285 9333 9381 9429 9477
60 cm aprox.	4599 4647 4695 4744 4794 4843 4892 4942 4991	23999 24050 24098 24145 24194 24242 24290 24338 24387	23663 23716 23763 23810 23857 23905 23953 24001 24050
60 cm aprox.	29017 29066 29114 29163 29213 29261	9397 9449 9498 9549 9601 9648 9698 9745	29700 29751 29798 29844 29892 29940

	29311 29360 29409	10139	29988 30036 30084
60 cm aprox.	23942 23993 24041 24089 24137 24186 24236 24285 24334		23995 24043 24093 24141 24190 24237 24286 24333 24382
60 cm aprox.	30020 30068 30116 30164 30212 30261 30310 30359 30409		
60 cm aprox.	30071 30118 30166 30214 30262 30311 30360 30409 30459		
60 cm aprox.	9482 9530 9578 9626 9674 9724 9772 9821 9871		

ANEXO 2: PRESUPUESTO

PRESUPUESTO

PRESUPUESTO DEL TRABAJO PRECEDENTE DESGLOSADO			
Materiales y componentes			
Material/componente	Precio unitario	Cantidad	Total
Módulos HC-SR04	3,37 €	9	30,33 €
Módulo NRF24L01	1,96 €	4	7,84 €
Placas Arduino UNO	19,95 €	3	59,85 €
Placas Arduino MEGA	32,86 €	1	32,86 €
Raspberry Pi 3 Model B	34,14 €	1	34,14 €
Gopigo 2 Base Kit	119,99 €	1	119,99 €
Sandisk MicroSDHC 32 GB	13,99 €	1	13,99 €
		TOTAL	299,00 €
Costes de desarrollo			
Trabajo realizado	Precio/h	Horas totales	Total
Análisis y documentación	10,00 €	20	200,00 €
Montaje del hardware	15,00 €	15	225,00 €
Programación	25,00 €	145	3.625,00 €
Pruebas	15,00 €	40	600,00 €
Documentación	10,00 €	45	450,00 €
		TOTAL	5.100,00 €
Balance total			
Material/componentes			299,00 €
Trabajo realizado			5.100,00 €
		TOTAL	5.399,00 €

Tabla 1: Presupuesto del trabajo de referencia

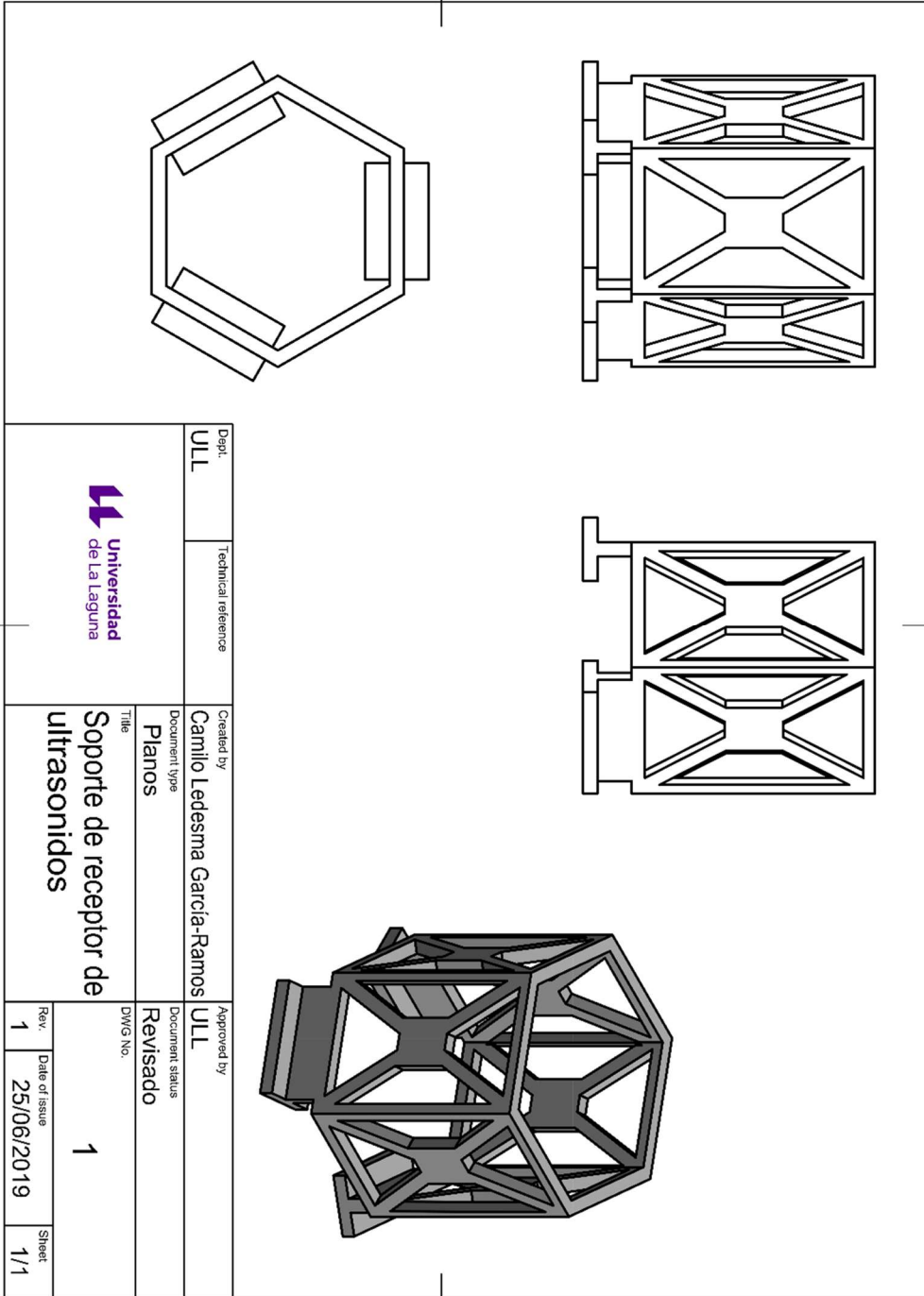
PRESUPUESTO DEL TRABAJO CONTINUADO DESGLOSADO			
Materiales y componentes			
Material/componente	Precio unitario	Cantidad	Total
Módulos HC-SR04	3,37 €	2	6,74 €
Módulo NRF24L01	1,96 €	2	3,92 €
Placas Arduino MEGA	32,86 €	1	32,86 €
		TOTAL	43,52 €
Costes de desarrollo			
Trabajo realizado	Precio/h	Horas totales	Total
Análisis y documentación	10,00 €	30	300,00 €
Programación	25,00 €	160	4.000,00 €
Pruebas	15,00 €	100	1.500,00 €
Documentación	10,00 €	45	450,00 €
		TOTAL	6.250,00 €
Balance total			
Material/componentes			43,52 €
Trabajo realizado			6.250,00 €
		TOTAL	6.293,52 €


Tabla 2: Presupuesto del trabajo continuado

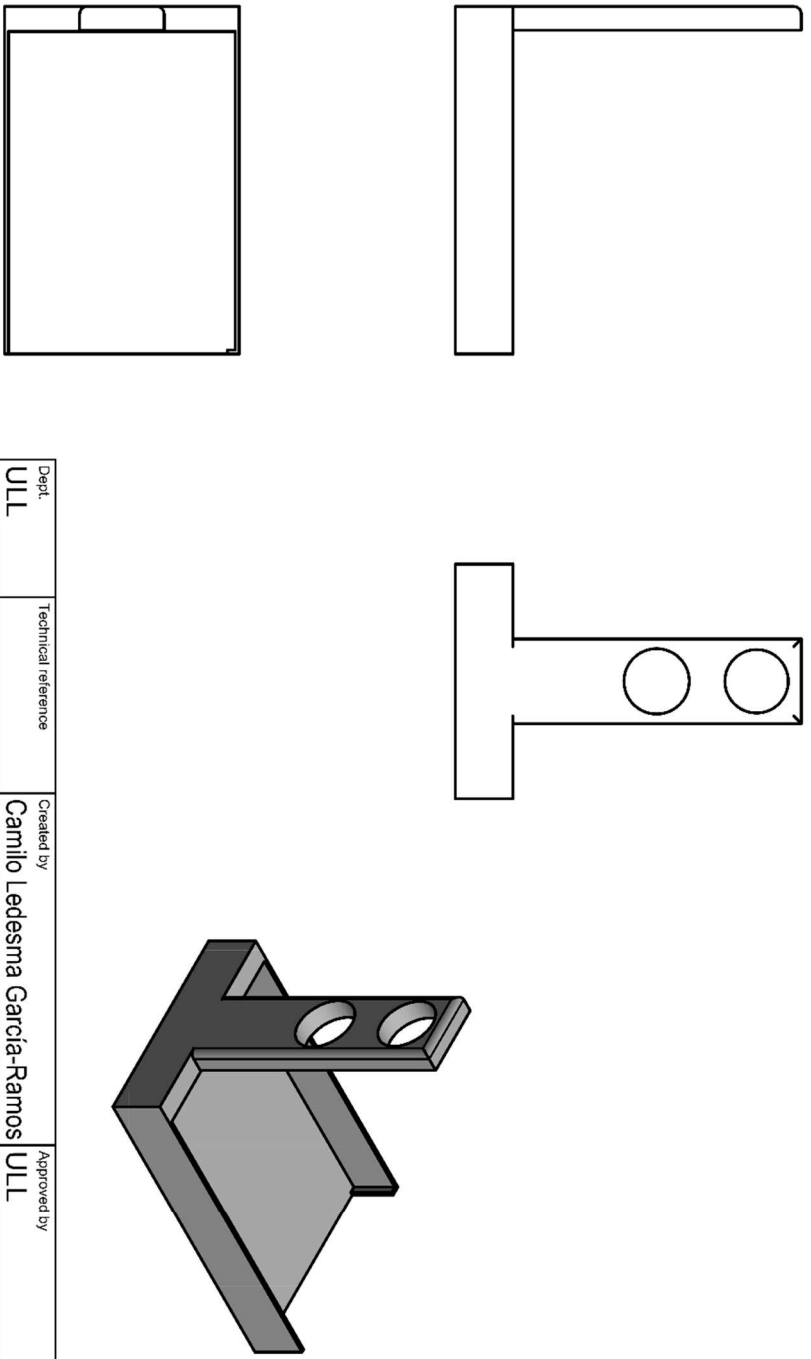
PRESUPUESTO TOTAL DEL PROYECTO		
Primer año de trabajo		5.399,00 €
Segundo año de trabajo		6.293,52 €
TOTAL		11.692,52 €


