



# Trabajo de Fin de Grado

Grado en Ingeniería Informática

# Sistemas de localización de robots para interiores

*Robot locating systems for interiors*

Alberto Martínez Chincho

La Laguna, 21 de febrero de 2017

D. **Jonay Tomás Toledo Carrillo**, con N.I.F. 78.698.554-Y profesor Titular de Universidad adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como tutor

**C E R T I F I C A (N)**

Que la presente memoria titulada:

*“Sistemas de localización de robots para interiores.”*

ha sido realizada bajo su dirección por D. **Alberto Martínez Chincho**, con N.I.F. 45.733.569-R.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 4 de marzo de 2015.

# Agradecimientos

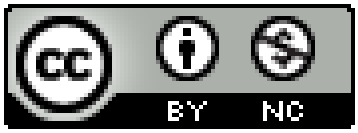
Siempre llega un momento en nuestra vida donde tenemos que cerrar una etapa para dejar paso a nuevas experiencias y nuevas oportunidades.

Con este proyecto llega el momento de cerrar una de las etapas más bonitas y duras de una persona, para dar paso a una nueva vida.

Como ingenieros tendremos muchos obstáculos y fracasos en nuestra vida pero tenemos que saber y aprender a levantarnos, siguiendo adelante, porque para encontrar el éxito primero debemos fracasar.

Gracias de corazón a mis padres, amigos, profesores y todas las personas que han estado y están a mi lado apoyándome y sacrificándose por mí.

# Licencia



## Resumen

*El objetivo de este trabajo ha sido desarrollar un sistema de localización de robots para interiores mediante sensores de bajo coste y módulos de radio-frecuencia. Estos se encargaran de establecer la comunicación entre el robot y los sensores cuya posición es conocida, además utilizaremos un algoritmo que se encargará de calcular la distancia del robots a dichos sensores para a continuación realizar los cálculos necesarios y estimar la posición en la que se encuentra dicho robot.*

*Para alcanzar esta tarea se han utilizado varios componentes electrónicos como sensores de ultrasonido y módulos de comunicación por radiofrecuencia. También hemos utilizado sistemas empotrados como Arduino y Raspberry Pi.*

*La localización se realizara mediante un algoritmo desarrollado en nuestro propio Arduino, siendo los sensores fijos los encargados de generar señales ultrasónicas sincronizadas a través de comunicaciones de radiofrecuencia y la comunicación para nuestro robot.*

*Por último, y para facilitar la comunicación del sistema de localización con nuestro robot se ha utilizado ROS.*



**Palabras clave:** Arduino, Raspberry Pi, Ultrasonido, Radio-frecuencia, Localización, Robots, Arduino Ros, Algoritmos.

## Abstract

The objective of this work was to develop a system for robots localization indoor, using low cost components and a certain algorithm, which will be responsible for calculating the distance of the robots to several fixed sensors to then perform the necessary calculations and estimate the situation in which it is.

Several electronic components such as ultrasound sensors and radio-frequency communication modules have been used to achieve this task. We have also used embedded systems like Arduino and Raspberry Pi.

The location will be made using an algorithm developed in our own Arduino, whereas in the fixed sensors we will have to take charge of sending the ultrasonic signal and the communication for our robot.

In order to simplify robot communication, ROS is used.



***Keywords:*** *Arduino, Raspberry Pi, Ultrasound, Radio-frequency, Localization, Robots, Arduino Ros, Algorithms.*

# Índice General

# Índice de figuras

# Índice de tablas

# Capítulo 1. Introducción

En este capítulo describimos brevemente la evolución del proyecto y la motivación para realizarlo. También se expondrá un breve resumen de los capítulos de nuestra memoria.

## 1.1 Introducción

Alguna vez se han preguntado como un robot es capaz de moverse libremente de manera autónoma sin producir ningún desperfecto. En este proyecto, utilizaremos los sistemas empotrados para dotar a un robot de un sistema de localización para interiores.

Los sistemas de localización son sistemas que nos permiten identificar o rastrear automáticamente la localización de objetos o personas en tiempo real, como es el caso de los GPS. Un sistema de localización GPS tiene una precisión de hasta unos pocos centímetros si se utiliza un GPS diferencial, pero los convencionales tienen una precisión de unos pocos metros.

El GPS funciona mediante una red de 24 satélites en órbita, con trayectorias sincronizadas para cubrir toda la superficie de la Tierra. Cuando queremos determinar la posición, el receptor que se utiliza localiza automáticamente como mínimo tres satélites de la red, de los que recibe unas señales indicando la identificación y la hora del reloj de cada uno de ellos. En base a estas señales, el dispositivo sincroniza el reloj del GPS y calcula el tiempo que tardan en llegar las señales al equipo, y de tal modo mide la distancia al satélite mediante triangulación.

La idea principal del proyecto esta basada en el funcionamiento de un sistema de localización GPS, pero para entornos cerrados o interiores. Por ellos, no podemos utilizar un GPS, ya que en primer lugar la precisión tendría que ser de unos metros y nosotros al encontrarnos en interiores necesitamos una precisión de centímetros. Pero no solo eso, sino que además un sistema de localización como este necesita de satélites y componentes externos a nosotros.

El sistema que vamos a desarrollar está orientado a la utilización de robot en almacenes, casas o naves industriales, es decir, entornos donde

podamos estar emisor y receptor continuamente en contacto, ya que la forma de distribuir nuestro sistema sería tan sencilla como la colocación de tres sensores fijos en cualquier punto del entorno y los receptores irían colocados en nuestro robot. Así se empezará a realizar la comunicación y a obtener en cada momento cual es la posición del mismo.

Si estuviéramos en exteriores esto sería fácil ya que podríamos utilizar un sistema de localización basado en los GPS, pero en interiores donde no es posible usar este tipo de sistemas, ya sea por el coste que conlleva o porque no son lo suficientemente efectivos, como hemos comentado anteriormente, debemos utilizar otro tipo, como por ejemplo el que estamos implementando baso en balizas ultrasónicas, el cual es capaz de generarnos la información suficiente para localizar un robot en el entorno que nos encontramos.

Para realizar esta tarea utilizamos diversos componentes como la placa Arduino Uno, este sistema deberá ir tanto en el robot, encargado de recibir tanto la señal como la comunicación y realizar los cálculos necesarios, como en los diversos sensores que distribuimos por la habitación y serán los encargados de enviar dicha señal y el mensaje de comunicación a nuestro robot.

Debemos construir diferentes algoritmos de procesamiento de señales y para realizar los cálculos necesarios según el sistema de localización que escojamos (explicado en el capítulo 3).

El fin de este proyecto es conseguir que nuestro robot se localice en entornos donde no es posible utilizar un sistema de localización baso en los GPS.

## 1.2 Componentes de la memoria

A continuación se realiza un breve resumen del contenido final de la memoria, exponiendo en cada punto un breve análisis de lo que contendrá cada uno de nuestros capítulos.

- **Capítulo 1:** INTRODUCCIÓN
  Descripción de los aspectos principales del proyecto, así como de los componentes utilizados y tipo de desarrollo expuesto. Breve descripción del contenido de la memoria.

- **Capítulo 2:** ANTECEDENTES
  Breve descripción de otros trabajos donde el fin sea el desarrollo de un sistema de localizar para robot.

- **Capítulo 3:** MARCO TEÓRICO
  Análisis sobre los métodos de localización existente y cuál es el más óptimo en nuestro caso. Sensores utilizados para nuestro método y cuáles son las ventajas y las desventajas de usarlos, explicación breve sobre la utilización de los módulos de radio-frecuencia. Por último, se hablara sobre el entorno de trabajo utilizado y donde se han realizado las pruebas, así como, los sistemas empotrados utilizados.

- **Capítulo 4:** MARCO PRÁCTICO
  Explicación de los pasos llevados a cabo hasta la obtención de nuestro objetivo. Se comentan todos los circuitos desarrollados, así como los diversos software para la comprobación del buen funcionamiento de los mismos y las pruebas realizadas para dicha comprobación. También se hablara de las dificultades encontradas en el camino y las soluciones a estos.

- **Capítulo 5:** CONCLUSIONES Y LINEAS FUTURAS
  Reflexión sobre los objetivos conseguidos y las pruebas finales realizadas. Además, algunas posibles formas de mejorar en un futuro nuestro proyecto.

- **Capítulo 6:** PRESUPUESTO
  Presupuesto acorde a los materiales usados y horas de trabajo realizadas durante el proyecto, donde se verá reflejado el coste que este supone.

- **Apéndice A:** CÓDIGOS
  Se presentaran los códigos fuente de los programas realizados en Arduino.

- **Apéndice B:** DOCUMENTACIÓN
  En este anexo encontraremos los datasheet de los componentes electrónicos utilizados en la construcción de nuestro hardware.

# Capítulo 2. Antecedentes

En este capítulo se expondrá el desarrollo de este mismo proyecto pero realizado de forma alternativa a la actual debido a lo que exponemos.

## 2.1 Sistemas de localización basado en balizas láseres

En etapas iniciales del proyecto, antes de decidir ambas partes que el proyecto se realizaría con balizas ultrasónicas, estaba orientado a la utilización de balizas láseres. El concepto general era el mismo, el realizar un sistema de localización.

A continuación se explica brevemente el funcionamiento del mismo:

En un primer lugar se realizó un circuito de recepción de la señal básico con un “TSOP1838”. Un TSOP1838 no es más que un sensor de infrarrojos que emplearemos para recibir la señal dada de otro circuito, su uso en muy frecuente en mandos a distancias. Otro de los circuitos montados sería el de emisión, el cual estaría encargado de enviar la señal laser mediante un LED.

Una vez montados ambos circuitos, se desarrollarían los códigos para establecer una comunicación. Estos realizaban la simple tarea de emitir una señal laser a través del LED y era recogida por nuestro sensor de infrarrojos. Debido a que el sensor solo detectaba señales a una frecuencia de 40kHz tuvimos que hacer uso de una librería propia de Arduino como es “TONE”, la cual nos permite generar una señal a una determinada frecuencia proporcionada por él desarrollador.

Una vez desarrollados los códigos se realizaron las pruebas para ver si esto era eficiente, descubriendo que esto solo funcionará de manera efectiva si la distancia a la que se encontraba receptor y emisor era inferior a un metro. Viendo que a medida que aumentábamos la distancia se iba perdiendo eficacia en la detección de la luz.

Además el ángulo de detección dependía demasiado de la distancia, dejando detectar a 90º en una distancia inferior al metro y cuando se superaba esa distancia no dejaba detectar a más de 30º.

Por lo que como primera conclusión se observa que a mayor distancia, menor ángulo y menor fuerza de llegada de la luz entre el emisor y el receptor.

Como solución a lo anteriormente mencionado, ya que fue desechado debido a que no podíamos trabajar con la frecuencia que se requiere generar, y por lo tanto para poder generar libremente una frecuencia se optó por añadir un nuevo componente receptor como el phototransistor “SFH309FA”. Este nuevo componente realizaba el mismo trabajo que el anterior pero con la ligera ventaja de que si dejaría generar un frecuencia de trabajo.

Además se cambiaron los primeros circuitos, en los que solo uno de ellos enviaba la señal y el otro la recibía, por otros dos nuevos con un funcionamiento diferente.

En este caso ambos circuitos recibirán y enviarán la señal de luz, es decir, el primer circuito encenderá su LED enviando luz al phototransistor del segundo circuito que una vez reciba la luz encenderá su LED, antes apagado, y enviara luz al circuito anterior y así sucesivamente.

Como en este circuito tuvimos que aplicar resistencias grandes la intensidad que circulaba hacia el LED era muy pequeña por lo que tuvimos que añadir un transistor “2N2222” para aumentar dicha intensidad.

Para comprobar que esta modificación funcionaba y antes de seguir desarrollando el proyecto, utilizamos el programa LTspice para simular el comportamiento. En este momento la corriente que se generaba era lineal y por lo tanto no se podía hacer que nuestro LED se apagara o encendiera.

Por lo tanto, se necesita que la corriente actuase como un pulso, añadiendo inversores a ambos circuito quedando como se muestra en la siguiente figura:

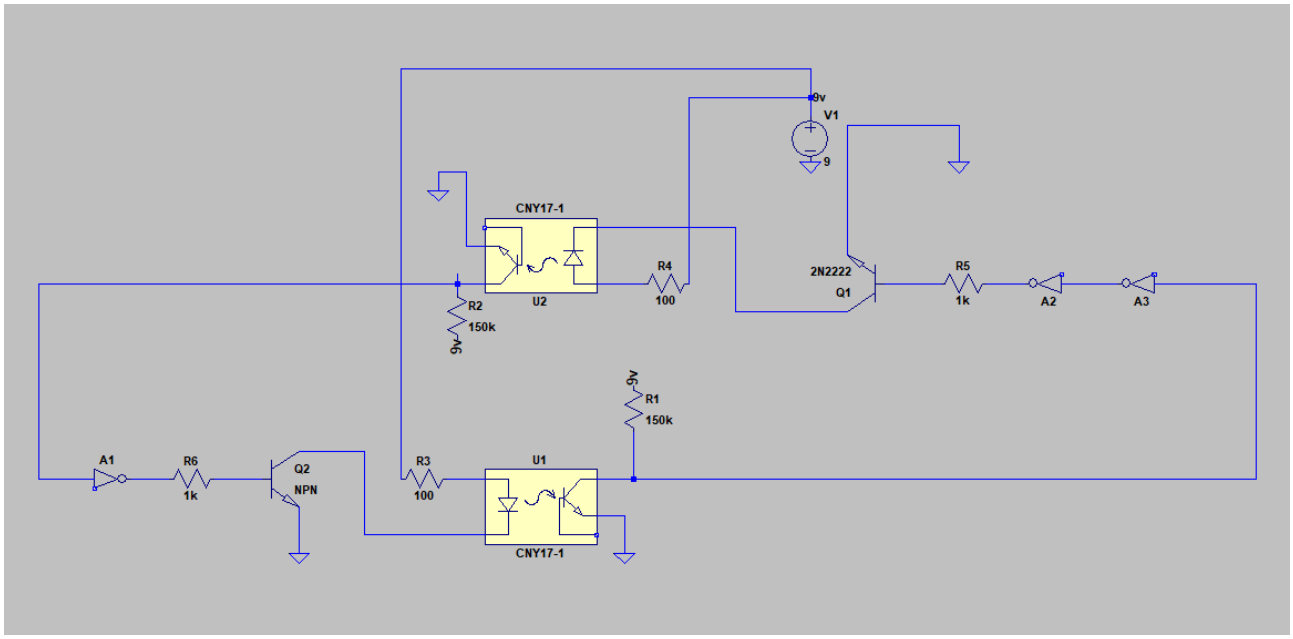



Figura 2.1: Circuito final frecuencia óptica

Una vez llegado a esta conclusión se realizaron unas pruebas, las cuales consistían en posicionar ambos circuitos en diferentes posiciones (ángulos y distancia) y observar que frecuencia se generaba.