Tabla 3: Presupuesto final del proyecto

ANEXO 3: PLANOS



Dept. ULL	Technical reference	Created by Camilo Ledesma García-Ramos	Approved by ULL
		Document type Planos	Document status Revisado
		Title Soporte de receptor de ultrasonidos	DWG No. 1
Rev. 1	Date of issue 25/06/2019	Sheet 1/1	



Dept. ULL	Technical reference	Created by Carnilo Ledesma García-Ramos	Approved by ULL
		Document type Planos	Document status Revisado
		Title Soporte de balizas ultrasónicas	
Rev. 1	Date of issue 25/06/2019	DWG No. 2	
		Sheet 1/1	

ANEXO 4: Códigos

Código emisor A

```
// CÓDIGO DEL EMISOR A  
  
// Autor: Camilo Ledesma García-Ramos  
  
//Este código irá implementado en el emisor que se elija como emisor A
```

```
#define TIEMPO_ENVIO 10000  
  
#define TIEMPO_ESPERA 2000  
  
  
  
//LIBRERÍAS  
  
#include <RF24_config.h>  
  
#include <nRF24L01.h>  
  
#include <RF24.h>  
  
#include <RF24_config.h>  
  
#include <SPI.h>  
  
//Configuración RADIO  
  
const int pinCE = 9;  
  
const int pinCSN = 10;  
  
RF24 radio(pinCE, pinCSN);  
  
const byte slaveAddress[5] = {'R','x','A','A','A'};  
  
  
  
//Configuración ULTRASONIDOS  
  
const int pin_SR04_Trigger = 6;  
  
const int pin_SR04_Echo = 5;  
  
const int pin_prueba = 2;  
  
unsigned long Ultimo_envio = 0;  
  
char mensaje[1];
```

```
char dataToSend[10] = "A";
```

```
void setup() {  
  Serial.begin(9600);  
  radio.begin();  
  radio.setDataRate( RF24_250KBPS );  
  radio.setRetries(3, 5);  
  radio.openWritingPipe(slaveAddress);  
  radio.openReadingPipe(1, slaveAddress);  
  radio.startListening();  
  pinMode(pin_SR04_Trigger, OUTPUT);  
  pinMode(pin_prueba, OUTPUT);  
  digitalWrite(pin_prueba, LOW);  
  Serial.println("Emisor A");  
}
```

```
void loop() {  
  
  while ((!radio.available()) && (millis() - Ultimo_envio < TIEMPO_ENVIO));  
  
  if (millis() - Ultimo_envio >= TIEMPO_ENVIO){  
    Ultimo_envio = millis();  
  
    radio.stopListening();  
    delayMicroseconds(10);  
    radio.write( &dataToSend, 1 );  
  
    delay(5);
```

```
delayMicroseconds(100);

digitalWrite(pin_prueba, HIGH);

delayMicroseconds(100);

digitalWrite(pin_prueba, LOW);

delay(5);

Serial.print("Mensaje A enviado, no se recibe mensaje C");

Serial.print("\n");

// Emitir el Ultrasoni

digitalWrite(pin_SR04_Trigger, HIGH);

delayMicroseconds(100);

digitalWrite(pin_SR04_Trigger, LOW);

delayMicroseconds(100);

delayMicroseconds(10);

radio.startListening();

} else {

radio.read(mensaje,1);

while (radio.available())

radio.read(mensaje,1);

if(mensaje[0] == 'C'){
```

```
Serial.println("Mensaje C recibido");

Serial.println("Nuevo Mensaje A");
Ultimo_envio = millis();

delay(TIEMPO_ESPERA-10);

mensaje[0] = 'A';

// Enviar el mensaje A
radio.stopListening();
delayMicroseconds(10);
radio.write( &dataToSend, 1);

delay(5);

delayMicroseconds(100);
digitalWrite(pin_prueba, HIGH);
delayMicroseconds(100);
digitalWrite(pin_prueba, LOW);
Serial.print("\n");
delay(5);

// Emitir el Ultrasonido
digitalWrite(pin_SR04_Trigger, HIGH);
delayMicroseconds(100);
digitalWrite(pin_SR04_Trigger, LOW);
delayMicroseconds(100);
```

```
    delayMicroseconds(10);  
    radio.startListening();  
  
    }  
    }  
}
```

Código emisor B

```
// CÓDIGO DEL EMISOR B
```

```
// Autor: Camilo Ledesma García-Ramos
```

```
//Este código irá implementado en el emisor que se elija como emisor B
```



```
#include <printf.h>

#include <RF24_config.h>

#define TIEMPO_ENVIO 3000

#define TIEMPO_ESPERA 2000

// Librerías
#include <nRF24L01.h>
#include <RF24.h>
#include <RF24_config.h>
#include <SPI.h>

//Configuración RADIO
const int pinCE = 9;
const int pinCSN = 10;
RF24 radio(pinCE, pinCSN);

const byte slaveAddress[5] = {'R','x','A','A','A'};

// Configuración ULTRASONIDOS
const int pin_SR04_Trigger = 6;
const int pin_SR04_Echo = 5;
const int pin_prueba = 2;
char mensaje[1];
char dataToSend[10] = "B";

void setup() {
```

```
// COMUNICACIÓN

Serial.begin(9600);

radio.begin();

radio.setDataRate( RF24_250KBPS );

radio.setRetries(3, 5);

radio.openWritingPipe(slaveAddress);

radio.openReadingPipe(1, slaveAddress);

radio.startListening();

pinMode(pin_SR04_Trigger, OUTPUT);

Serial.println("Emisor B");

}

void loop() {

while(!radio.available());

radio.read(mensaje,1);

while (radio.available())

  radio.read(mensaje,1);

if(mensaje[0] == 'A'){

  Serial.print("Mensaje A recibido");

  Serial.println("\n");

  Serial.println("Nuevo Mensaje B");

  Serial.println("\n");
```

```
    delay(TIEMPO_ESPERA);

    // Enviar el mensaje B
    radio.stopListening();
    delayMicroseconds(10);
    radio.write( &dataToSend, 1 );

    delay(7);

    // Emitir el Ultrasonido
    digitalWrite(pin_SR04_Trigger, HIGH);
    delayMicroseconds(100);
    digitalWrite(pin_SR04_Trigger, LOW);
    delayMicroseconds(100);

    Serial.print("Mensaje B emitido");
    Serial.println("\n");

    delayMicroseconds(10);
    radio.startListening();

}

}
```

Código emisor C

```
// CÓDIGO DEL EMISOR C
```

```
// Autor: Camilo Ledesma García-Ramos
```

```
//Este código irá implementado en el emisor que se elija como emisor C
```

```
#include <printf.h>
```

```
#include <RF24_config.h>
```

```
#define TIEMPO_ENVIO 3000
```

```
// Librerías

#include <nRF24L01.h>

#include <RF24.h>

#include <RF24_config.h>

#include <SPI.h>

#define TIEMPO_ESPERA 2000

//Configuración RADIO

const int pinCE = 9;

const int pinCSN = 10;

RF24 radio(pinCE, pinCSN);

const byte slaveAddress[5] = {'R','x','A','A','A'};

//Configuración ULTRASONIDOS

const int pin_SR04_Trigger = 6;

const int pin_SR04_Echo = 5;

const int pin_prueba = 2;

char mensaje[1];

char datatoSend[10] = "C";

void setup() {

    // COMUNICACIÓN

    Serial.begin(9600);

    radio.begin();

    radio.setDataRate( RF24_250KBPS );
```

```
radio.setRetries(3, 5);

radio.openWritingPipe(slaveAddress);

radio.openReadingPipe(1, slaveAddress);

radio.startListening();

pinMode(pin_SR04_Trigger, OUTPUT);

Serial.println("Emisor C");

}

void loop() {

while(!radio.available());

radio.read(mensaje,1);

while (radio.available())

radio.read(mensaje,1);

if(mensaje[0] == 'B'){

Serial.println("Mensaje recibido B");

Serial.println("\n");

Serial.println("Nuevo Mensaje C");

Serial.println("\n");

delay(TIEMPO_ESPERA);

// Enviar el mensaje C
```

```
radio.stopListening();  
  
delayMicroseconds(10);  
  
radio.write( &datatoSend, 1 );  
  
  
delay(7);  
  
  
// Emitir el Ultrasonido  
digitalWrite(pin_SR04_Trigger, HIGH);  
delayMicroseconds(100);  
digitalWrite(pin_SR04_Trigger, LOW);  
delayMicroseconds(100);  
  
delayMicroseconds(10);  
radio.startListening();  
  
}  
}
```