Para ello nos ayudamos de un osciloscopio y volvimos a observar que el problema residía en que la distancia era muy pequeña en comparación a lo que se necesita, por lo que se empezó a cambiar los componentes que teníamos anteriormente por unos nuevos y quizás más efectivos, como inversores Schmitt-Trigger. Estos inversores nos permitían mantener un umbral de conmutación, es decir, cambiara su estado de salida cuando la tensión en su entrada supera dicho umbral, esto es lo que se necesitaba. Para hacer que la señal bajara necesitábamos un valor de casi 0v y a nosotros se nos generaba 1,2v por lo que nunca bajaría con un inversor convencional.

Se puede observar en la imagen siguiente los resultados obtenidos con el osciloscopio:

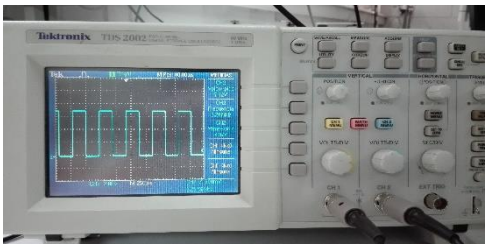



Figura 2.2: Osciloscopio pruebas con frecuencia generada

Al añadir esa pequeña modificación y realizar de nuevo las pruebas se observó que en la última salida del inversor el voltaje caía demasiado lo que seguía siendo insuficiente para que el led recibiera la intensidad necesaria para aumentar la luz y trabajar a distancias más largar. A continuación se muestran tres imágenes de cómo quedaría dichos circuitos finalmente:

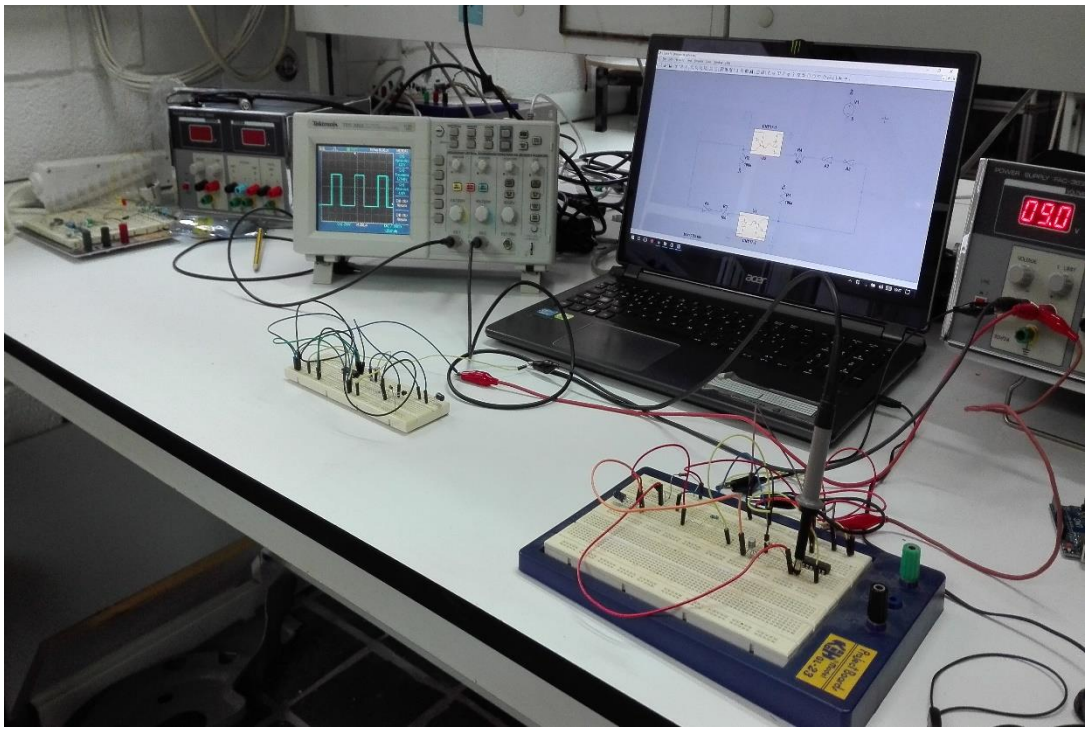


Figura 2.3: Montaje final real

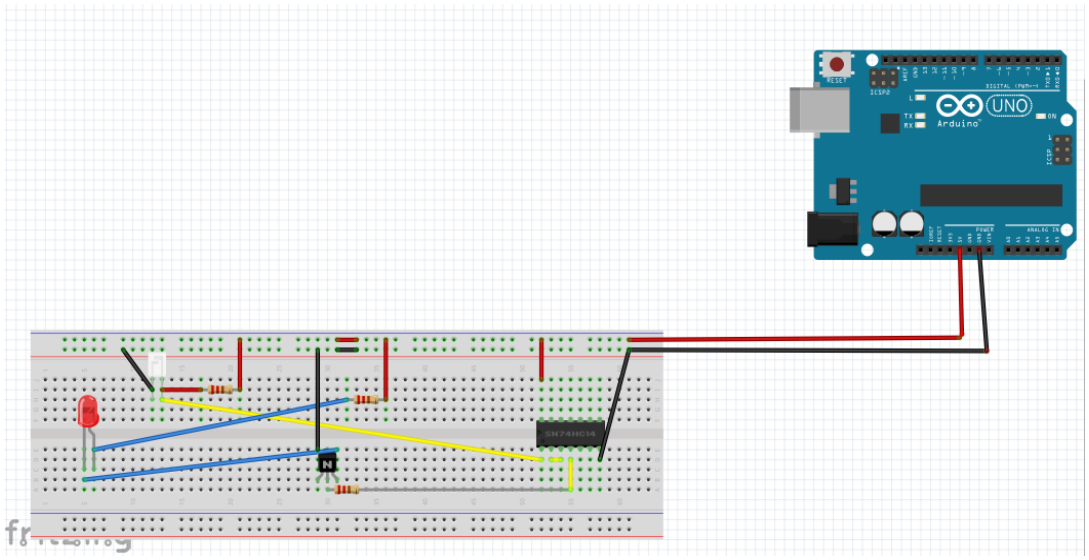



Figura 2.4: Circuito final (directo)

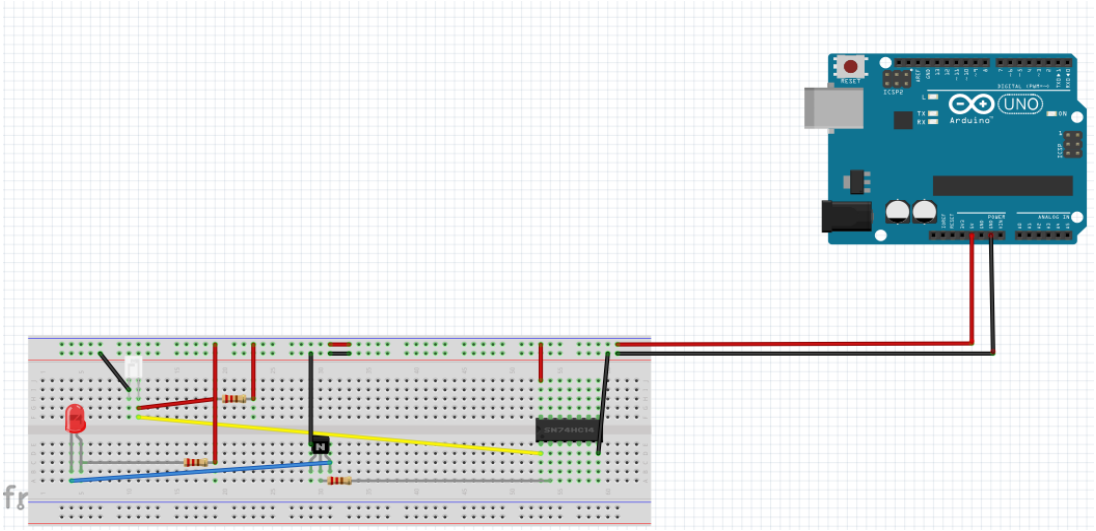


Figura 2.5: Circuito final (inverso)

Debido a todos los problemas comentados anteriormente tanto el tutor del proyecto como yo llegamos a la conclusión de que si seguíamos por este camino el proyecto se convertiría en algo más de electrónica que de informática.

Por lo tanto, para quitarnos estos problemas decidimos cambiar la manera de enviar la señal y utilizar ultrasonidos. Aunque el concepto seguiría siendo el mismo ya que lo único que cambia es que en vez de transmitir la señal mediante balizas láseres lo haríamos mediante balizas ultrasónicas.

# Capítulo 3.
# Marco Teórico

En el siguiente capítulo se expone teóricamente cada uno de los componentes utilizados durante el desarrollo del proyecto así como las ventajas y desventajas de utilizarlos. Toda la información que a continuación se expone se pueden encontrar en los enlacen siguiente de la bibliografía: [1], [3], [4], [12], [13], [14], [15], [17], [18], [19].

## 3.1 Localización

La localización es el proceso por el cual se conoce la posición de un objeto en el espacio.

Esto se consigue a partir de  la información recogida del medio, se puede conseguir mediante el propio objeto o a través de diferentes sistemas sensoriales, como los usados en este proyecto.

### 3.1.1 Tipos de localización

Se puede realizar una clasificación de los sistemas de localización existentes según los datos que tengamos:

- Conociendo o no la posición del objeto:
    - Localización local: Aquella donde conocemos la posición inicial del robot y se pretende estimar la siguiente posición.
    - Localización global: Se intenta conocer la posición del robot sin conocer información adicional de la misma.
- Representación de la posición:
    - Sistemas determinísticos: conociendo la posición actual del sistema, las observaciones de los sensores y el comportamiento del robot ante los cambios del entorno, entonces se puede predecir sin ningún riesgo de error la siguiente posición del robot.
    - Sistemas probabilísticos: son aquellos en los que las diferentes variables que intervienen en el proceso de localización se ven afectadas por variables aleatorias tales como el ruido y por tanto la forma en la que se representa la posición del robot es en base a una probabilidad.

### 3.1.2 Técnicas de estimación de la posición

Para poder desarrollar un sistema de localización es importante que se conozca la posición en la que encuentra y de este modo poder conocer la información necesaria para poder alcanzar su destino de manera eficaz.

Hoy en día existen diferentes técnicas utilizadas como la trilateración, triangulación, análisis de la escena y proximidad. A continuación se explica brevemente en qué consisten tanto la trilateración como la triangulación, ya que se realizó un estudio previo para considerar cuál de las dos era más eficaz para el proyecto.

- Triangulación: En esta técnica en lugar de utilizar distancias se utilizan ángulos para determinar la posición de un nodo. Se necesitan dos ángulos y la distancia entre dos puntos de referencia en un entorno bidimensional. Varios receptores miden el tiempo de llegada de una señal, pero para poder calcular el ángulo se necesitaría de un sensor adicional único para este trabajo.

  La desventaja de esta técnica es la forma en la que hemos de calcular el ángulo y el coste computacional que este conlleva para operar con fórmulas trigonométricas.

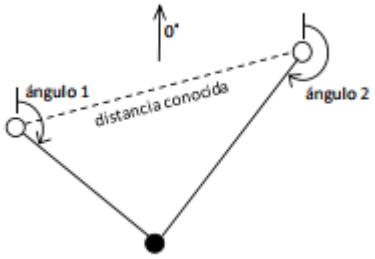



Figura 3.1: Triangulación

- Trilateración: La técnica de trilateración calcula la posición de un nodo midiendo las distancias desde él mismo hasta varias posiciones de referencia (balizas). Para calcular la posición de un nodo en dos dimensiones es necesario conocer al menos las distancias desde tres balizas en diferentes líneas (no colineales), mientras que en tres dimensiones son necesarias cuatro balizas en planos distintos.

  Una vez que el receptor conoce la distancia a los diferentes emisores, es capaz de conocer su posición intentando encontrar aquel punto del espacio en el cual las medias obtenidas de los diferentes emisores son coherentes.

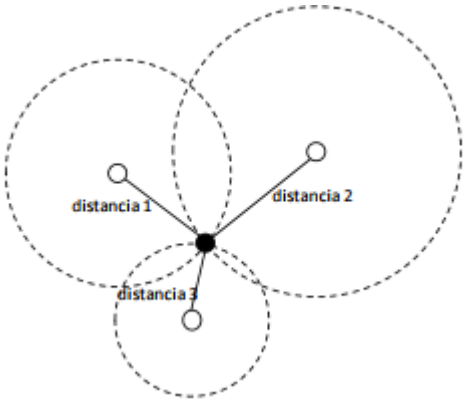



Figura 3.2: Trilateración

### 3.1.3 Sistema de localización escogido

De los sistemas de localización comentados anteriormente, en este proyecto se ha optado por el método de trilateración debido a que se busca un sistema básico, que tenga un bajo coste tanto operacional como computacional y económico. Además, para la forma en la que desarrollamos el proyecto no nos ha surgido la necesidad de calcular la orientación para el cual se tendría que recurrir a sistemas auxiliares.

### 3.1.4 Cálculos para la trilateración

La trilateración se basa en una serie de ecuaciones que deberemos llevar a cabo para el cálculo de la posición del objeto.

Para empezar vamos a partir de la ecuación de distancia, siendo $r_i$ la distancia que hay entre los emisores, cuya coordenadas son ($x_i$, $y_i$) y el receptor obtenemos nuestra posición. Por lo tanto partiremos de las siguientes tres ecuaciones:

$$(X - x_1)^2 + (Y - Y_1)^2 = r_1^2$$
$$(X - x_2)^2 + (Y - Y_2)^2 = r_2^2$$
$$(X - x_3)^2 + (Y - Y_3)^2 = r_3^2$$

Figura 3.3: Ecuaciones de la distancia

Podemos ampliar los cuadrados de cada uno:

$$x^2 - 2x_1X + x_1^2 + Y^2 - 2y_1Y + Y_1^2 = r_1^2$$
$$x^2 - 2x_2X + x_2^2 + Y^2 - 2y_2Y + Y_2^2 = r_2^2$$
$$x^2 - 2x_3X + x_3^2 + Y^2 - 2y_3Y + Y_3^2 = r_3^2$$

Si restamos la segunda ecuación con la primera obtenemos:

$$(-2x_1 + 2x_2)X + (-2y_1 + 2y_2)y = r_1^2 - r_2^2 - x_1^2 + x_2^2 - Y_1^2 + Y_2^2.$$

Del mismo modo, restando la tercera con la segunda:

$$(-2x_2 + 2x_3)X + (-2y_2 + 2y_3)y = r_2^2 - r_3^2 - x_2^2 + x_3^2 - Y_2^2 + Y_3^2.$$

Este es un sistema de dos ecuaciones con dos incógnitas:

$$\begin{aligned} Ax + by &= C \\ Dx + EY &= F \end{aligned}$$

Cuya solución es:

$$x = \frac{CE - FB}{EA - BD}$$

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

Figura 3.4: Sistema de ecuaciones de la trilateración

## 3.2 Sensor ultrasónico SRF04

Como se ha comentado anteriormente, en robótica móvil y para poder realizar la localización se usan una gran variedad de sensores que miden distintas magnitudes.

En este proyecto se ha hemos optado por la utilización del ultrasonido como magnitud para realizar la medida de las distancias y proceder a realizar la localización. En concreto se ha utilizado el sensor ultrasónico SRF04 debido a su eficacia y bajo coste. A continuación se expondrán varios aspectos técnicos de este sensor.

SRF04 es un sensor de distancias por ultrasonidos capaz de detectar objetos y calcular las distancias a la que se encuentran en un rango de 3 a 300 cm.

### 3.2.1 Funcionamiento del sensor

El sensor funciona emitiendo impulsos de ultrasonidos inaudibles para el oído humado. Los impulsos emitidos viajan a la velocidad del sonido hasta alcanzar un objeto, entonces el sonido es reflejado y captado de nuevo por el receptor de ultrasonidos. Lo que hace el controlador incorporado es emitir una ráfaga de impulsos y a continuación empieza a contar el tiempo que tarda en llegar el echo. Este tiempo se traduce en un pulso de echo de anchura proporcional a la distancia a la que se encuentra el objeto. A continuación se observa una imagen a través de un osciloscopio:

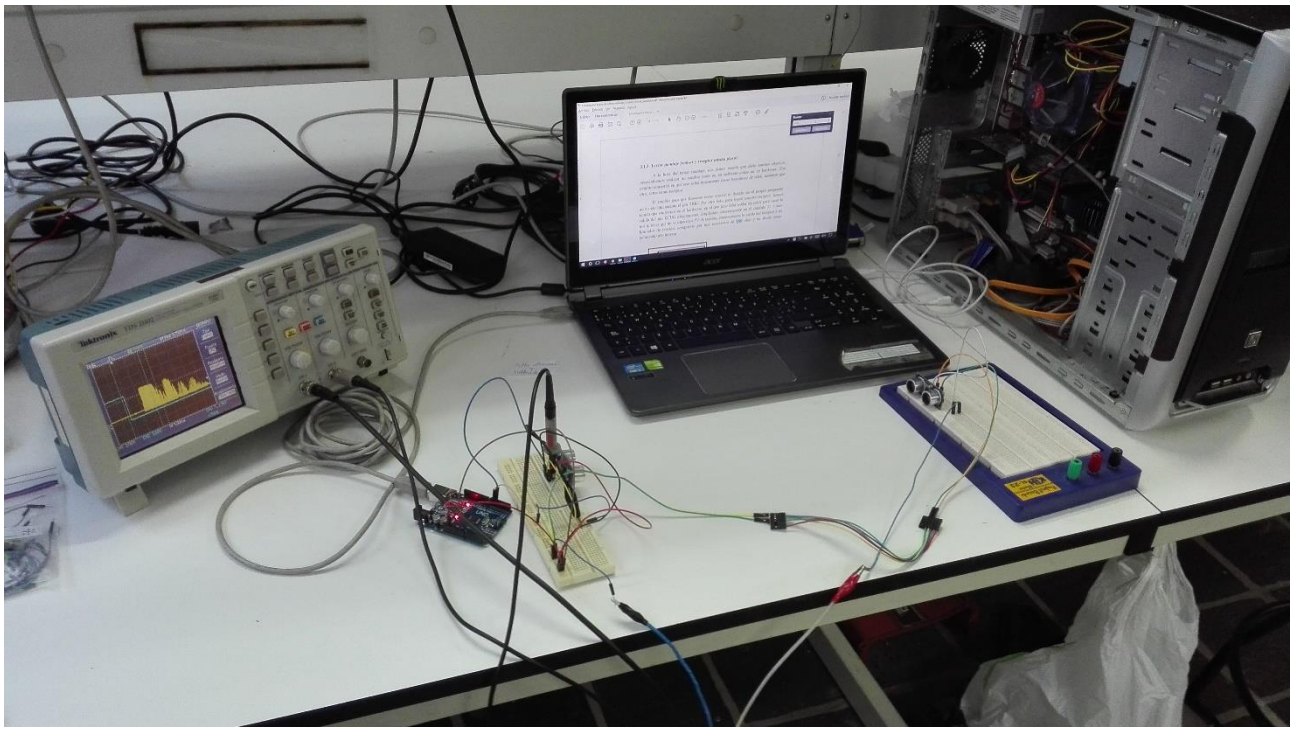


Figura 3.5: Diagrama de tiempo del ultrasonido en un montaje físico

Desde un punto de vista más práctico, lo que se hace es suministrar un pulso corto en el pin Trigger para iniciar la medición y a continuación se envía una ráfaga de ocho pulsos ultrasónicos a 40kHz para posteriormente poner el pin Echo a 1 (estado alto). Cuando se detecta otro pulso en el Trigger (algo ha sido devuelto) este bajará su línea de Echo. La anchura del pulso Echo será proporcional a la distancia a la que se encuentra el objeto. Si no se tiene Echo (no se encuentra objeto alguno) este pulso tendrá una longitud aproximada de 36ms. Es recomendable dejar un retardo de 10ms desde que se hace la lectura hasta que se realiza la siguiente para que el circuito esté estabilizado.

A continuación se muestra un esquema del proceso de funcionamiento:

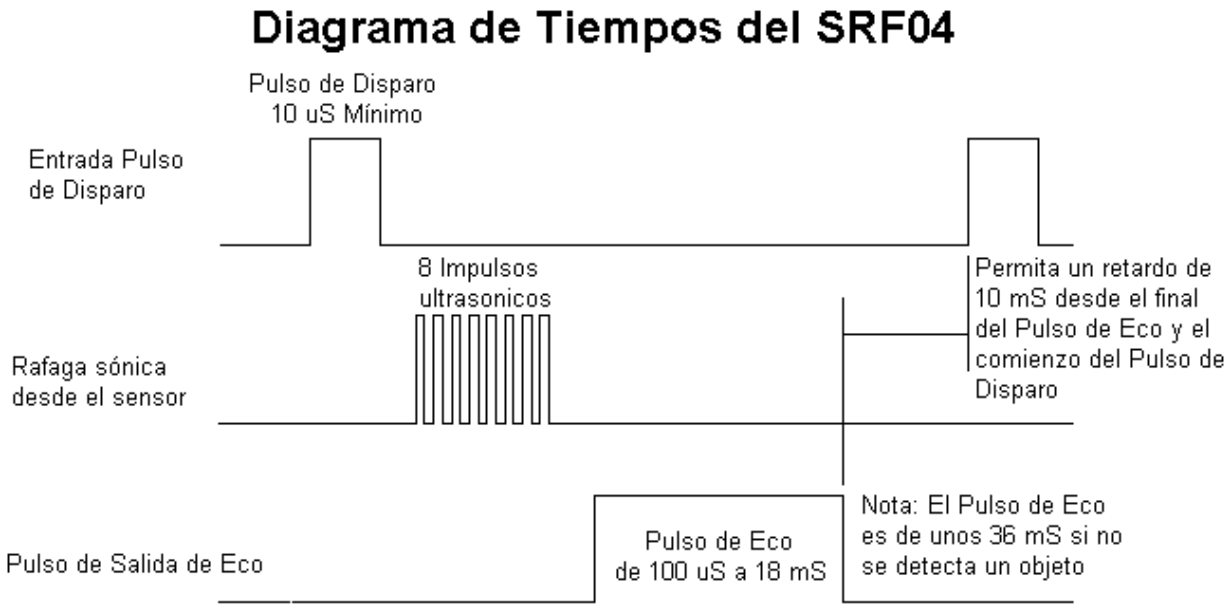



Figura 3.6: Diagrama de Tiempos del SRF04

A continuación se muestra cuales son los pines correspondientes del sensor ultrasónico:

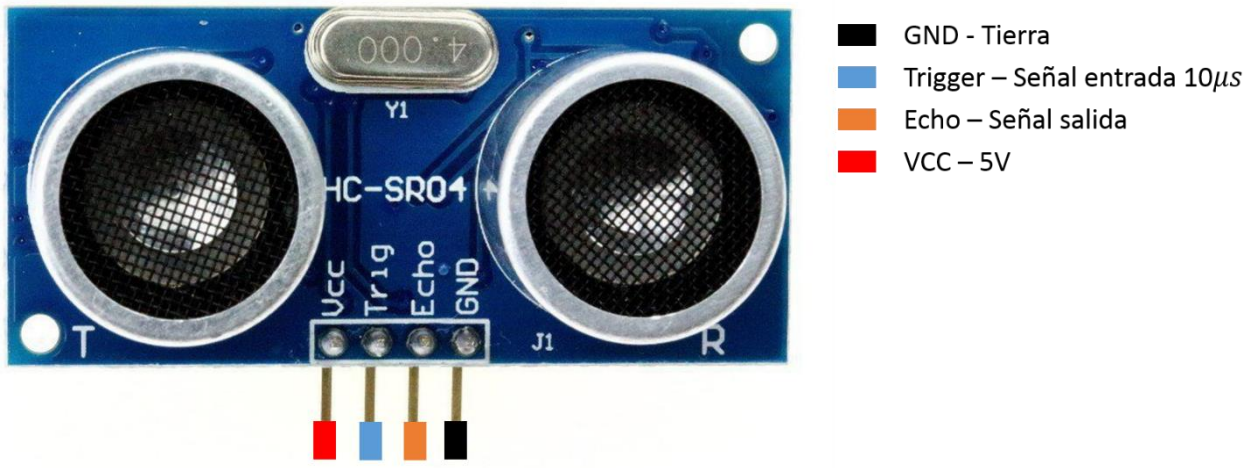



Figura 3.7: Sensor ultrasónico y sus pines

### 3.2.2 Calculo de la distancia

La obtención de la distancia hacia un objeto viene dada por la fórmula de la distancia en relación a la velocidad y el tiempo, la cual es “d = v * t”; donde “d” es la distancia que queremos conocer, “v” es la velocidad del sonido y “t” es el tiempo transcurrido desde el envió del pulso hasta su recepción.

Si suponemos que la velocidad del sonido es constante, 340 m/s, la distancia la obtendremos multiplicando dicha velocidad por el tiempo en que tardemos en enviar y recibir nuestro pulso.

Una pequeña variación a lo anterior es que sabemos que la distancia obtenida en este caso sería el resultado de ida y vuelta de la onda por lo que la expresión anterior quedaría de la siguiente forma:

$$d = \frac{340m/s * t}{2}$$

### 3.2.3 Características técnicas

A continuación se expone una imagen con algunas de las características técnicas a tener en cuenta a la hora de utilizar nuestro sensor ultrasónico.

| | |
|---|---|
| Tensión | 5V |
| Consumo | 30 mA Tip. 50mA Max. |
| Frecuencia: | 40 Khz. |
| Distancia Mínima: | 3 cm. |
| Distancia Máxima: | 300 cm. |
| Sensibilidad: | Detecta un palo de escoba a 3 m. |
| Pulso de Disparo | 10 uS min. TTL |
| Pulso de Eco: | 100 uS - 18 mS |
| Retardo entre pulsos: | 10 mS Mínimo |
| Pulso de Eco: | 100 uS - 18 mS |
| Tamaño: | 43 x 20 x 17 mm |
| Peso: | 10 gr. |

Tabla 1.1: Características Técnicas del SFR04

### 3.2.4 Conceptos necesarios

Para poder utilizar de forma eficiente nuestro sensor ultrasónico tendremos que tener en cuenta algunas consideraciones previas. En este apartado solo se expone algunos conceptos que creemos esenciales a la hora de la utilización, en la bibliografía se podrá encontrar más información respecto a esto.

#### 3.2.4.1 Zona muerta

Los sensores de ultrasonido tienen una zona muerta en la cual no pueden detectar exactamente el objeto. Esta es la distancia entre la membrana sensora y el mínimo rango de sensibilidad. Si el objeto está demasiado cercano la señal ultrasónica puede chocar contra el objeto antes de que dicha señal haya dejado el sensor, por la tanto la información del echo es ignorada.

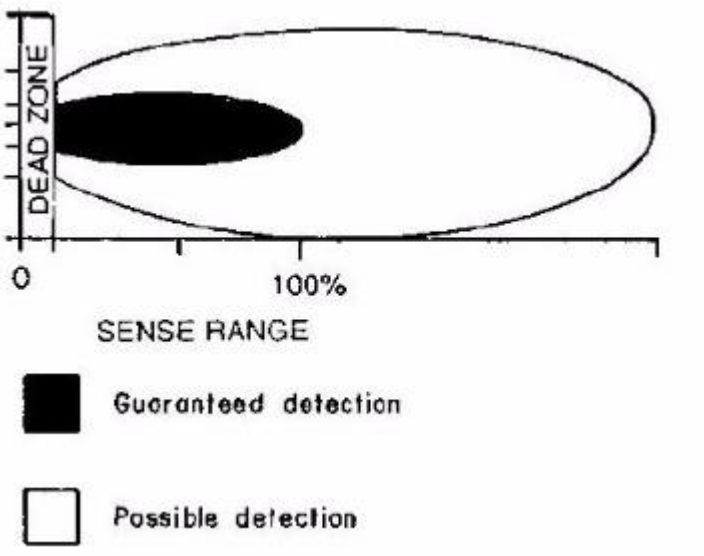



Figura 3.9: Zona muerta

#### 3.2.4.2 Rango efectivo

Se debe prestar especial atención al rango efectivo de disparo, ya que fuera de este la señal ultrasónica seguiría existiendo pero sería demasiado débil para poder ser detectada. Tal y como dijimos en apartados anteriores se puede observar que este rango es de unos 30º aproximadamente.

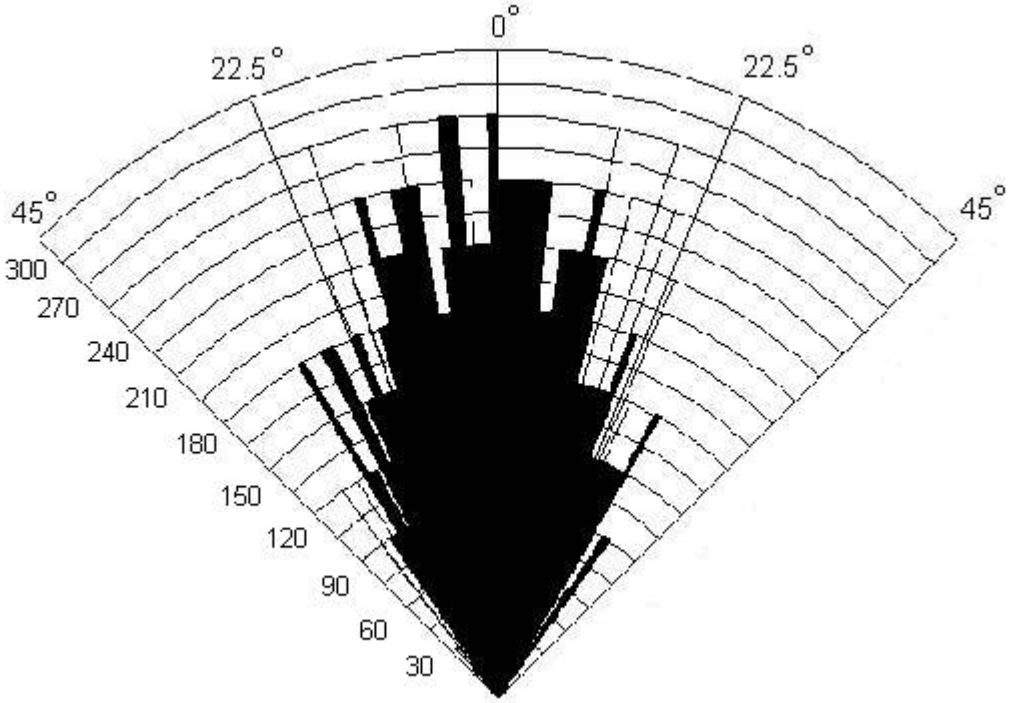



Figura 3.10: Rango efectivo del SRF04

### 3.2.4.3 Inclinación del haz de ultrasonido

Si un objeto liso es inclinado más de 3º con respecto a la normal al eje del haz de emisión la señal de ultrasonido es desviada del sensor y la distancia de detección disminuye. Sin embargo, para objetos pequeños situados cerca del sensor, la desviación respecto a la normal puede aumentar hasta 8º, si el objeto está inclinado más de 12º.