Código del receptor

```
// CÓDIGO DEL RECEPTOR  
  
// Autor: Camilo Ledesma García-Ramos  
  
//Este código irá implementado en la placa Arduino MEGA 2560 correspondiente al receptor  
  
  
//Librerías  
  
#include <nRF24L01.h>  
  
#include <RF24.h>  
  
#include <RF24_config.h>  
  
#include <SPI.h>  
  
#include <TimerThree.h>
```

```
//Configuración RADIO
const int pinCE = 9;
const int pinCSN = 10;
const byte thisSlaveAddress[5] = {'R','x','A','A','A'};
RF24 radio(pinCE, pinCSN);
const int pinEntrada1 = 2;
const int pinEntrada2 = 3;
const int pinEntrada3 = 19;

const int pinMedidaEmisorOn = 21;

#define NUM_INT 9
int num_int1 = 0;
int num_int2 = 0;
int num_int3 = 0;
unsigned long tiempoInt1[NUM_INT];
unsigned long tiempoInt2[NUM_INT];
unsigned long tiempoInt3[NUM_INT];
char dataReceived[10];
float distancia[6];
int i;
int j;
int cuenta;
```



```
unsigned long TiempoInicial;
```

```
void setup() {  
  Serial.begin(9600);  
  Serial.println("Escuchando...");  
  radio.begin();  
  radio.setDataRate( RF24_250KBPS );  
  radio.setRetries(3, 5);  
  radio.openReadingPipe(1, thisSlaveAddress);  
  radio.startListening();  
  Timer3.initialize(50000);  
  attachInterrupt(digitalPinToInterrupt(pinEntrada1), calcDistancia1, RISING);  
  attachInterrupt(digitalPinToInterrupt(pinEntrada2), calcDistancia2, RISING);  
  attachInterrupt(digitalPinToInterrupt(pinEntrada3), calcDistancia3, RISING);  
  Serial.print ("\n");  
  Serial.println("INICIANDO");  
  pinMode(pinEntrada1, INPUT);  
  pinMode(pinEntrada2, INPUT);  
  pinMode(pinEntrada3,INPUT);  
  cuenta=0;  
  #define TIEMPO_REINICIO 100000  
}
```

```
void calcDistancia1(void){  
  noInterrupts();  
  if (num_int1 < NUM_INT)  
    tiempoInt1[num_int1++] = TCNT3;
```

```
    interrupts();
}

//void calcDistancia2(void){
    noInterrupts();
    if (num_int2 < NUM_INT)
        tiempoInt2[num_int2++] = TCNT3;
    interrupts(); //cambio introducido
}

void calcDistancia3(void){
    noInterrupts();
    if (num_int3 < NUM_INT)
        _tiempoInt3[num_int3++] = TCNT3;
    _interrupts();
}

void loop() {
    cuenta ++;

    unsigned long cuenta1 = 0;
    unsigned long cuenta2 = 0;
    unsigned long cuenta3 = 0;

    unsigned long TEspera;
    unsigned long TespSig;

    while (!(radio.available()));

    cuenta1 = micros();
```

```
radio.read(&dataReceived, 1);  
noInterrupts();  
num_int1 = 0;  
num_int2 = 0;  
num_int3 = 0;  
Timer3.restart();  
interrupts();  
//puesta a 0 de la cuenta  
  
Tiempolnicial = micros();  
  
while (radio.available())  
    radio.read(&dataReceived, 1);  
  
if (dataReceived[0] == 'A'){  
    while (digitalRead(pinMedidaEmisorOn) == 0);  
  
    }  
  
    cuenta2 = micros();  
  
    while ((digitalRead(pinEntrada1) == 0)&& (micros() - Tiempolnicial <  
TIEMPO_REINICIO)) ;  
  
    cuenta3 = micros();
```

```
TEspera = cuenta3 - cuenta2;

TespSig = cuenta3 - cuenta1;

if (micros() - TiempoInicial >= TIEMPO_REINICIO) {

    Serial.print(cuenta);

    Serial.print(" ERRORMAXTIME ");

    Serial.print(dataReceived[0]);

    Serial.print(" TUltraWifi ");

    Serial.print(TespSig);

    Serial.print(" TUltraPin ");

    Serial.print(TEspera);

    Serial.print(" ints ");

    for (int j=0; j < NUM_INT; j++) {

        Serial.print(tiempoInt1[j]);

        Serial.print(" ");

    }

    Serial.print("\n");

    return;

}

if(dataReceived[0] == 'A'){

    Serial.print(cuenta);

    Serial.print(" DATO ");

    Serial.print(dataReceived[0]);

    Serial.print(" TUltraWifi ");

    Serial.print(TespSig);

    Serial.print(" TUltraPin ");

    Serial.print(TEspera);
```

```
Serial.print("\n ");
Serial.print("E1 ints ");

for (int j=0; j < NUM_INT; j++) {
    Serial.print(tiempoInt1[j]);
    Serial.print(" ");
}
Serial.print("\n ");
Serial.print("E2 ints ");
for (int j=0; j < NUM_INT; j++) {
    Serial.print(tiempoInt2[j]);
    Serial.print(" ");
}
Serial.print("\n ");
Serial.print("E3 ints ");
for (int j=0; j < NUM_INT; j++) {
    Serial.print(tiempoInt3[j]);
    Serial.print(" ");
}
Serial.print("\n ");
}
```

```
else{
    Serial.print(cuenta);
    Serial.print(" DATO ");
    Serial.print(dataReceived[0]);
    Serial.print("\n ");
}
```

```
Serial.print("E1 ints ");
for (int j=0; j < NUM_INT; j++) {
    Serial.print(tiempoInt1[j]);
    Serial.print(" ");
}
Serial.print("\n ");
Serial.print("E2 ints ");
for (int j=0; j < NUM_INT; j++) {
    Serial.print(tiempoInt2[j]);
    Serial.print(" ");
}
Serial.print("\n ");
Serial.print("E3 ints ");
for (int j=0; j < NUM_INT; j++) {
    Serial.print(tiempoInt3[j]);
    Serial.print(" ");
}
}
Serial.print("\n ");
}
```

ANEXO 5: Documentación

ROBOT . HEAD to TOE

Product User's Manual – HC-SR04 Ultrasonic Sensor

Cytron
Technologies



User's Manual

V1.0

May 2013

Information contained in this publication regarding device applications and the like is intended through suggestion only and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. No representation or warranty is given and no liability is assumed by Cytron Technologies Incorporated with respect to the accuracy or use of such information or infringement of patents or other intellectual property rights arising from such use or otherwise. Use of Cytron Technologies's products as critical components in life support systems is not authorized except with express written approval by Cytron Technologies. No licenses are conveyed, implicitly or otherwise, under any intellectual property rights.

Index

1.	Introduction	3
2.	Packing List	4
3.	Product Layout	5
4.	Product Specification and Limitation	6
5.	Operation	7
6.	Hardware Interface	8
7.	Example Code	9
8.	Warranty	10

1.0 INTRODUCTION

The [HC-SR04](#) ultrasonic sensor uses sonar to determine distance to an object like bats or dolphins do. It offers excellent non-contact range detection with high accuracy and stable readings in an easy-to-use package. From 2cm to 400 cm or 1" to 13 feet. It operation is not affected by sunlight or black material like Sharp rangefinders are (although acoustically soft materials like cloth can be difficult to detect). It comes complete with ultrasonic transmitter and receiver module.