La señal que choca contra un objeto de superficie rugosa se difunde y refleja en todas las direcciones y par de la energía vuelve al sensor como un echo débil.

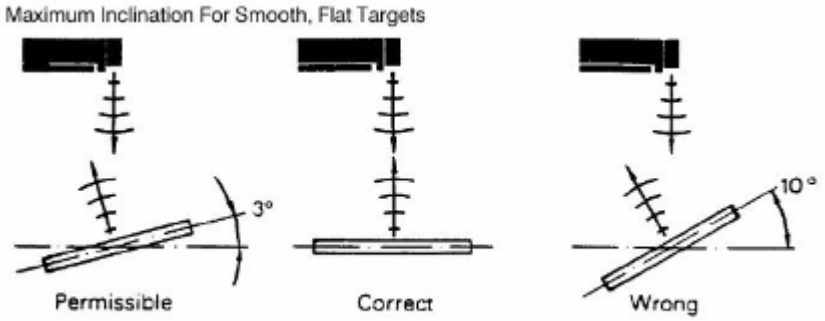



Figura 3.11: Dependencia del rango de inclinación

## 3.3 Módulos de radio-frecuencia

Los módulos de radio-frecuencia 433 Mhz son un equipo de radio-frecuencia que está compuesto por dos módulos, un transmisor y un receptor inalámbricos que funcionan a una frecuencia de 433 Mhz. Dado que sólo uno de ellos es transmisor, la comunicación de datos sólo funcionará en un sentido, por lo que se necesitan dos pares.

Estos módulos son muy sencillos y reciben gran cantidad de ruido. Tanto el transmisor como el receptor trabajan en frecuencias comunes. Por lo tanto, será necesario un método de filtrar este ruido y para emparejar el transmisor y el receptor.

Su alcance depende del voltaje con el que alimentemos el módulo y si usamos o no una antena, es decir, con un voltaje de 5V y con la antena del módulo el alcance no superara los 2 metros mientras que si lo alimentamos con 12V y utilizamos una antena de cobre su alcance podría llegar a los 300 metros.

En nuestro proyecto hemos optado por el uso de estos módulos debido a su fácil programación, así como, su bajo precio y medio-largo alcance. Se debe tener en cuenta que para su uso se necesita de una librería llamada “VirtualWire” la cual incluiremos en nuestro código de arduino.

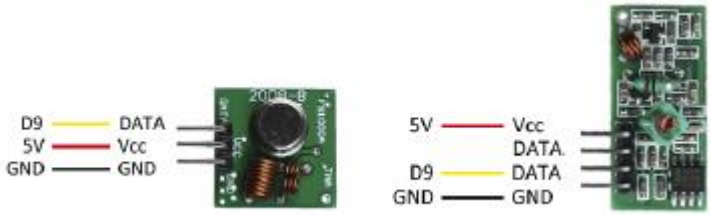



Figura 3.12: Pines de los módulos RF (transmisor-receptor)

### 3.3.1 Uso de los módulos RF

Para poder realizar la comunicación inalámbrica entre varias placas Arduino se tendrá que utilizar la librería anteriormente comentada que nos proporciona varias funciones para el uso de estos módulos.

A continuación se exponen algunas de las funciones utilizadas en los códigos, aunque se podrán encontrar más en la bibliografía.

- vw_set_tx_pin (transmit_pin)→Configura el pin de transmisión.
- vw_set_rx_pin (receive_pin)→Configura el pin de recepción.
- vw_setup (2000)→ inicializa la biblioteca.
- vw_send (mensaje, longitud)→ Transmite un mensaje con una longitud dada.
- vw_wait_tx ()→Espera a que el mensaje sea transmitido en su totalidad.
- vw_get_message (buf, y buflen))→Lee el último mensaje recibido. Esta debe ser llamada sólo cuando se conoce un mensaje para ser recibido.

## 3.4 Arduino

Arduino es una plataforma de prototipos de electrónica de código abierto basada en hardware y software flexible y fácil de usar. Se trata de una placa programable con entrada y salidas digitales y analógicas cuyo bajo coste es ideal para iniciarse en proyectos tanto de electrónicas como informáticos.

En este proyecto se utilizan dos tipos de placa debido al uso y funcionalidades que nos proporciona cada una de ellas. Para los sensores que estarán distribuidos por la habitación hemos utilizado la placa “Arduino Uno” y para los sensores que llevara nuestro robot incorporado utilizaremos la placa “Arduino MEGA”.

A continuación, se especifican cada una de las características de estas placas.

### 3.4.1 Arduino UNO

Arduino UNO es la placa que se basa en el microcontrolador ATmega328 y que se utilizara para enviar tanto la señal ultrasónica como la señal de radio frecuencia.

Esta placa cuenta con 14 pines de entrada/salida de los cuales 6 pueden ser salidas PWM (Modulación de ancho de pulsos) y otras 6 como entradas de tipo analógico.

También cuenta con dos pines para la utilización de las interrupciones, lo cuales son el 2 y 3, por ello, y como se comentara en el capítulo cuatro, se desecha el uso del Arduino Uno para los receptores, ya que al estar utilizando seis no se puede dedicar una interrupción a cada uno de ellos.

Algunas de las características más importantes de esta placa son las siguientes:

- Microcontrolador ATmega328
- Puertos I/O Digital 14 (6 PWM)
- Puertos de Entrada Analógica 6
- Memoria flash 32 KB
- SRAM 2KB
- EEPROM 1KB
- Velocidad del reloj: 16 MHz

Figura 3.13: Placa Arduino UNO

### 3.4.2 Arduino MEGA

Arduino MEGA es la placa que se basa en el microcontrolador ATmega2560 y que se utilizara para la recepción de la señal ultrasónica y radio frecuencia así como de los cálculos necesarios de nuestro sistema de localización.

Esta placa cuenta con 54 pines de entrada/salida de los cuales 14 pueden ser salidas PWM (Modulación de ancho de pulsos) y otras 16 como entradas de tipo analógico.

Además esta placa cuenta con seis pines dedicados al uso de las interrupciones, los cuales son el 2, 3, 18, 19, 20 y 21, por ellos, y como se comentara en el capítulo cuatro, se decide utilizar esta placa para nuestros receptores, pudiendo de esta forma dedicar una interrupción a cada uno de ellos.

Algunas de las características más importantes de esta placa son las siguientes:

- Microcontrolador ATmega2560
- Puertos I/O Digital 54 (14 PWM)

- Puertos de Entrada Analógica 16
- Memoria flash 256 KB
- EEPROM 4KB
- SRAM 8KB
- Velocidad del reloj: 16 MHz

Figura 3.14: Placa Arduino MEGA

## 3.5 Raspberry Pi

En el proyecto se utiliza este sistema empotrado para el uso de ROS, el cual comentaremos más adelante. Simplemente incorporaremos este sistema al robot para conseguir una simulación en tiempo real.

Como en todo proyecto se ha tenido que determinar cuál era la mejor placa, llegando a la conclusión que por eficacia y utilidad escogeríamos la Raspberry Pi 3 model b.

Algunas de las características de esta placa son las siguientes:

- Memoria
  - 512 KB Caché L2 Compartida
  - 1 GB RAM DDR2
- CPU
  - 64 bit quad-core
  - Procesador ARM Cortex-A53 a 1,2 GHz
- GPU
  - 1080p30 H.264/MPEG-a AVC
  - OpenGl ED 2.0, MPEG-2, VC-1
- Almacenamiento mediante MicroSD

- Salidas de vídeo HDMI, RCA y DSI
- 4xPuerto USB 2.0 Tipo A
- Conectividad de red
    - 10/100 Ethernet
    - Wifi 802.11n
    - Bluetooth 4.1
- 17xPuerto GPIO

Figura 3.15: Raspberry Pi 3 model B

## 3.6 Rosserial (ROS)

En el proyecto vamos a utilizar rosserial para comunicar nuestro robot con nuestra placa arduino y así poder generar una simulación en tiempo real y física.

Ros es un sistema open source meta-operativo para un robot. Proporciona los servicios que se espera de cualquier sistema operativo, incluyendo abstracción de hardware, control de dispositivos de bajo nivel, entre otros.

Este es un framework distribuido que permite que los ejecutables sean diseñados individualmente y ligeramente acoplados en tiempo de ejecución.

Ros actualmente sólo funciona en plataformas basadas en UNIX, mientras que el software se prueba principalmente en sistemas Ubuntu o Mac-OS

Se tienen tres niveles de conceptos, sistemas de archivos, gráfico de la computación y el de la comunidad. A continuación vemos de lo que está compuesto cada nivel.

- Nivel de sistemas de archivos: Este nivel está compuesto por los paquetes, metapaquetes, manifiesto de paquetes, repositorios, tipos de mensajes (msg) y tipos de servicios (srv).
- Nivel de gráfico de computación: Este nivel está compuesto por los nodos, master, servidor de parámetros, mensajes, temas, servicios y bolsas

- Nivel de la comunidad: Este último nivel está compuesto por las distribuciones y los repositorios. Aquí solo hemos nombrado lo más importante, ya que este nivel está compuesto por algunas características más.

Ros está basado en una arquitectura de grafos y está compuesto por dos partes básicas, Ros como sistema operativo y ros-pkg.

Existen otros metasistemas muy parecidos a Ros como Carmen, Microsoft Robotics Studio y Player, entre otros.

Por último, para construir algo con un nombre en Ros tenemos que seguir las siguientes características:

1. El primer carácter es [a-z][A-Z], ~ o /.
2. Los siguientes caracteres pueden ser alfanuméricos, subrayados o /.

# Capítulo 4.
# Marco Práctico

En el presente capítulo se muestra el desarrollo del proyecto paso a paso, es decir, desde las primeras implementaciones de los códigos así como de los primeros montajes realizados y los resultados obtenidos.

## 4.1 Desarrollo del proyecto

Como se ha comentado a lo largo de esta memoria el objetivo principal del proyecto es dotar de un sistema de localización a un robot. Para ello se ha tenido que realizar varias investigaciones y pruebas tanto con el hardware como con el software del proyecto, empezando desde lo más sencillo hasta el propio montaje final.

Esto es necesario para poder determinar el uso y el montaje final de nuestro proyecto.

### 4.1.1 Primer montaje (un ultrasonido)

El primer montaje se ha centrado en estudiar el funcionamiento del dispositivo ultrasónico SRF04 como sistema estándar de medida. Esto se ha realizado en una placa Arduino UNO y a continuación se muestra un esquema del montaje realizado:

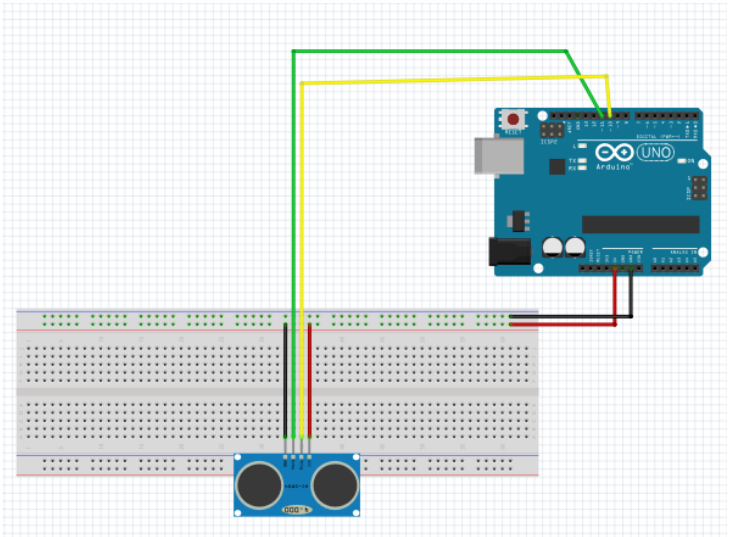


Figura 4.1: Montaje inicial SRF04

Mediante la ayuda del programa (medido_LED.ino), que se puede ver en el apéndice A, se ha verificado el funcionamiento de dicho medidor de distancia obteniendo los resultados que ponemos ver:

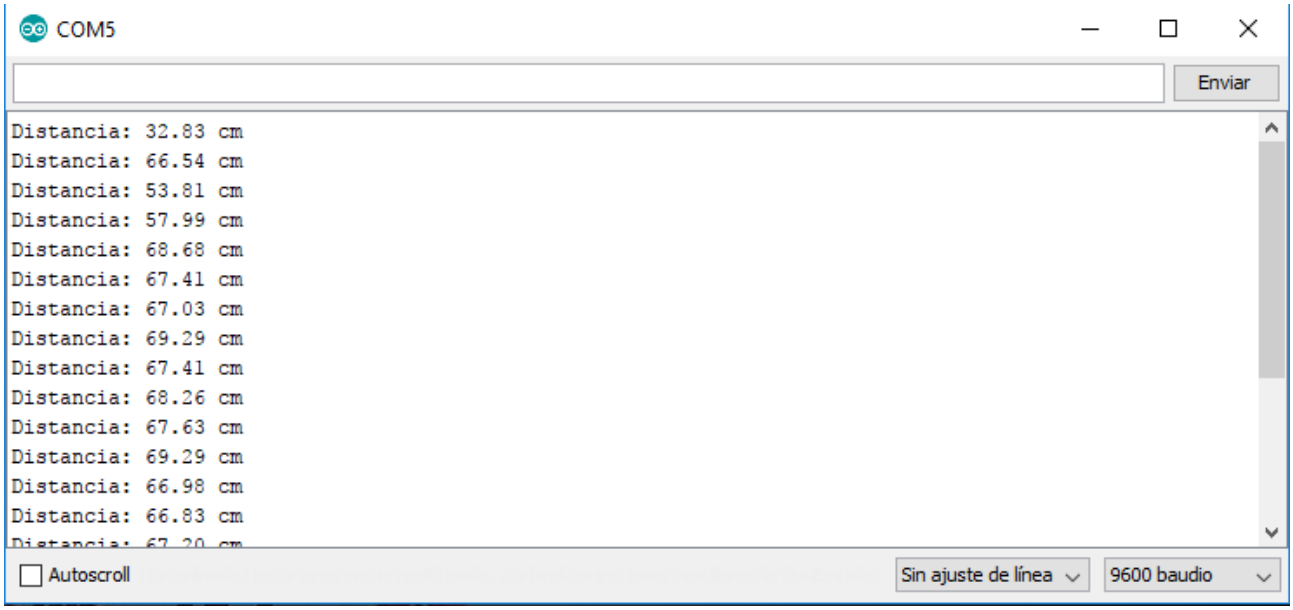



Figura 4.2: Medidas del sensor SRF04

Además con la ayuda de un osciloscopio se comprueba que el diagrama de tiempos del sensor, explicado en el capítulo anterior, funciona de manera correcta dándonos cuenta de que a medida que la distancia a medir es mayor o menos el tren de pulsos del Echo se alejaba o se acercaba.

Aquí podemos ver una imagen real del osciloscopio donde se prueba:

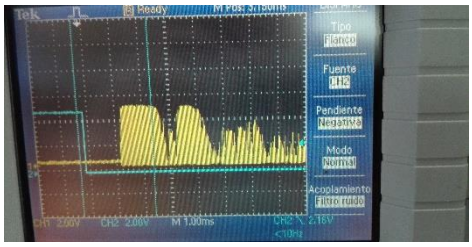


Figura 4.3: Distancia alejada

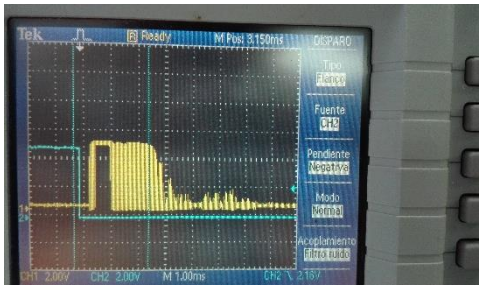


Figura 4.4: Distancia cercana

### 4.1.2 Segundo montaje (dos ultrasonidos independientes)

A medida que el proyecto va avanzando se utilizan varios sensores, optando por comprobar cómo sería el funcionamiento de dos de los sensores ultrasónicos conectados a una misma placa Arduino UNO.

Esto no supuso muchos cambios ni en el hardware, ya que solo fue añadir un sensor más y sus respectivas conexiones, ni el software, ya que como se podrá ver en el apéndice A solo tuvimos que añadir algunas líneas más al código anterior (medidor_2_SR04.ino).

A continuación se muestra el nuevo montaje:

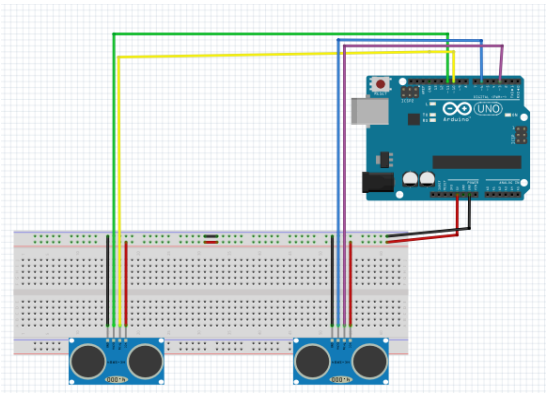


Figura 4.5: Segundo montaje (SRF04 independientes)

Para verificar que este montaje es correcto se ha realizado las pruebas con el código desarrollado, obteniendo los resultados que se pueden ver en la próxima imagen. Como se ha podido comprobar cada sensor mide su propia distancia de manera independiente sin molestarse, por lo que se concluye que en una misma placa podremos utilizar más de un medidor de distancia.

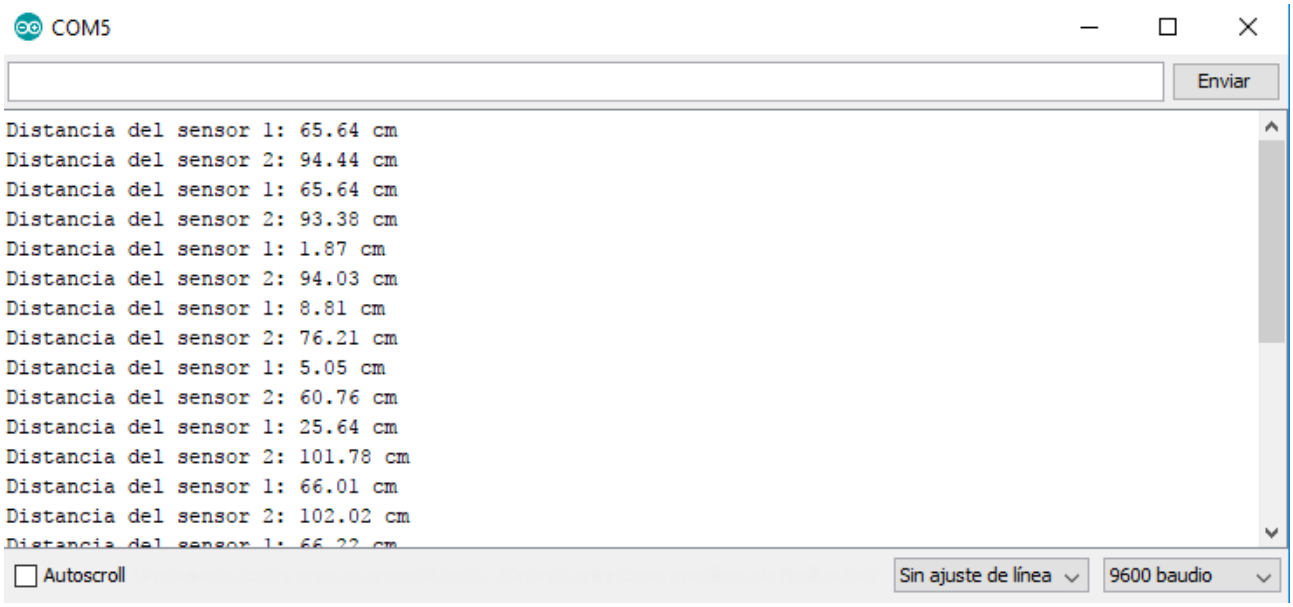



Figura 4.6: Medidas de los sensores independientes

### 4.1.3 Tercer montaje (Módulos de radio frecuencia)

Una vez realizadas las pruebas anteriormente comentadas, se pasa a realizar tanto el código, que se podrá ver en el apéndice A (RF_emisor.ino y RF_receptor.ino) como unas pequeñas pruebas del funcionamiento y transmisión de datos de los módulos de radio frecuencia.

Fijándonos en las conexiones anteriormente comentadas en el capítulo 2 el circuito quedaría de la siguiente forma:

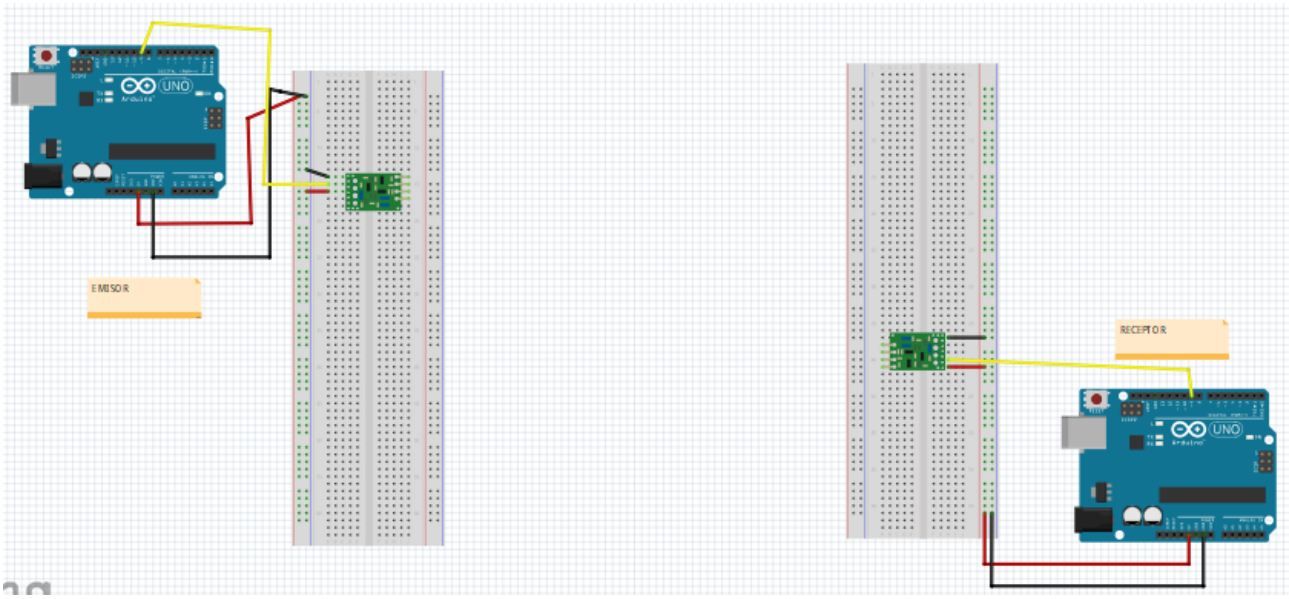



Figura 4.7: Montaje de los módulos RF

Una vez desarrollado el hardware y utilizando en los códigos tanto la librería “VirtualWire”, anteriormente comentada, como las funciones, también comentadas, necesarias para establecer una correcta comunicación se han realizado, mediante la ayuda de un herramienta externa como RealTerm, una prueba para verificar que la comunicación es estable y el mensaje enviado por el emisor es recibido y verificado por el receptor. A continuación vemos la solución a la prueba realizada.

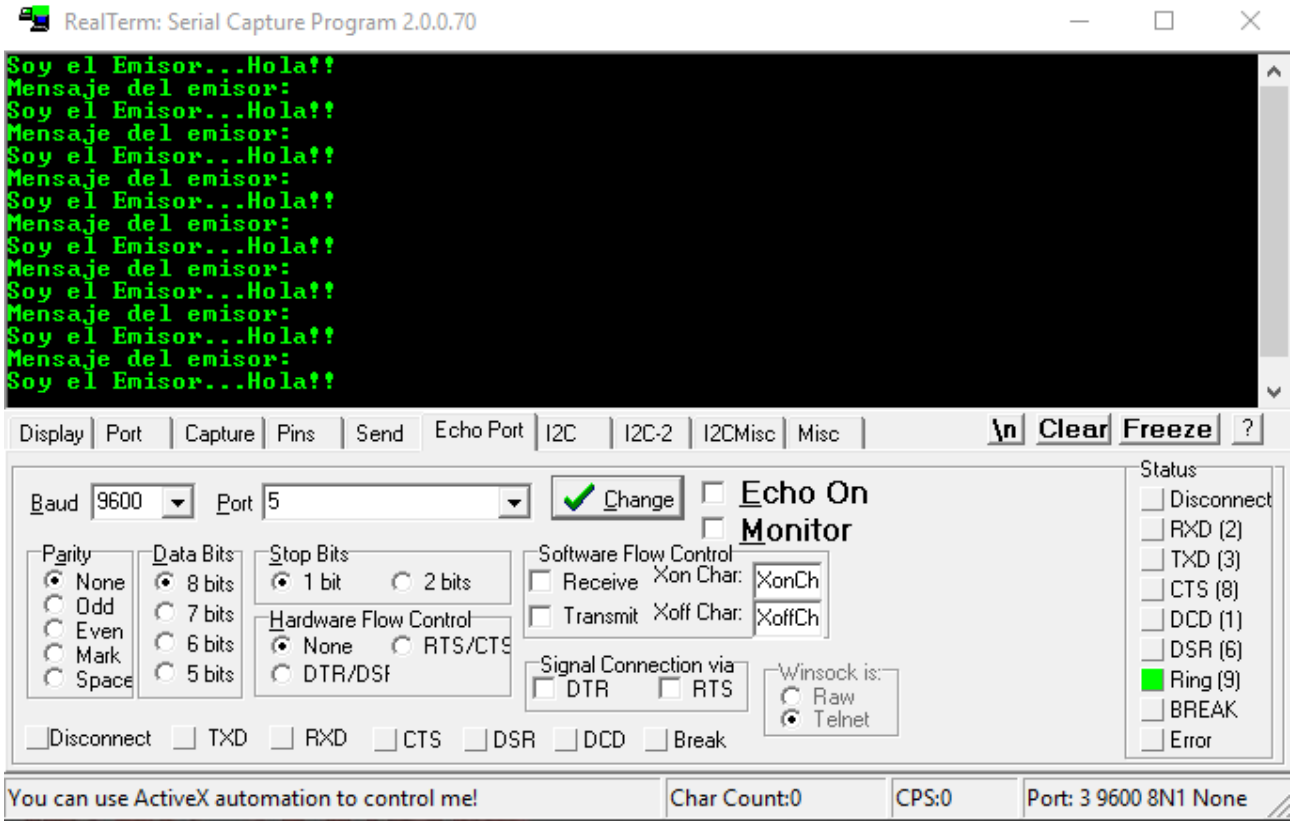



Figura 4.8: Prueba comunicación por radio frecuencia

Para poder utilizar esta herramienta se ha tenido que realizar unas configuraciones previas que a continuación comentamos:

- En la pestaña “Display”, seleccionamos la opción Ascii y luego Binay Sync Chars.
- En la pestaña “Port” cambiamos los parámetros que viene por defecto, poniendo la velocidad de baudios en 9600 y el puerto correspondiente, en nuestro caso el 4(puerto del TRIG).
- Por último, en la pestaña “Echo Port” volvemos a cambiar los parámetros que vienen por defecto, volviendo a poner la velocidad de baudios en 9600, el puerto correspondiente, en nuestro caso el 4 (puerto del Echo) y activamos las opciones de Echo On y el monitor para poder recibir el mensaje.

### 4.1.4 Cuarto montaje (Emisor y receptor)

Una vez realizadas todas las pruebas iniciales, se ha empezado a orientar nuestro proyecto a su fin. A la hora de realizar este montaje, nos dimos cuenta que dado nuestro objetivo, se necesitaban que los sensores no funcionaran de manera tradicional si no que uno de ellos lo hiciera como únicamente el emisor de la señal y otro solo como receptor de dicha señal.

Para realizar este objetivo se ha tenido que realizar algunos cambios tanto en el software como en el hardware.

El cambio para que funcione únicamente como emisor se realiza en el propio programa, activando únicamente el pin TRIG de nuestro sensor y jamás activando el pin Echo del mismo. Por otro lado, para lograr que el resto de sensores actuase sólo como receptores se ha tenido que realizar los cambios en el propio hardware del sensor, es decir, se ha tenido que soldar un cable para sacar nuestra salida del pin Echo directamente.