Features:

- Power Supply :+5V DC
- Quiescent Current : <2mA
- Working Currnt: 15mA
- Effectual Angle: <15°
- Ranging Distance : 2cm – 400 cm/1" - 13ft
- Resolution : 0.3 cm
- Measuring Angle: 30 degree
- Trigger Input Pulse width: 10uS
- Dimension: 45mm x 20mm x 15mm

2.0 PACKING LIST

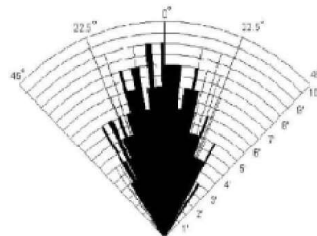
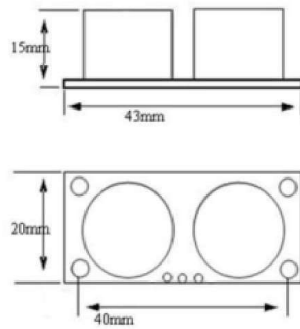


1. 1 x [HC-SR04 module](#)

3.0 PRODUCT LAYOUT



VCC = +5VDC
 Trig = Trigger input of Sensor
 Echo = Echo output of Sensor
 GND = GND



4.0 PRODUCT SPECIFICATION AND LIMITATIONS

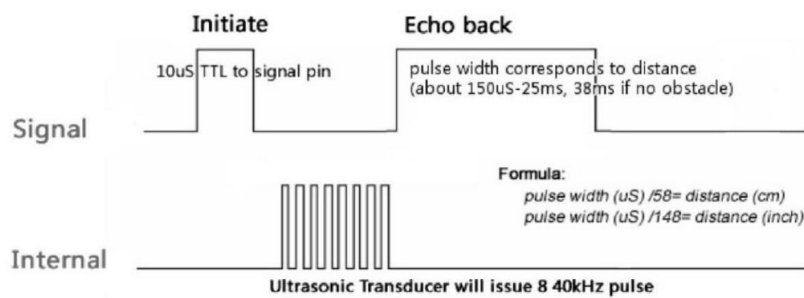
Parameter	Min	Typ.	Max	Unit
Operating Voltage	4.50	5.0	5.5	V
Quiescent Current	1.5	2	2.5	mA
Working Current	10	15	20	mA
Ultrasonic Frequency	-	40	-	kHz

5.0 OPERATION

The timing diagram of [HC-SR04](#) is shown. To start measurement, Trig of SR04 must receive a pulse of high (5V) for at least 10us, this will initiate the sensor will transmit out 8 cycle of ultrasonic burst at 40kHz and wait for the reflected ultrasonic burst. When the sensor detected ultrasonic from receiver, it will set the Echo pin to high (5V) and delay for a period (width) which proportion to distance. To obtain the distance, measure the width (Ton) of Echo pin.

Time = Width of Echo pulse, in uS (micro second)

- Distance in centimeters = Time / 58
- Distance in inches = Time / 148
- Or you can utilize the speed of sound, which is 340m/s

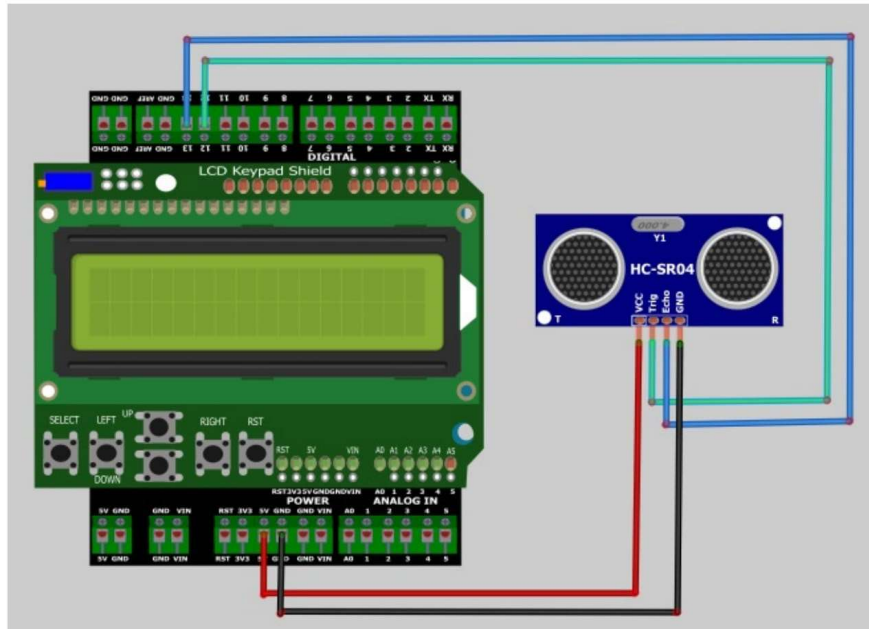


Note:

- Please connect the GND pin first before supplying power to VCC.
- Please make sure the surface of object to be detect should have at least 0.5 meter² for better performance.

6.0 HARDWARE INTERFACE

Here is example connection for Ultrasonic Ranging module to Arduino UNO board. It can be interface with any microcontroller with digital input such as PIC, [SK40C](#), [SK28A](#), [SKds40A](#), [Arduino series](#).



7.0 EXAMPLE CODE

This is [example code](#) Ultrasonic Ranging module. Please download the complete code at the product page.

```
#include "Ultrasonic.h"
#include <LiquidCrystal.h>
LiquidCrystal lcd(8, 9, 4, 5, 6, 7);
Ultrasonic ultrasonic(12,13);

void setup() {
  lcd.begin(16, 2);
  lcd.setCursor(0, 0);
  lcd.print("HC-SR4 testing.");
  delay(1000);
}

void loop()
{
  //lcd.clear();
  lcd.setCursor(0, 1);
  lcd.print(ultrasonic.Ranging(CM));
  lcd.print("cm ");

  delay(100);
}
```

8.0 WARRANTY

- Product warranty is valid for 6 months.
- Warranty only applies to manufacturing defect.
- Damaged caused by miss-use is not covered under warranty
- Warranty does not cover freight cost for both ways.

Prepared by

Cytron Technologies Sdn. Bhd.

19, Jalan Kebudayaan 1A,
Taman Universiti,
81300 Skudai,
Johor, Malaysia.

Tel: +607-521 3178

Fax: +607-521 1861

URL: www.cytron.com.my

Email: support@cytron.com.my

sales@cytron.com.my



nRF24L01+

Single Chip 2.4GHz Transceiver

Preliminary Product Specification v1.0

Key Features

- Worldwide 2.4GHz ISM band operation
- 250kbps, 1Mbps and 2Mbps on air data rates
- Ultra low power operation
- 11.3mA TX at 0dBm output power
- 13.5mA RX at 2Mbps air data rate
- 900nA in power down
- 26µA in standby-I
- On chip voltage regulator
- 1.9 to 3.6V supply range
- Enhanced ShockBurst™
- Automatic packet handling
- Auto packet transaction handling
- 6 data pipe MultiCeiver™
- Drop-in compatibility with nRF24L01
- On-air compatible in 250kbps and 1Mbps with nRF2401A, nRF2402, nRF24E1 and nRF24E2
- Low cost BOM
- ±60ppm 16MHz crystal
- 5V tolerant inputs
- Compact 20-pin 4x4mm QFN package

Applications

- Wireless PC Peripherals
- Mouse, keyboards and remotes
- 3-in-1 desktop bundles
- Advanced Media center remote controls
- VoIP headsets
- Game controllers
- Sports watches and sensors
- RF remote controls for consumer electronics
- Home and commercial automation
- Ultra low power sensor networks
- Active RFID
- Asset tracking systems
- Toys

All rights reserved.
Reproduction in whole or in part is prohibited without the prior written permission of the copyright holder.
March 2008



Liability disclaimer

Nordic Semiconductor ASA reserves the right to make changes without further notice to the product to improve reliability, function or design. Nordic Semiconductor ASA does not assume any liability arising out of the application or use of any product or circuits described herein.

All application information is advisory and does not form part of the specification.

Limiting values

Stress above one or more of the limiting values may cause permanent damage to the device. These are stress ratings only and operation of the device at these or at any other conditions above those given in the specifications are not implied. Exposure to limiting values for extended periods may affect device reliability.

Life support applications

These products are not designed for use in life support appliances, devices, or systems where malfunction of these products can reasonably be expected to result in personal injury. Nordic Semiconductor ASA customers using or selling these products for use in such applications do so at their own risk and agree to fully indemnify Nordic Semiconductor ASA for any damages resulting from such improper use or sale.

Data sheet status	
Objective product specification	This product specification contains target specifications for product development.
Preliminary product specification	This product specification contains preliminary data; supplementary data may be published from Nordic Semiconductor ASA later.
Product specification	This product specification contains final product specifications. Nordic Semiconductor ASA reserves the right to make changes at any time without notice in order to improve design and supply the best possible product.

Contact details

Visit www.nordicsemi.no for Nordic Semiconductor sales offices and distributors worldwide

Main office:

Otto Nielsens vei 12
7004 Trondheim
Phone: +47 72 89 89 00
Fax: +47 72 89 89 89
www.nordicsemi.no



Writing Conventions

This product specification follows a set of typographic rules that makes the document consistent and easy to read. The following writing conventions are used:

- Commands, bit state conditions, and register names are written in *Courier*.
- Pin names and pin signal conditions are written in **Courier bold**.
- Cross references are [underlined and highlighted in blue](#).

Revision History

Date	Version	Description
March 2008	1.0	

Attention!

Observe precaution for handling
Electrostatic Sensitive Device.