El nuevo montaje quedaría de la siguiente forma:

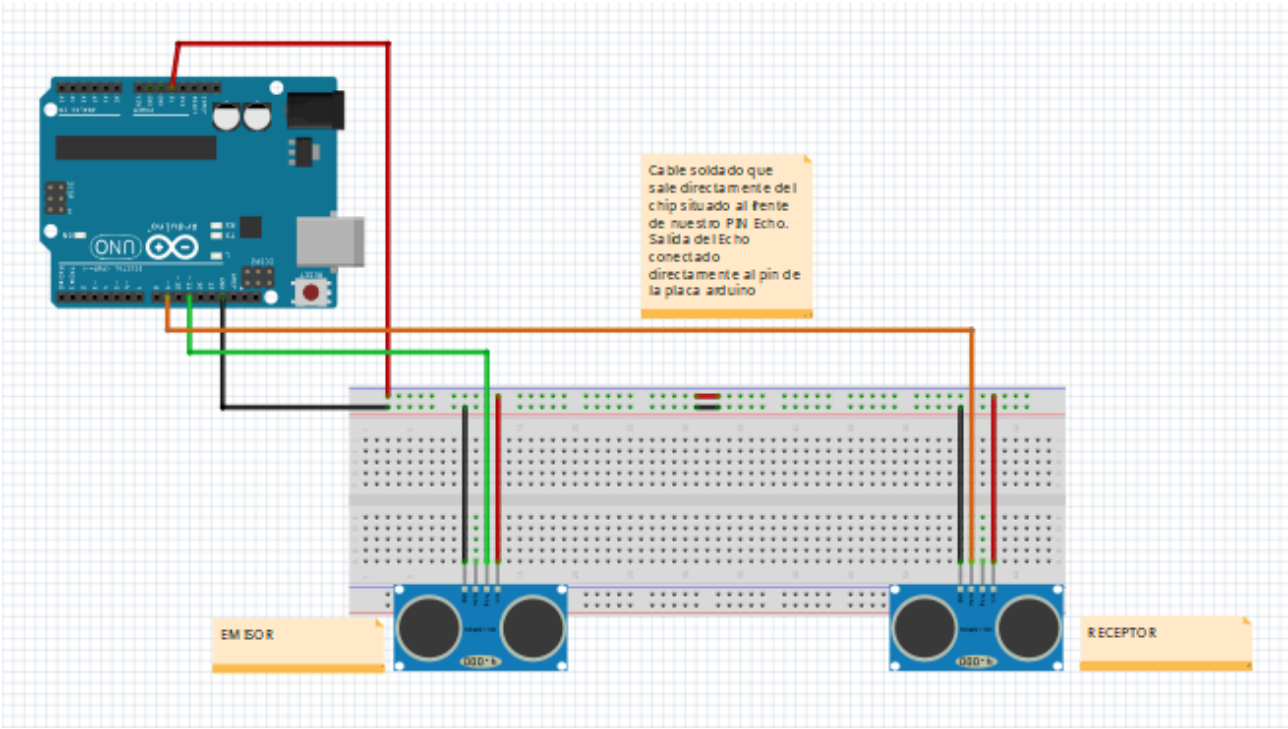



Figura 4.9: Montaje de sensor emisor y sensor receptor

Cuando realizamos este montaje nos fijamos que la medición tanto del tiempo como de la distancia no se podía realizar de la manera tradicional por lo que tendríamos que encontrar alguna manera de poder obtener los tiempos de llegada de la señal así como del cálculo de la distancia a la que se encuentran el emisor del receptor.

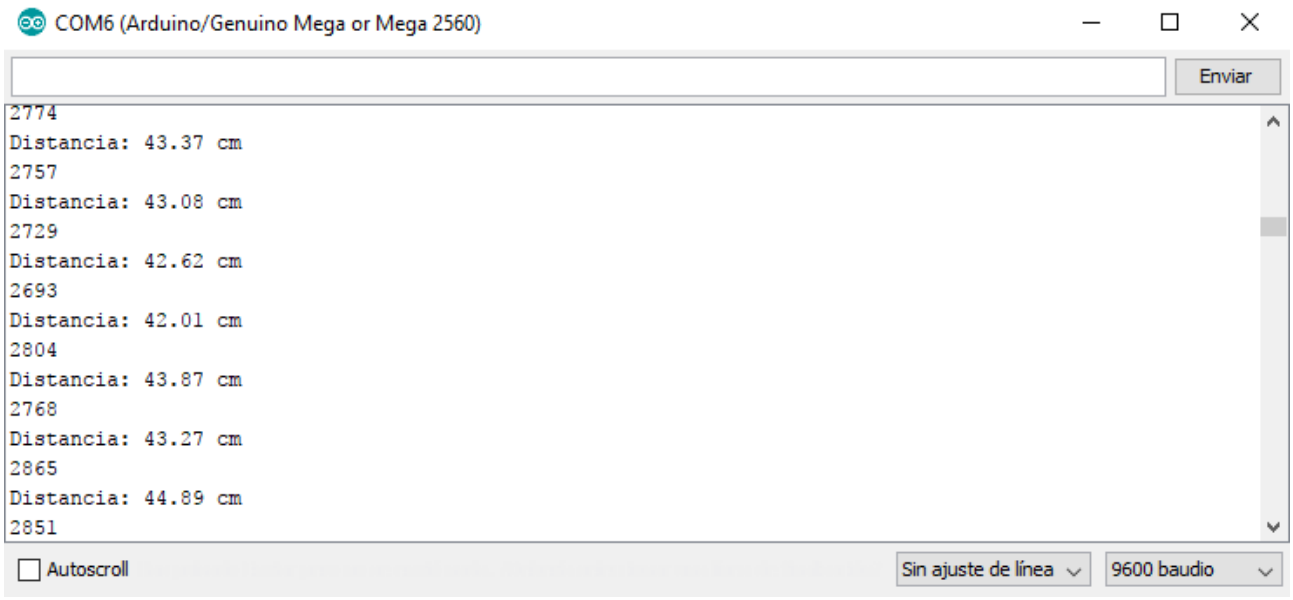



Figura 4.10: Prueba con emisor y receptor en la misma placa

### 4.1.5 Quinto montaje (Emisor y receptor + RF)

Al montaje anteriormente realizado se le hicieron unas pequeñas modificaciones, como añadirle los módulos de radio-frecuencia y colocar cada sensor en un protoboard diferente.

El nuevo montaje quedaría de la siguiente manera:

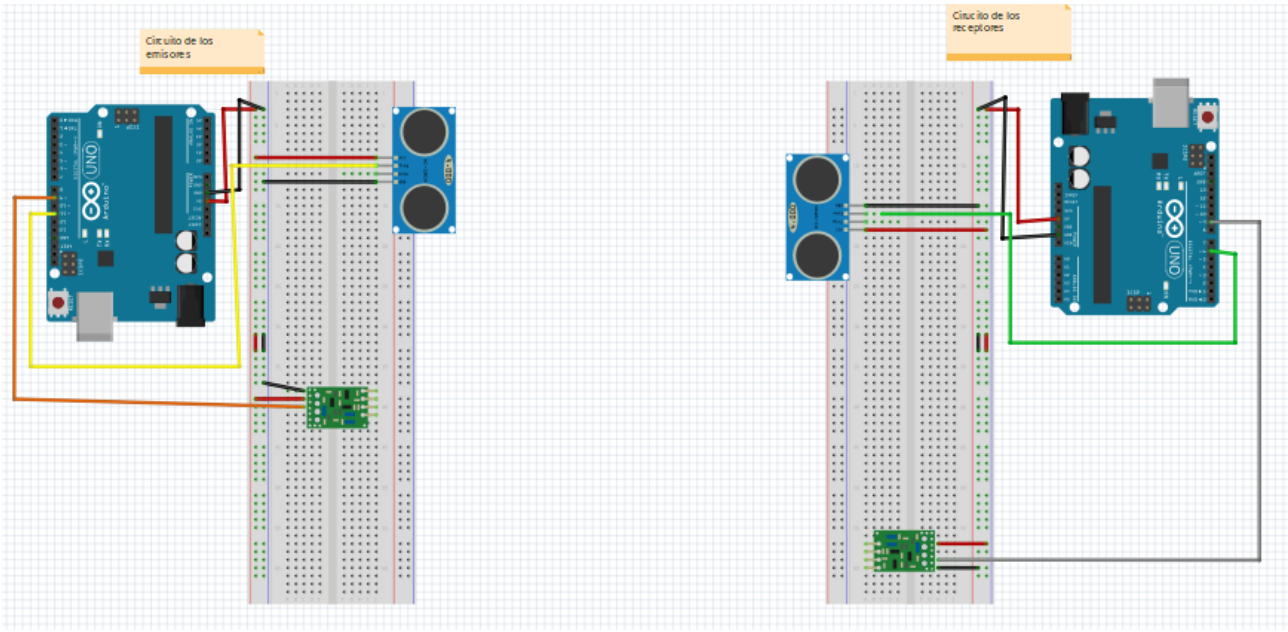



Figura 4.11: Montaje emisor y receptor + RF

Fue en esta parte del proyecto donde empezamos a desarrollar un método para obtener el tiempo que tardaba en llegar la señal del emisor al receptor y donde calculamos la nueva fórmula que se utilizaría para la distancia. Todo esto se podrá ver en el código que incluiremos en el apéndice A (SR04_2_Receptor.ino).

A continuación se procede a explicar los métodos utilizados en cada caso:

- Tiempo de llegada de la señal: En esta parte del proyecto se utiliza la librería “TimerThree” para poder controlar y configurar nuestro propio timer (interrupción) en arduino. Utilizando las funciones de la propia librería para realizar un preescalado del timer y la propia función de arduino que hace que salte una interrupción en un pin asignado por nosotros en el momento que le digamos llevándonos a la ISR que hemos creado. En la ISR configuramos nuestra interrupción para que vaya guardando el valor del timer cada vez que se genere una y así hasta cuatro veces, ya que luego en el programa principal (loop) realizaremos una media para obtener el tiempo medio de llegada de la señal, esta obtención del valor del timer se hace con la variable “TCNT3”. Por último para poder realizar esta media se tendrá que tener en cuenta que nos ha llegado también el mensaje transmitido por radio frecuencia, ya que este llegara antes que el ultrasonido, también se tendrá que tener en cuenta que se nos ha generado la interrupción para poder realizar nuestra media, ya que tendremos que tener los cuatro tiempos.

- Cálculo de la ecuación de la distancia: Con los sensores funcionando de forma distinta a la tradicional y con un nuevo timer generado por nosotros, para poder calcular la distancia entre los sensores se ha tenido que realizar un sondeo de medidas obtenidas de la distancia real y el tiempo que se genera. Se han tenido que sacar varias muestras para poder refinar lo máximo posible la nueva ecuación.

| Tiempo (ms) | Distancia (cm) |
|---|---|
| 550 | 4 |
| 982 | 10 |
| 1562 | 20 |
| 2727 | 40 |
| 3336 | 50 |
| 3604 | 65 |
| 4960 | 80 |
| 5532 | 90 |
| 5586 | 100 |
| 7893 | 125 |
| 9044 | 150 |
| 10681 | 175 |
| 11929 | 200 |
| 13820 | 225 |
| 15538 | 250 |

Tabla 1.2: Relación tiempo y distancia real

Partiendo de la tabla anterior y con la ayuda de la herramienta Excel, podemos hallar las constantes que los relacionan, ya que al final será lo que se necesitara para la nueva ecuación. Para esta parte nos ayudaremos de la ecuación de la recta (Y = m*X+b).

En Excel se crea la tabla anteriormente expuesta, luego seleccionamos la tabla e insertamos un gráfico en forma de recta y por último, seleccionando los puntos de nuestra gráfica nos saldrá un panel con opciones en el cual tendremos que seleccionar “Agregar línea de tendencia” y la opción de “Presentar ecuación en el gráfico”. De esta forma obtenemos tanto la gráfica que nos muestra la línea de tendencia de nuestra recta así como los nuevos valores.

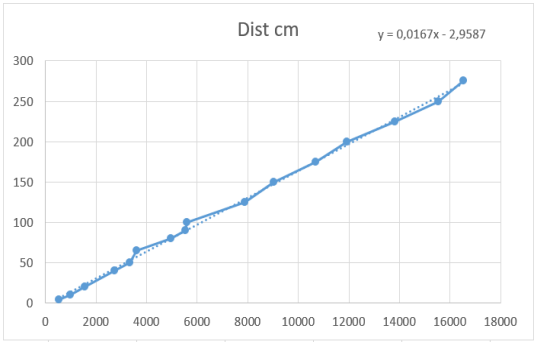



Figura 4.12: Gráfica de la ecuación de la recta

Por lo tanto la nueva ecuación de la distancia será:

distancia = ((0.0167* tiempo) – 2.9587)

Figura 4.13: Nueva Ecuación de la distancia

Para verificar que esta nueva ecuación realizaba la medida correctamente, se ha realizado una tabla donde se refleja la distancia real y la distancia experimental, observando que los valores se acercan bastante a los valores reales, aunque en algunos casos se nos alejan hasta 2 o 3 cm.

| Distancia real (cm) | Distancia experimental (cm) |
|---|---|
| 40 | 43,22 |
| 100 | 101 |
| 150 | 149,83 |
| 200 | 198,13 |

Tabla 1.3: Relación distancia real y experimental

### 4.1.6 Sexto montaje (Montaje final)

Una vez se han realizado todas las pruebas con cada uno de los circuitos y programas comentados anteriormente, hemos realizado el montaje del circuito final, quedando este de la siguiente forma:

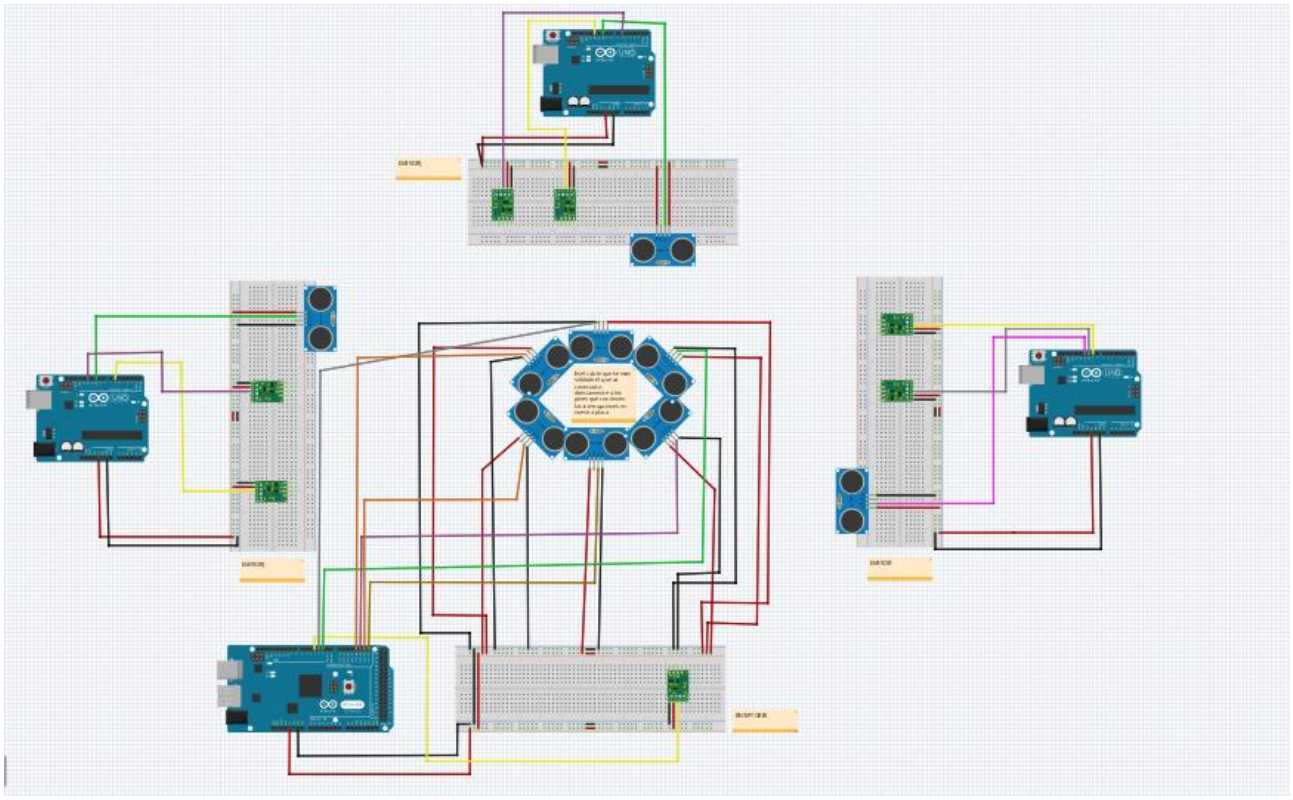


Figura 4.15: Montaje final

Como podemos observar en la imagen este montaje no varía mucho en comparación con el quinto, mencionado en el apartado anterior.

Se han utilizado tres emisores, cada uno de ellos con su propio emisor de radio frecuencia y su propia placa Arduino UNO, en estos emisores irán los software encargados de la emisión de la señal ultrasónica y de radio frecuencia. Como podemos observar estos emisores realizan el mismo trabajo por lo que el software no cambiara mucho de uno a otro, estos códigos los podremos ver en el apéndice A (SR04_2_Emisor).

Una modificación que se ha tenido que realizar en el hardware de los emisores es que además de llevar su propio emisor también tendrán que llevar su propio receptor. Esto se hace para lanzar y ejecutar las emisiones tanto de radio-frecuencia como ultrasónica de manera ordenada, es decir, el emisor A no emitirá hasta pasado dos segundo desde que reciba el mensaje del emisor C, el emisor B no emitirá hasta pasado dos segundos desde que le llega el mensaje del emisor A y el emisor C no emitirá nada hasta que pasen dos segundos desde que le llega el mensaje de B.

Con esto se consigue que ambas señales no se solapen las unas con las otras y hagan que los receptores no funcionen correctamente, ya que estamos siguiendo un orden de emisión.

Lo que si cambia con respecto a los anteriores montajes es el receptor, ya que en primer lugar, al utilizar seis ultrasonidos cada uno de ellos tendrá que ir conectado a los pines de nuestra placa que genera una interrupción, por ello hemos utilizado la placa Arduino Mega, ya que contiene seis pines para este propósito y se usan exactamente seis ultrasonidos. Además tendremos únicamente un receptor de radio frecuencia para recibir los mensajes de los emisores. En cuanto al software del receptor, que lo podremos ver en el apéndice A (SR04_2_Receptor), se ha tenido que añadir una interrupción por cada sensor y un cálculo del tiempo y la distancia, donde se diferencia de que emisor viene la interrupción.

Otra modificación que se ha realizado en el software ha sido el incorporar todas las formulas y el sistema de ecuaciones necesarios para realizar la localización de nuestro robot.

A continuación mostramos una imagen de cómo quedaría físicamente este montaje:

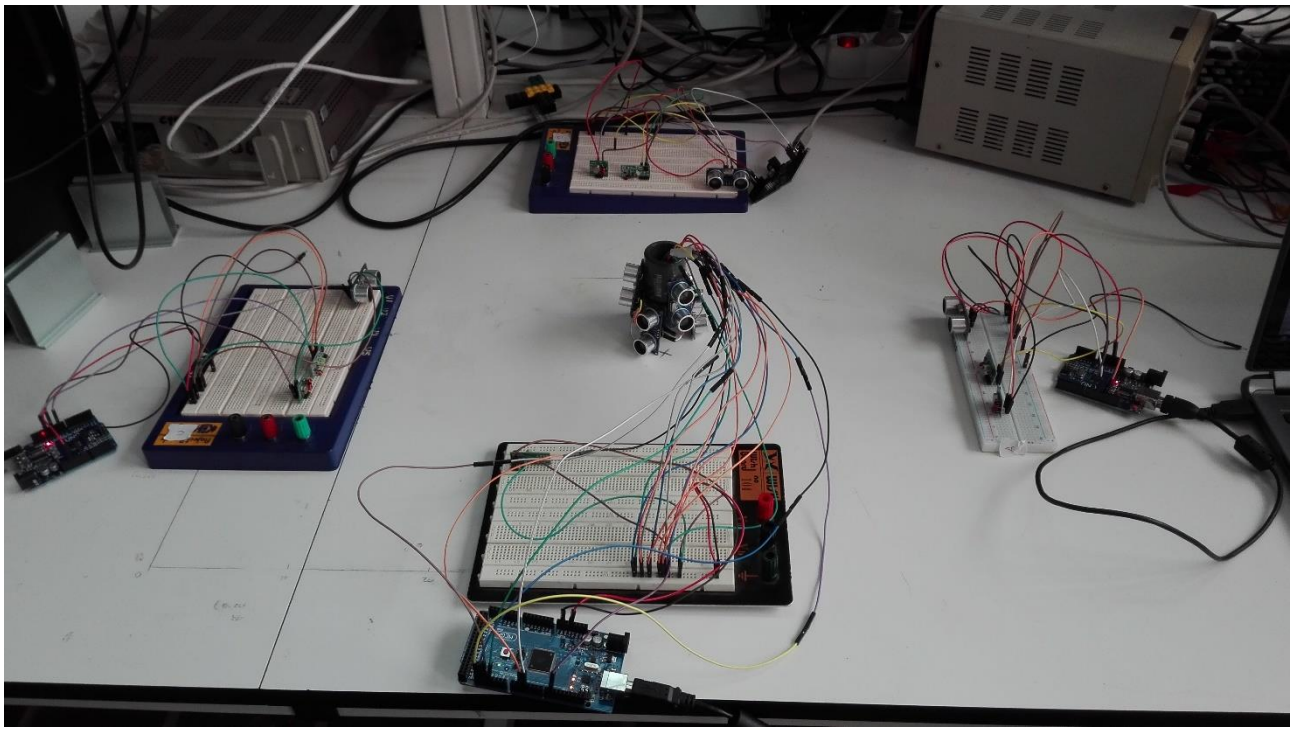


Figura 4.16: Montaje real final

Una vez todo el hardware montado y todo el software comprobado y desarrollado hemos realizado una prueba para observar que el cálculo de la distancia se ha realizado de forma correcta.

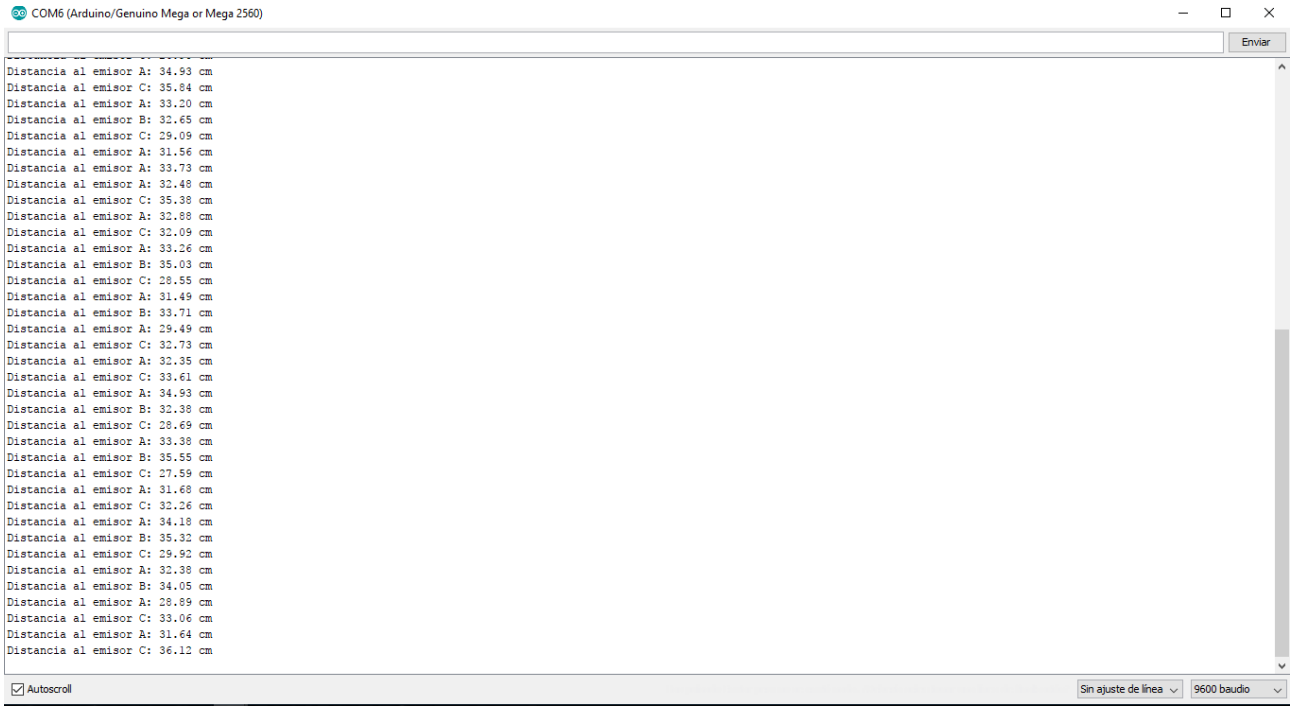



Figura 4.17: Medición de la distancia con los tres emisores

Por último, y ya comprobado que el cálculo de la distancia lo realiza de forma correcta, se ha realizado una prueba para comprobar que la localización se realizaba de manera correcta, obteniendo los siguientes resultados:

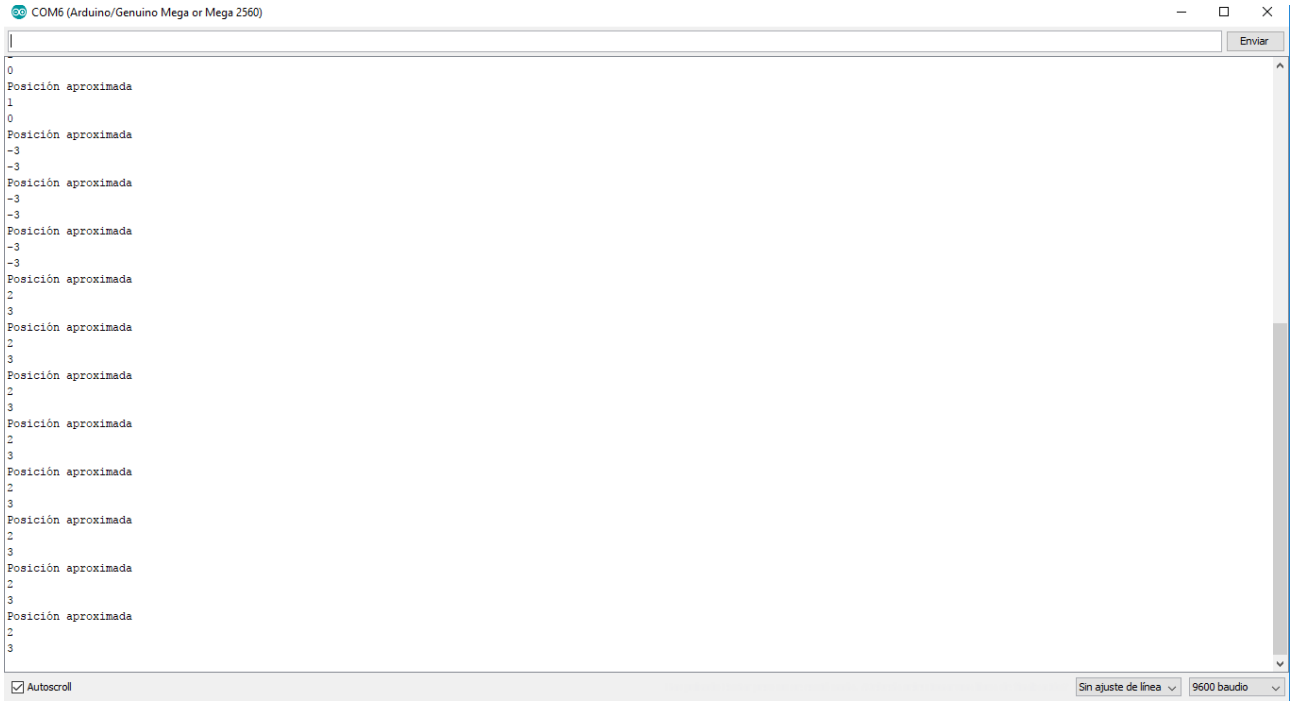



Figura 4.18: Posición mediante trilateración

### 4.1.7 Séptimo montaje (Montaje final + ROS)

En este último montaje se ha añadido a nuestro código del receptor las funciones y librerías necesarias para poder trabajar con rosserial.

Esto se utilizará para determinar la posición de nuestro robot de manera visual y para que el robot entienda lo que debe hacer gracias a nuestro código de arduino.

Para ello se utiliza un paquete denominado Odom que viene incluido en la propia librería ros.h y que permite publicar la información de posición, que será utilizada por el robot para situarse en el plano.

Tenemos que tener en cuenta una consideración previa antes de ponernos con esta parte. El código será ejecutado en la Raspberry Pi, que será el sistema encargado de gestionar este paquete y nuestro nuevo software y será el que trasmita la información a nuestro robot.

Para poder desarrollar estas modificaciones en nuestro código hemos tenido que utilizar una serie de parámetros que a continuación se detallan, todos ellos están incluidos en la librería.

- geometry_msgs::TransformStamped → Instancia de un mensaje para la comunicación.
- tf::TransformBroadcaster → Radiodifusor para la comunicación
- /base_link y /odom → Marcos sobre los que se realiza la transformación
- nh.initNode → Iniciamos el funcionamiento del nodo
- broadcaster.init(nh) → Iniciamos la comunicación con el nodo para poder tranferir la información al robot y que este la entienda.
- ros::NodeHandle nh → Objeto nodo para ROS y que será el que contendrá la información de posicionamiento

Por último, una vez desarrollado el código de manera correcta, se han realizado unas pruebas para comprobar que la localización no solo se realiza como lo hemos podido observar hasta ahora, sino que también lo podemos observar de manera visual en un plano. Esto se ha realizado con la herramienta rviz que nos trae la distribución que nos descargamos de rosserial.