Contents

1	Introduction	7
1.1	Features	8
1.2	Block diagram	9
2	Pin Information	10
2.1	Pin assignment.....	10
2.2	Pin functions.....	11
3	Absolute maximum ratings	12
4	Operating conditions	13
5	Electrical specifications	14
5.1	Power consumption.....	14
5.2	General RF conditions	15
5.3	Transmitter operation	15
5.4	Receiver operation	16
5.5	Crystal specifications	19
5.6	DC characteristics	19
5.7	Power on reset	19
6	Radio Control	20
6.1	Operational Modes.....	20
6.1.1	State diagram	20
6.1.2	Power Down Mode	21
6.1.3	Standby Modes.....	21
6.1.4	RX mode.....	22
6.1.5	TX mode	22
6.1.6	Operational modes configuration.....	23
6.1.7	Timing Information.....	23
6.2	Air data rate.....	24
6.3	RF channel frequency	24
6.4	Received Power Detector measurements.....	24
6.5	PA control.....	25
6.6	RX/TX control	25
7	Enhanced ShockBurst™	26
7.1	Features	26
7.2	Enhanced ShockBurst™ overview	26
7.3	Enhanced Shockburst™ packet format.....	27
7.3.1	Preamble	27
7.3.2	Address	27
7.3.3	Packet Control Field	27
7.3.4	Payload.....	28
7.3.5	CRC (Cyclic Redundancy Check)	28
7.4	Automatic packet handling	28
7.4.1	Static and Dynamic Payload Length.....	29
7.4.2	Automatic packet assembly	29
7.4.3	Automatic packet validation	30
7.4.4	Automatic packet disassembly	30

7.5	Automatic packet transaction handling	31
7.5.1	Auto Acknowledgement	31
7.5.2	Auto Retransmission (ART)	31
7.6	Enhanced ShockBurst flowcharts	33
7.6.1	PTX operation	33
7.6.2	PRX operation	35
7.7	MultiCeiver™	37
7.8	Enhanced ShockBurst™ timing	40
7.9	Enhanced ShockBurst™ transaction diagram	42
7.9.1	Single transaction with ACK packet and interrupts	42
7.9.2	Single transaction with a lost packet	43
7.9.3	Single transaction with a lost ACK packet	43
7.9.4	Single transaction with ACK payload packet	44
7.9.5	Single transaction with ACK payload packet and lost packet	44
7.9.6	Two transactions with ACK payload packet and the first ACK packet lost	45
7.9.7	Two transactions where max retransmissions is reached	45
7.10	Compatibility with ShockBurst™	46
7.10.1	ShockBurst™ packet format	46
8	Data and Control Interface	47
8.1	Features	47
8.2	Functional description	47
8.3	SPI operation	47
8.3.1	SPI Commands	47
8.3.2	SPI timing	49
8.4	Data FIFO	52
8.5	Interrupt	53
9	Register Map	54
9.1	Register map table	54
10	Peripheral RF Information	61
10.1	Antenna output	61
10.2	Crystal oscillator	61
10.3	nRF24L01+ crystal sharing with an MCU	61
10.3.1	Crystal parameters	61
10.3.2	Input crystal amplitude and current consumption	61
10.4	PCB layout and decoupling guidelines	62
11	Application example	63
11.1	PCB layout examples	64
12	Mechanical specifications	68
13	Ordering information	70
13.1	Package marking	70
13.2	Abbreviations	70
13.3	Product options	70
13.3.1	RF silicon	70
13.3.2	Development tools	70
14	Glossary of Terms	71



Appendix A - Enhanced ShockBurst™ - Configuration and Communication Example	72
Enhanced ShockBurst™ Transmitting Payload	72
Enhanced ShockBurst™ Receive Payload	73
Appendix B - Configuration for compatibility with nRF24XX.....	74
Appendix C - Constant carrier wave output for testing.....	75
Configuration	75

1 Introduction

The nRF24L01+ is a single chip 2.4GHz transceiver with an embedded baseband protocol engine (Enhanced ShockBurst™), suitable for ultra low power wireless applications. The nRF24L01+ is designed for operation in the world wide ISM frequency band at 2.400 - 2.4835GHz.

To design a radio system with the nRF24L01+, you simply need an MCU (microcontroller) and a few external passive components.

You can operate and configure the nRF24L01+ through a Serial Peripheral Interface (SPI). The register map, which is accessible through the SPI, contains all configuration registers in the nRF24L01+ and is accessible in all operation modes of the chip.

The embedded baseband protocol engine (Enhanced ShockBurst™) is based on packet communication and supports various modes from manual operation to advanced autonomous protocol operation. Internal FIFOs ensure a smooth data flow between the radio front end and the system's MCU. Enhanced ShockBurst™ reduces system cost by handling all the high speed link layer operations.

The radio front end uses GFSK modulation. It has user configurable parameters like frequency channel, output power and air data rate. nRF24L01+ supports an air data rate of 250 kbps, 1 Mbps and 2Mbps. The high air data rate combined with two power saving modes make the nRF24L01+ very suitable for ultra low power designs.

nRF24L01+ is drop-in compatible with nRF24L01 and on-air compatible with nRF2401A, nRF2402, nRF24E1 and nRF24E2. Intermodulation and wideband blocking values in nRF24L01+ are much improved in comparison to the nRF24L01 and the addition of internal filtering to nRF24L01+ has improved the margins for meeting RF regulatory standards.

Internal voltage regulators ensure a high Power Supply Rejection Ratio (PSRR) and a wide power supply range.



1.1 Features

Features of the nRF24L01+ include:

- Radio
 - ▶ Worldwide 2.4GHz ISM band operation
 - ▶ 126 RF channels
 - ▶ Common RX and TX interface
 - ▶ GFSK modulation
 - ▶ 250kbps, 1 and 2Mbps air data rate
 - ▶ 1MHz non-overlapping channel spacing at 1Mbps
 - ▶ 2MHz non-overlapping channel spacing at 2Mbps
- Transmitter
 - ▶ Programmable output power: 0, -6, -12 or -18dBm
 - ▶ 11.3mA at 0dBm output power
- Receiver
 - ▶ Fast AGC for improved dynamic range
 - ▶ Integrated channel filters
 - ▶ 13.5mA at 2Mbps
 - ▶ -82dBm sensitivity at 2Mbps
 - ▶ -85dBm sensitivity at 1Mbps
 - ▶ -94dBm sensitivity at 250kbps
- RF Synthesizer
 - ▶ Fully integrated synthesizer
 - ▶ No external loop filter, VCO varactor diode or resonator
 - ▶ Accepts low cost ± 60 ppm 16MHz crystal
- Enhanced ShockBurst™
 - ▶ 1 to 32 bytes dynamic payload length
 - ▶ Automatic packet handling
 - ▶ Auto packet transaction handling
 - ▶ 6 data pipe MultiCeiver™ for 1:6 star networks
- Power Management
 - ▶ Integrated voltage regulator
 - ▶ 1.9 to 3.6V supply range
 - ▶ Idle modes with fast start-up times for advanced power management
 - ▶ 26 μ A Standby-I mode, 900nA power down mode
 - ▶ Max 1.5ms start-up from power down mode
 - ▶ Max 130 μ s start-up from standby-I mode
- Host Interface
 - ▶ 4-pin hardware SPI
 - ▶ Max 10Mbps
 - ▶ 3 separate 32 bytes TX and RX FIFOs
 - ▶ 5V tolerant inputs
- Compact 20-pin 4x4mm QFN package

1.2 Block diagram

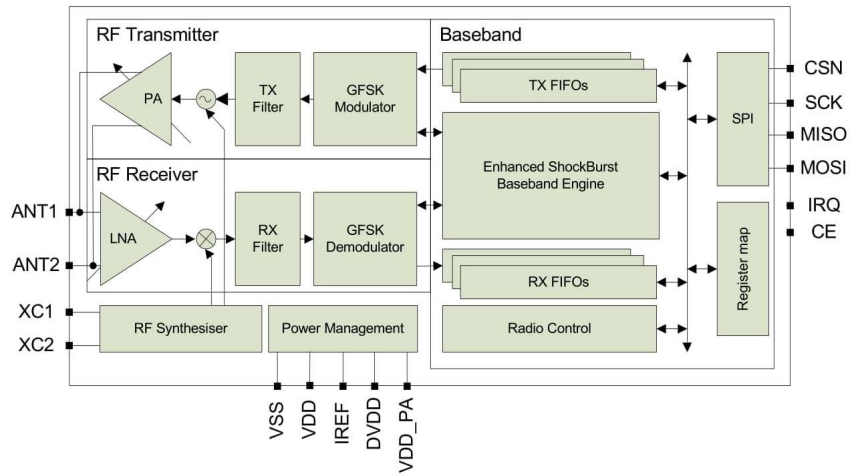


Figure 1. nRF24L01+ block diagram



2 Pin Information

2.1 Pin assignment

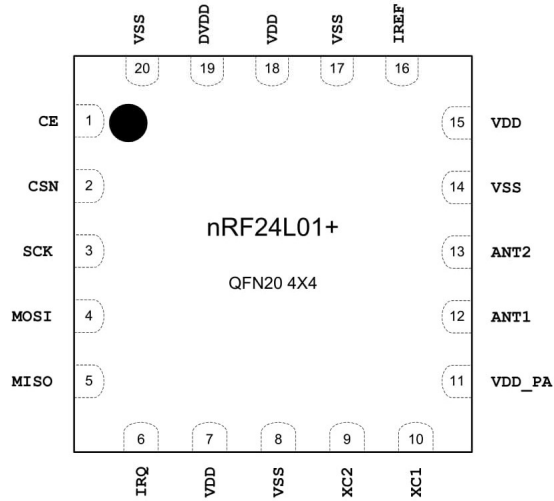


Figure 2. nRF24L01+ pin assignment (top view) for the QFN20 4x4 package

- **SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK).** These pins support SPI communication using the [SPI library](#).
- **LED: 13.** There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.

The Uno has 6 analog inputs, labeled A0 through A5, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and the [analogReference\(\)](#) function. Additionally, some pins have specialized functionality:

- **TWI: A4 or SDA pin and A5 or SCL pin.** Support TWI communication using the [Wire library](#).

There are a couple of other pins on the board:

- **AREF.** Reference voltage for the analog inputs. Used with [analogReference\(\)](#).
- **Reset.** Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

See also the [mapping between Arduino pins and ATmega328 ports](#). The mapping for the Atmega8, 168, and 328 is identical.

Communication

The Arduino Uno has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An ATmega16U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The '16U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, [on Windows, a .inf file is required](#). The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the Arduino board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer (but not for serial communication on pins 0 and 1).

A [SoftwareSerial library](#) allows for serial communication on any of the Uno's digital pins.

The ATmega328 also supports I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus; see the [documentation](#) for details. For SPI communication, use the [SPI library](#).

Programming

The Arduino Uno can be programmed with the Arduino software ([download](#)). Select "Arduino Uno from the **Tools > Board** menu (according to the microcontroller on your board). For details, see the [reference](#) and [tutorials](#).

The ATmega328 on the Arduino Uno comes preburned with a [bootloader](#) that allows you to upload new code to it without the use of an external hardware programmer. It communicates using the original STK500 protocol ([reference](#), [C header files](#)).

You can also bypass the bootloader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header; see [these instructions](#) for details.

The ATmega16U2 (or 8U2 in the rev1 and rev2 boards) firmware source code is available. The ATmega16U2/8U2 is loaded with a DFU bootloader, which can be activated by:

- On Rev1 boards: connecting the solder jumper on the back of the board (near the map of Italy) and then resetting the 8U2.
- On Rev2 or later boards: there is a resistor that pulling the 8U2/16U2 HWB line to ground, making it easier to put into DFU mode.

You can then use [Atmel's FLIP software](#) (Windows) or the [DFU programmer](#) (Mac OS X and Linux) to load a new firmware. Or you can use the ISP header with an external programmer (overwriting the DFU bootloader). See [this user-contributed tutorial](#) for more information.

Automatic (Software) Reset

Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Clock Speed	16 MHz

Schematic & Reference Design

EAGLE files: [arduino-uno-Rev3-reference-design.zip](#) (NOTE: works with Eagle 6.0 and newer)

Schematic: [arduino-uno-Rev3-schematic.pdf](#)

Note: The Arduino reference design can use an Atmega8, 168, or 328, Current models use an ATmega328, but an Atmega8 is shown in the schematic for reference. The pin configuration is identical on all three processors.

Power

The Arduino Uno can be powered via the USB connection or with an external power supply. The power source is selected automatically.

External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector.

The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

The power pins are as follows:

- **VIN.** The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.
- **5V.** This pin outputs a regulated 5V from the regulator on the board. The board can be supplied with power either from the DC power jack (7 - 12V), the USB connector (5V), or the VIN pin of the board (7-12V). Supplying voltage via the 5V or 3.3V pins bypasses the regulator, and can damage your board. We don't advise it.
- **3V3.** A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.
- **GND.** Ground pins.

Memory

The ATmega328 has 32 KB (with 0.5 KB used for the bootloader). It also has 2 KB of SRAM and 1 KB of EEPROM (which can be read and written with the [EEPROM library](#)).

Input and Output

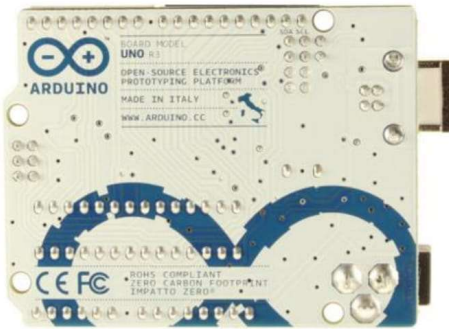
Each of the 14 digital pins on the Uno can be used as an input or output, using [pinMode\(\)](#), [digitalWrite\(\)](#), and [digitalRead\(\)](#) functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 kOhms. In addition, some pins have specialized functions:

- **Serial: 0 (RX) and 1 (TX).** Used to receive (RX) and transmit (TX) TTL serial data. These pins are connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip.
- **External Interrupts: 2 and 3.** These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the [attachInterrupt\(\)](#) function for details.
- **PWM: 3, 5, 6, 9, 10, and 11.** Provide 8-bit PWM output with the [analogWrite\(\)](#) function.

Arduino Uno



Arduino Uno R3 Front



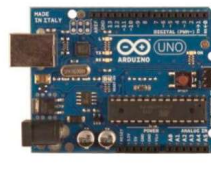
Arduino Uno R3 Back



Arduino Uno R2 Front



Arduino Uno SMD



Arduino Uno Front



Arduino Uno Back

Overview

The Arduino Uno is a microcontroller board based on the ATmega328 ([datasheet](#)). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.

The Uno differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega16U2 (Atmega8U2 up to version R2) programmed as a USB-to-serial converter.

| [Revision 2](#) of the Uno board has a resistor pulling the 8U2 HWB line to ground, making it easier to put into [DFU mode](#).

| [Revision 3](#) of the board has the following new features:

- 1.0 pinout: added SDA and SCL pins that are near to the AREF pin and two other new pins placed near to the RESET pin, the IOREF that allow the shields to adapt to the voltage provided from the board. In future, shields will be compatible both with the board that use the AVR, which operate with 5V and with the Arduino Due that operate with 3.3V. The second one is a not connected pin, that is reserved for future purposes.
- Stronger RESET circuit.
- Atmega 16U2 replace the 8U2.

"Uno" means one in Italian and is named to mark the upcoming release of Arduino 1.0. The Uno and version 1.0 will be the reference versions of Arduino, moving forward. The Uno is the latest in a series of USB Arduino boards, and the reference model for the Arduino platform; for a comparison with previous versions, see the [index of Arduino boards](#).

Summary

Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recommended)	7-12V

Rather than requiring a physical press of the reset button before an upload, the Arduino Uno is designed in a way that allows it to be reset by software running on a connected computer. One of the hardware flow control lines (DTR) of the ATmega8U2/16U2 is connected to the reset line of the ATmega328 via a 100 nanofarad capacitor. When this line is asserted (taken low), the reset line drops long enough to reset the chip. The Arduino software uses this capability to allow you to upload code by simply pressing the upload button in the Arduino environment. This means that the bootloader can have a shorter timeout, as the lowering of DTR can be well-coordinated with the start of the upload. This setup has other implications. When the Uno is connected to either a computer running Mac OS X or Linux, it resets each time a connection is made to it from software (via USB). For the following half-second or so, the bootloader is running on the Uno. While it is programmed to ignore malformed data (i.e. anything besides an upload of new code), it will intercept the first few bytes of data sent to the board after a connection is opened. If a sketch running on the board receives one-time configuration or other data when it first starts, make sure that the software with which it communicates waits a second after opening the connection and before sending this data. The Uno contains a trace that can be cut to disable the auto-reset. The pads on either side of the trace can be soldered together to re-enable it. It's labeled "RESET-EN". You may also be able to disable the auto-reset by connecting a 110 ohm resistor from 5V to the reset line; see [this forum thread](#) for details.

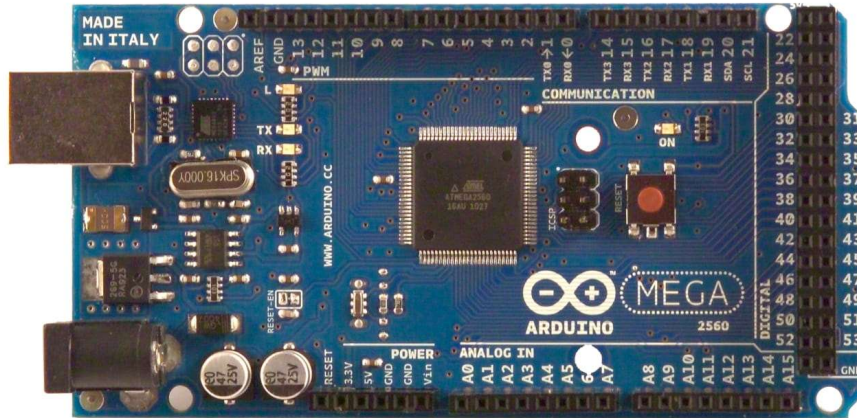
USB Overcurrent Protection

The Arduino Uno has a resettable polyfuse that protects your computer's USB ports from shorts and overcurrent. Although most computers provide their own internal protection, the fuse provides an extra layer of protection. If more than 500 mA is applied to the USB port, the fuse will automatically break the connection until the short or overload is removed.

Physical Characteristics

The maximum length and width of the Uno PCB are 2.7 and 2.1 inches respectively, with the USB connector and power jack extending beyond the former dimension. Four screw holes allow the board to be attached to a surface or case. Note that the distance between digital pins 7 and 8 is 160 mil (0.16"), not an even multiple of the 100 mil spacing of the other pins.

Arduino MEGA 2560

Product Overview

The Arduino Mega 2560 is a microcontroller board based on the ATmega2560 ([datasheet](#)). It has 54 digital input/output pins (of which 14 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. The Mega is compatible with most shields designed for the Arduino Duemilanove or Diecimila.

Index

Technical Specifications	Page 2
How to use Arduino Programming Enviroment, Basic Tutorials	Page 6
Terms & Conditions	Page 7
Enviromental Policies half sqm of green via Impatto Zero®	Page 7



radiospares

RADIONICS



Technical Specification

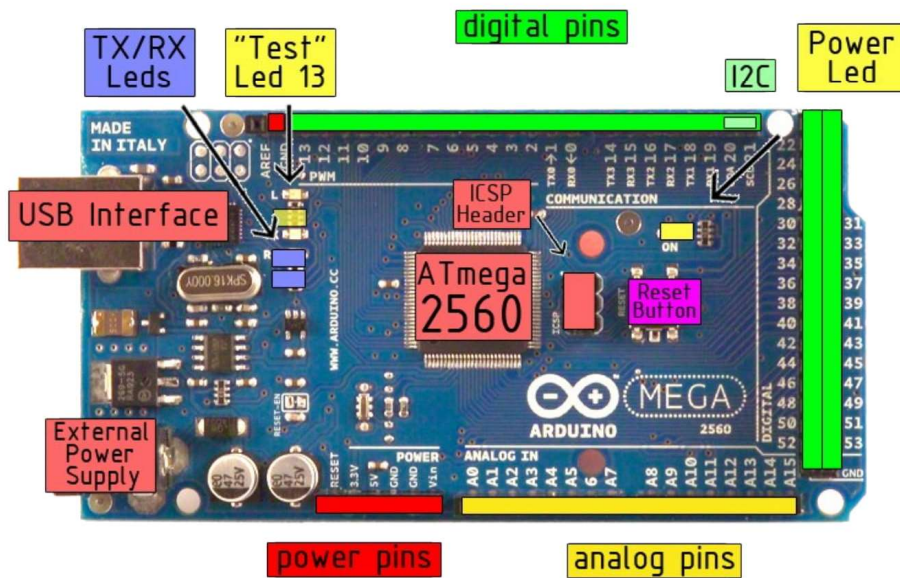


EAGLE files: [arduino-mega2560-reference-design.zip](#) Schematic: [arduino-mega2560-schematic.pdf](#)

Summary

Microcontroller	ATmega2560
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	54 (of which 14 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	256 KB of which 8 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz

the board



radiospares

RADIONICS



Power

The Arduino Mega2560 can be powered via the USB connection or with an external power supply. The power source is selected automatically. External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector.

The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

The Mega2560 differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega8U2 programmed as a USB-to-serial converter.

The power pins are as follows:

- **VIN.** The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.
- **5V.** The regulated power supply used to power the microcontroller and other components on the board. This can come either from VIN via an on-board regulator, or be supplied by USB or another regulated 5V supply.
- **3V3.** A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.
- **GND.** Ground pins.

Memory

The ATmega2560 has 256 KB of flash memory for storing code (of which 8 KB is used for the bootloader), 8 KB of SRAM and 4 KB of EEPROM (which can be read and written with the [EEPROM library](#)).

Input and Output

Each of the 54 digital pins on the Mega can be used as an input or output, using [pinMode\(\)](#), [digitalWrite\(\)](#), and [digitalRead\(\)](#) functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 kOhms. In addition, some pins have specialized functions:

- **Serial: 0 (RX) and 1 (TX); Serial 1: 19 (RX) and 18 (TX); Serial 2: 17 (RX) and 16 (TX); Serial 3: 15 (RX) and 14 (TX).** Used to receive (RX) and transmit (TX) TTL serial data. Pins 0 and 1 are also connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip.
- **External Interrupts: 2 (interrupt 0), 3 (interrupt 1), 18 (interrupt 5), 19 (interrupt 4), 20 (interrupt 3), and 21 (interrupt 2).** These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the [attachInterrupt\(\)](#) function for details.
- **PWM: 0 to 13.** Provide 8-bit PWM output with the [analogWrite\(\)](#) function.
- **SPI: 50 (MISO), 51 (MOSI), 52 (SCK), 53 (SS).** These pins support SPI communication, which, although provided by the underlying hardware, is not currently included in the Arduino language. The SPI pins are also broken out on the ICSP header, which is physically compatible with the Duemilanove and Diecimila.
- **LED: 13.** There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.
- **I²C: 20 (SDA) and 21 (SCL).** Support I²C (TWI) communication using the [Wire library](#) (documentation on the Wiring website). Note that these pins are not in the same location as the I²C pins on the Duemilanove.

The Mega2560 has 16 analog inputs, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though it is possible to change the upper end of their range using the AREF pin and [analogReference\(\)](#) function.

There are a couple of other pins on the board:

- **AREF.** Reference voltage for the analog inputs. Used with [analogReference\(\)](#).
- **Reset.** Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.



RADIOSPARES

RADIONICS



Communication

The Arduino Mega2560 has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega2560 provides four hardware UARTs for TTL (5V) serial communication. An ATmega8U2 on the board channels one of these over USB and provides a virtual com port to software on the computer (Windows machines will need a .inf file, but OSX and Linux machines will recognize the board as a COM port automatically). The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the board. The RX and TX LEDs on the board will flash when data is being transmitted via the ATmega8U2 chip and USB connection to the computer (but not for serial communication on pins 0 and 1).

A [SoftwareSerial library](#) allows for serial communication on any of the Mega's digital pins.

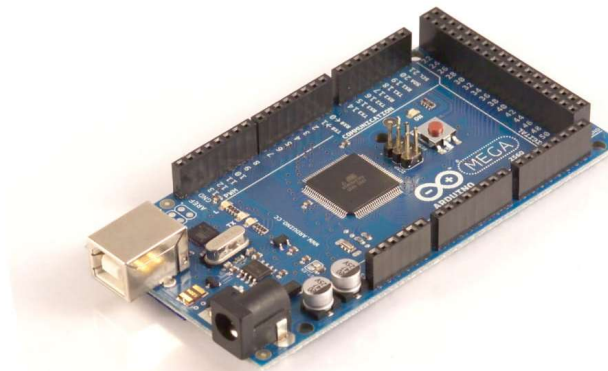
The ATmega2560 also supports I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus; see the [documentation on the Wiring website](#) for details. To use the SPI communication, please see the ATmega2560 datasheet.

Programming

The Arduino Mega2560 can be programmed with the Arduino software ([download](#)). For details, see the [reference](#) and [tutorials](#).

The ATmega2560 on the Arduino Mega comes preburned with a [bootloader](#) that allows you to upload new code to it without the use of an external hardware programmer. It communicates using the original STK500 protocol ([reference](#), [C header files](#)).

You can also bypass the bootloader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header; see [these instructions](#) for details.



radiospares

RADIONICS



Automatic (Software) Reset

Rather than requiring a physical press of the reset button before an upload, the Arduino Mega2560 is designed in a way that allows it to be reset by software running on a connected computer. One of the hardware flow control lines (DTR) of the ATmega8U2 is connected to the reset line of the ATmega2560 via a 100 nanofarad capacitor. When this line is asserted (taken low), the reset line drops long enough to reset the chip. The Arduino software uses this capability to allow you to upload code by simply pressing the upload button in the Arduino environment. This means that the bootloader can have a shorter timeout, as the lowering of DTR can be well-coordinated with the start of the upload.

This setup has other implications. When the Mega2560 is connected to either a computer running Mac OS X or Linux, it resets each time a connection is made to it from software (via USB). For the following half-second or so, the bootloader is running on the Mega2560. While it is programmed to ignore malformed data (i.e. anything besides an upload of new code), it will intercept the first few bytes of data sent to the board after a connection is opened. If a sketch running on the board receives one-time configuration or other data when it first starts, make sure that the software with which it communicates waits a second after opening the connection and before sending this data.

The Mega contains a trace that can be cut to disable the auto-reset. The pads on either side of the trace can be soldered together to re-enable it. It's labeled "RESET-EN". You may also be able to disable the auto-reset by connecting a 110 ohm resistor from 5V to the reset line; see [this forum thread](#) for details.

USB Overcurrent Protection

The Arduino Mega has a resettable polyfuse that protects your computer's USB ports from shorts and overcurrent. Although most computers provide their own internal protection, the fuse provides an extra layer of protection. If more than 500 mA is applied to the USB port, the fuse will automatically break the connection until the short or overload is removed.

Physical Characteristics and Shield Compatibility

The maximum length and width of the Mega PCB are 4 and 2.1 inches respectively, with the USB connector and power jack extending beyond the former dimension. Three screw holes allow the board to be attached to a surface or case. Note that the distance between digital pins 7 and 8 is 160 mil (0.16"), not an even multiple of the 100 mil spacing of the other pins.

The Mega is designed to be compatible with most shields designed for the Diecimila or Duemilanove. Digital pins 0 to 13 (and the adjacent AREF and GND pins), analog inputs 0 to 5, the power header, and ICSP header are all in equivalent locations. Further the main UART (serial port) is located on the same pins (0 and 1), as are external interrupts 0 and 1 (pins 2 and 3 respectively). SPI is available through the ICSP header on both the Mega and Duemilanove / Diecimila. **Please note that I²C is not located on the same pins on the Mega (20 and 21) as the Duemilanove / Diecimila (analog inputs 4 and 5).**

**radiospares****RADIONICS**

How to use Arduino



Arduino can sense the environment by receiving input from a variety of sensors and can affect its surroundings by controlling lights, motors, and other actuators. The microcontroller on the board is programmed using the [Arduino programming language](#) (based on [Wiring](#)) and the Arduino development environment (based on [Processing](#)). Arduino projects can be stand-alone or they can communicate with software on running on a computer (e.g. Flash, Processing, MaxMSP).

Arduino is a cross-platform program. You'll have to follow different instructions for your personal OS. Check on the [Arduino site](#) for the latest instructions. <http://arduino.cc/en/Guide/HomePage>

Linux Install

Windows Install

Mac Install

Once you have downloaded/unzipped the arduino IDE, you can Plug the Arduino to your PC via USB cable.

Blink led

Now you're actually ready to "burn" your first program on the arduino board. To select "blink led", the physical translation of the well known programming "hello world", select

File>Sketchbook>
Arduino-0017>Examples>
Digital>Blink

Once you have your sketch you'll see something very close to the screenshot on the right.

In **Tools>Board** select MEGA

Now you have to go to **Tools>SerialPort** and select the right serial port, the one arduino is attached to.

```

Blink | Arduino 0017
File Edit Sketch Tools Help
Blink $
int ledPin = 13; // LED connected to digital pin 13

// The setup() method runs once, when the sketch starts

void setup() {
  // initialize the digital pin as an output:
  pinMode(ledPin, OUTPUT);
}

// the loop() method runs over and over again,
// as long as the Arduino has power

void loop()
{
  digitalWrite(ledPin, HIGH); // set the LED on
  delay(1000); // wait for a second
  digitalWrite(ledPin, LOW); // set the LED off
  delay(1000); // wait for a second
}
    
```

Press Compile button (to check for errors)

Upload

TX RX Flashing

Blinking Led!



radiospares

RADIONICS



Terms & Conditions



1. Warranties

1.1 The producer warrants that its products will conform to the Specifications. This warranty lasts for one (1) years from the date of the sale. The producer shall not be liable for any defects that are caused by neglect, misuse or mistreatment by the Customer, including improper installation or testing, or for any products that have been altered or modified in any way by a Customer. Moreover, The producer shall not be liable for any defects that result from Customer's design, specifications or instructions for such products. Testing and other quality control techniques are used to the extent the producer deems necessary.

1.2 If any products fail to conform to the warranty set forth above, the producer's sole liability shall be to replace such products. The producer's liability shall be limited to products that are determined by the producer not to conform to such warranty. If the producer elects to replace such products, the producer shall have a reasonable time to replacements. Replaced products shall be warranted for a new full warranty period.

1.3 EXCEPT AS SET FORTH ABOVE, PRODUCTS ARE PROVIDED "AS IS" AND "WITH ALL FAULTS." THE PRODUCER DISCLAIMS ALL OTHER WARRANTIES, EXPRESS OR IMPLIED, REGARDING PRODUCTS, INCLUDING BUT NOT LIMITED TO, ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE

1.4 Customer agrees that prior to using any systems that include the producer products, Customer will test such systems and the functionality of the products as used in such systems. The producer may provide technical, applications or design advice, quality characterization, reliability data or other services. Customer acknowledges and agrees that providing these services shall not expand or otherwise alter the producer's warranties, as set forth above, and no additional obligations or liabilities shall arise from the producer providing such services.

1.5 The Arduino™ products are not authorized for use in safety-critical applications where a failure of the product would reasonably be expected to cause severe personal injury or death. Safety-Critical Applications include, without limitation, life support devices and systems, equipment or systems for the operation of nuclear facilities and weapons systems. Arduino™ products are neither designed nor intended for use in military or aerospace applications or environments and for automotive applications or environment. Customer acknowledges and agrees that any such use of Arduino™ products which is solely at the Customer's risk, and that Customer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

1.6 Customer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products and any use of Arduino™ products in Customer's applications, notwithstanding any applications-related information or support that may be provided by the producer.

2. Indemnification

The Customer acknowledges and agrees to defend, indemnify and hold harmless the producer from and against any and all third-party losses, damages, liabilities and expenses it incurs to the extent directly caused by: (i) an actual breach by a Customer of the representation and warranties made under this terms and conditions or (ii) the gross negligence or willful misconduct by the Customer.

3. Consequential Damages Waiver

In no event the producer shall be liable to the Customer or any third parties for any special, collateral, indirect, punitive, incidental, consequential or exemplary damages in connection with or arising out of the products provided hereunder, regardless of whether the producer has been advised of the possibility of such damages. This section will survive the termination of the warranty period.

4. Changes to specifications

The producer may make changes to specifications and product descriptions at any time, without notice. The Customer must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." The producer reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The product information on the Web Site or Materials is subject to change without notice. Do not finalize a design with this information.



Environmental Policies



The producer of Arduino™ has joined the Impatto Zero® policy of LifeGate.it. For each Arduino board produced is created / looked after half squared Km of Costa Rica's forest's.



radiospares

RADIONICS



Bibliografía

- [1] A. Ollero Baturone, «Manipuladores y robots móviles,» Marcombo, Barcelona, 2001.
- [2] A. León García, «Sistema de localización de robots en interiores,» 2018.
- [3] F. Torres, J. Pomares, P. Gil, S. T. Puente y R. Aracil, «Robots y Sistemas Sensoriales,» Prentice-Hall, Madrid, 2002.
- [4] Arduino, «Arduino MEGA 2560,» [En línea]. Available: <http://arduino.cl/arduino-mega-2560/>.
- [5] Wikipedia, «Arduino IDE,» [En línea]. Available: https://en.wikipedia.org/wiki/Arduino_IDE.
- [6] Wikipedia, «Arduino,» [En línea]. Available: <https://es.wikipedia.org/wiki/Arduino>.
- [7] A. Ward, P. Steggles, R. Curwen, P. Webster, M. Addlesee, J. Newman, P. Osborn y S. Hodges, «The Bat Ultrasonic Location System,» [En línea]. Available: <https://www.cl.cam.ac.uk/research/dtg/attarchive/bat/>.
- [8] P. C. García, «Sistema de localización en interiores por ultrasonido,» *Avances en ciencias e ingenierías*, p. 6, 2012.
- [9] E. M. García, «Técnicas de Localización en Redes Inalámbricas de Sensores,» [En línea]. Available: https://www.researchgate.net/publication/228705728_Tecnicas_de_Localizacion_en_Redess_Inalambricas_de_Sensores.
- [10] J. Roa, A. Jiménez, F. Seco, C. Prieto, J. Ealo, F. Ramos y J. Guevara, «Distribución optimizada de balizas para sistemas de localización fundamentados en Trilateración Elíptica e Hiperbólica,» [En línea]. Available: <http://www.car.upm-csic.es/lopsi/static/publicaciones/Congreso%20Internacional/MAEBRoa2007.pdf>.
- [11] F. Seco Granja, «Sistema de localización en interiores basados en radiofrecuencia,» [En línea]. Available: <http://www.car.upm-csic.es/lopsi/static/publicaciones/docencia/Apuntes%20RF-LPS.pdf>.
- [12] C. Sánchez, «Shakey, el primer robot inteligente de la historia y el abuelo del coche autónomo,» *eldiario.es*.
- [13] «Repetier Software,» [En línea]. Available: <https://www.repetier.com/>.
- [14] D. Gómez Bertoli, «Estudio, implementación y análisis de métodos de trilateración para la localización de usuarios desde sus terminales móviles,» [En línea]. Available: <https://core.ac.uk/download/pdf/29403656.pdf>.
- [15] «Slic3r- Open source 3D printing toolbox,» [En línea]. Available: <https://slic3r.org/>.
- [16] «¿Cómo funcionan los dispositivos GPS? Trilateración vs Triangulación,» [En línea]. Available: <https://acolita.com/como-funcionan-los-dispositivos-gps-trilateracion-vs-triangulacion/>.
- [17] «Situm RTLS: Localización en interiores de Situm Technologies,» [En línea]. Available: <https://situm.es/es/blog/situm-rtls-localizacion-en-interiores-de-situm-technologies>.