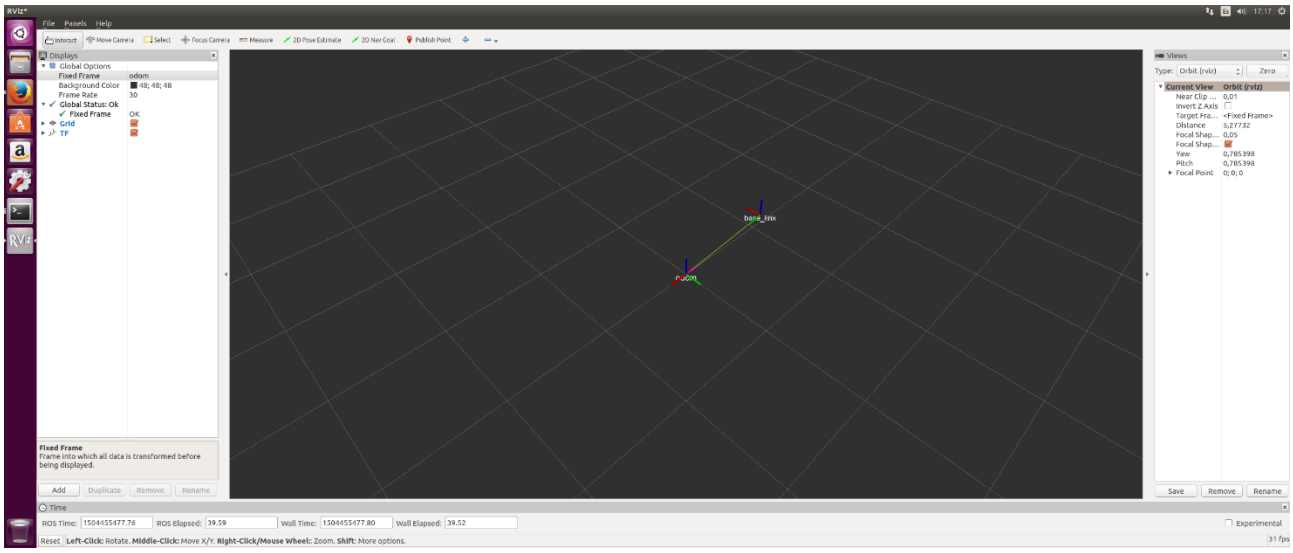



Figura 4.19: Prueba posición con rosserial

## 4.2 Observaciones

Una vez finalizado los montajes que se han llevado a cabo durante todo el desarrollo del proyecto y realizadas las pruebas oportunas, se observa una serie de puntos de interés a la hora de realizar la localización de nuestro robot en un entorno cerrado a partir de nuestro sistema diseñado. Observamos lo siguiente:

- Una de las limitaciones del prototipo ha sido el tiempo de medida, ya que al utilizar un Timer externo y varias interrupciones tenemos que tener en cuenta y tener sumo cuidado con el uso del timer y la optimización de nuestro código. Esto puede ralentizar el procesamiento y obtención del tiempo de llegada de la señal y perderíamos exactitud a la hora de calcular la distancia.
- La distancia máxima de medida no llega a los 4 metros debido a los sensores ultrasónicos. Aunque hemos realizado algunas modificaciones en el hardware de estos sensores y en el software de funcionamiento, una vez llegamos a los 4 metros observamos que la exactitud con la que se recibe el tiempo no es del todo correcta, ya que no llega de forma precisa y no siempre es recibida por el receptor. Por lo tanto hemos determinado que la distancia máxima a la que podemos estar entre los emisores y receptores sería de 3 metros. Esto se podría mejorar de alguna forma, para aumentar la distancia y que tanto el tiempo como la llegada de la señal siempre sean exactos. Esto puede deberse principalmente a la manera en la que estamos tratando la señal ultrasónica y la optimización de nuestro propio código.

- Por último, hemos observado que en el momento de calcular la distancia el error de medida es mínimo, como podemos observar en las tablas del capítulo anterior, se diferencia de la distancia real en pocas décimas y a veces en algún centímetro arriba o abajo. Lo que si tenemos que tener en cuenta es que en el momento en el que empieza la comunicación tenemos que desechar la primera obtención de la distancia ya que en algunos momentos se pueden recibir cosas que no concuerdan pero a partir de la segunda ya se estabiliza.

# Capítulo 5.
# Conclusiones y líneas futuras

Como conclusión en este trabajo de fin de grado se ha diseñado un sistema para la localización en un entorno cerrado de un robot mediantes sensores ultrasónicos y con el uso de sistemas empotrados como RaspBerry Pi 3 y Arduino.

Gracias a este proyecto hemos sido capaces de desarrollar el software de localización de un robot así como parte del hardware que se usaría en el mismo, gracias a este desarrollo se comprende el funcionamiento de un sistema de localización básico y como conseguir que un robot se mueva y se localice en un entorno cerrado como ha sido el caso. Aunque en un principio se pensó en realizarlo con sensores láseres debido a la problemática que estos estaban generando en el desarrollo del proyecto se optó por utilizar los sensores ultrasónicos.

Como ya hemos ido comentando en el transcurso de esta memoria, hemos tenido que realizar ciertas modificaciones tanto en el hardware como el software del proyecto para que este funcionara de la manera que nosotros habíamos pensado.

En cuanto a las líneas futuras, para continuar con el trabajo, se podría desarrollar una manera distinta de obtener el tiempo de llegada de la señal y realizar los cálculos de la distancia, esto entraría dentro de la optimización del código, por lo tanto lo que se podría realizar para mejorar el proyecto sería una optimización del código para conseguir que el tiempo sea más exacto de lo que ya es. Además podríamos desarrollar una manera distinta de comunicar los emisores y los receptores mediante la radio frecuencia, ya que nuestra idea no sería la única. Esto es en cuanto a los algoritmos diseñados para nuestro proyecto.

En relación al hardware de nuestro proyecto se podría usar otro tipo de sensores más potentes, aumentado así la distancia entre emisores y receptores o incluso, como el sistema empotrado de los receptores debe soportar gran cantidad de datos y realizar un procesamiento rápido de ellos para sacar dichos datos con exactitud podríamos pensar en cambiarlo por algún sistema más potente, o cuya velocidad de procesamiento sea superior a la que estamos utilizando ahora mismo.

# Capítulo 6.
# Summary and Conclusions

In conclusion in this work of end of degree has been designed a system for the location in a closed environment of a robot by means of ultrasonic sensors and with the use of systems embedded like RaspBerry Pi 3 and Arduino.

Thanks to this project we have been able to develop the localization software of a robot as well as part of the hardware that would be used in it, thanks to this development we have understood the operation of a basic localization system and how to get a robot to move and be located in a closed environment as has been the case. Although at first it was thought to realize it with sensors lasers because of the problematic that they were generating in the development of the project one chose to use the ultrasonic sensors.

As we have already commented in the course of this memory, we have had to make certain modifications in both the hardware and the software of our project so that it works in the way we had thought.

As for the future lines, to continue with the work, it could develop a different way to obtain the time of arrival of the signal and to make the calculations of the distance, this would enter into the optimization of the code, therefore what it could be done to improve our project would be a code optimization to get the time to be more accurate than it already is. In addition we could develop a different way of communicating the transmitters and receivers by radio frequency, since our idea would not be the only one. This is in terms of the algorithms designed for our project.

In relation to the hardware of our project could use other types of sensors more powerful, thus increasing the distance between emitters and receivers or even, as the embedded system of the receivers must support a large amount of data and perform a rapid processing of them to remove such data can accurately think of changing it by some more powerful system, or whose processing speed is higher than the one we are using right now.

# Capítulo 7. Presupuesto

En este capítulo se expone un presupuesto estimado del coste total del proyecto. Se diferencia entre un presupuesto de los materiales utilizados y un presupuesto del coste del desarrollo del trabajo. Por último, se expone el presupuesto final del proyecto.

## 7.1 Presupuesto de materiales

| Materiales | Cantidad | Precio/Unidad | Total |
|---|---|---|---|
| Módulos RF | 4 | 1,50 € | 6 € |
| Sensor SFR04 | 6 | 2,50 € | 22,5 € |
| Arduino UNO | 3 | 7,50 € | 22,50 € |
| Arduino Mega | 1 | 12,50 € | 12,50 € |
| Raspberry Pi 3 model B | 1 | 49,95 € | 49,95 € |
| TOTAL | | | 93,45 € |

Tabla 1.4: Presupuesto de los materiales utilizados.

## 7.2 Presupuesto del trabajo realizado

| Trabajo realizado | Horas | Precio/Hora | Total |
|---|---|---|---|
| Análisis y documentación | 50 | 10,00 € | 500 € |
| Montaje del hardware | 5 | 15,00 € | 75 € |
| Programación | 200 | 25,00 € | 5000 € |
| Pruebas | 15 | 5,00 € | 75 € |
| Documentación | 15 | 10,00 € | 150 € |
| TOTAL | | | 5.800 € |

Tabla 1.5: Presupuesto del trabajo realizado.

## 7.3 Presupuesto final del proyecto

| Descripción | Precio |
|---|---|
| Materiales | 93,45 € |
| Desarrollo | 5.800 € |
| TOTAL | 5.893,45 € |

Tabla 1.6: Presupuesto total del proyecto.

# Apéndice A.
# Códigos utilizados en el proyecto

## 1. Código para los emisores

En este apartado se encuentra únicamente un código de uno de los emisores, ya que en lo único que se diferencian con el resto es el mensaje que se envía y que además este código sería el que inicia la comunicación.

Además el resto de código de los emisores podremos encontrarlos subidos en github. En el apartado siguiente se provee del enlace para acceder a su visualización.

```
/*****************************************************************************
*
* Fichero SR04_2_Emisor.ino
*
*****************************************************************************
*
* AUTORES Alberto Martínez Chincho
*
* DESCRIPCION
*     Código que envía un carácter para poder identificar el emisor.
 *  Esto se enviara al mismo tiempo que se envía el pulso del ultrasonido emisor
*
*****************************************************************************/
//libreria necesaria para los modulos RF
#include <VirtualWire.h>
//#include <TimerOne.h>

const int pin_RF_Emisor = 11; //pin del emisor
const int pin_RF_Receptor = 9; //pin del receptor
const int pin_SR04_Trigger = 13; //pin del ultrasonido (solo trigger)

double Ultimo_envio = 0; //para saber el ultimo envio que realiza
```

```
  //Creamos un mensaje
  uint8_t sms[VW_MAX_MESSAGE_LEN];
  uint8_t sms_len = VW_MAX_MESSAGE_LEN;

  void setup() {
    Serial.begin(9600);
    pinMode(pin_SR04_Trigger, OUTPUT); //activamos el pin del ultrasonido como salida

    vw_setup(2000); //inicializamos la libreria
    vw_set_tx_pin(pin_RF_Emisor); //Configuramos el pin de emision
    vw_set_rx_pin(pin_RF_Receptor); //Configuramos el pin de recepcion
    vw_rx_start(); //Activamos el proceso de escucha
    Serial.println("Enviando...");
  }

  void loop() {
    //Serial.print("M ");
    //Serial.println(millis());
    if(vw_get_message(sms, &sms_len)){
      if(sms[0] == 'C'){ //Emitimos cuando recibamos el mensaje del emisor C
        Serial.println("Mensaje C Recibido");
        delay(2000);
        //Comunicacion RF
        const char *mensaje = "A"; //mensaje identificativo que enviaremos al receptor
        vw_send((uint8_t *)mensaje, strlen(mensaje)); //transmite el mensaje con la long
dada
        vw_wait_tx(); //esperamos que el mensaje sea transmitido en su totalidad

        //SR04(Ultrasonido)
        digitalWrite(pin_SR04_Trigger, LOW); //Para estabilizar el sensor
        delayMicroseconds(5);
        digitalWrite(pin_SR04_Trigger, HIGH); //Activamos el envio del pulso

        //Ponemos el contador a cero para que en la interrupcion no se envie señales si
todo va bien
        Ultimo_envio = millis();
      }
    }
    if (millis() - Ultimo_envio > 20000) {
      Ultimo_envio = millis();
      const char *mensaje = "A"; //mensaje identificativo que enviaremos al receptor
```

```
      vw_send((uint8_t *)mensaje, strlen(mensaje)); //transmite el mensaje con la long
dada
      vw_wait_tx(); //esperamos que el mensaje sea transmitido en su totalidad

      //SR04(Ultrasonido)
      digitalWrite(pin_SR04_Trigger, LOW); //Para estabilizar el sensor
      delayMicroseconds(5);
      digitalWrite(pin_SR04_Trigger, HIGH); //Activamos el envio del pulso

      Serial.println("Nuevo Mensaje A");
    }
  }
```

## 2. Código de los receptores

En este apartado se encuentra el código que hace que nuestros receptores funcionen de manera correcta. Debido a que este código es bastante extenso hemos decidido subirlo a la aplicación GitHub y proveeremos el enlace necesario para poder verlo. Además podremos ver el resto de código de los emisores, tal y como se comentó anteriormente.

https://github.com/alu0100698893/Sistema-de-localizacion-para-robots

```
/*********************************************************************************
*
* Fichero SR04_2_Receptor.ino
*
*********************************************************************************
*
* AUTORES Alberto Martínez Chincho
* DESCRIPCION
*    Código que se  encargara de recibir el mensaje del emisor, identificarlo y realizar
*    los cálculos del tiempo (con él timer) y la distancia (nueva fórmula a la anterior)
*    Además realizaremos mediante el sistema de trilateración y sus fórmulas una
     localización para nuestro robot.
*    Vamos a desarrollar un código que nos permita realizar una comunicación entre
*    nuestro Arduino y ROS. Además este tendrá que escenificar y obtener como mensaje
*    la información de la posición que estamos obteniendo mediante el sistema de
*    trilateración.

*********************************************************************************/
```

# Apéndice B. DataSheet componentes electrónicos

En este apéndice podrán encontrar los datasheet de los componentes electrónicos que se han utilizado a lo largo del desarrollo del proyecto.

## 1. Sensor SRF04

ELEC Freaks Tech Support: services@elecfreaks.com

### Ultrasonic Ranging Module HC - SR04

#### Product features:

Ultrasonic ranging module HC - SR04 provides 2cm - 400cm non-contact measurement function, the ranging accuracy can reach to 3mm. The modules includes ultrasonic transmitters, receiver and control circuit. The basic principle of work:
(1) Using IO trigger for at least 10us high level signal,
(2) The Module automatically sends eight 40 kHz and detect whether there is a pulse signal back.
(3) IF the signal back, through high level , time of high output IO duration is the time from sending ultrasonic to returning.
Test distance = (high level time×velocity of sound (340M/S) / 2,

#### Wire connecting direct as following:

- 5V Supply
- Trigger Pulse Input
- Echo Pulse Output
- 0V Ground

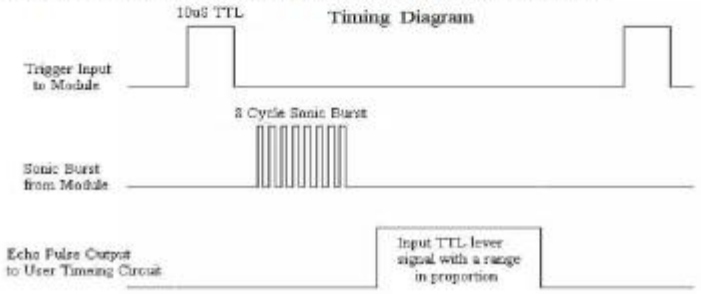
#### Electric Parameter

| | |
|---|---|
| Working Voltage | DC 5 V |
| Working Current | 15mA |
| Working Frequency | 40Hz |
| Max Range | 4m |
| Min Range | 2cm |
| MeasuringAngle | 15 degree |
| Trigger Input Signal | 10uS TTL pulse |
| Echo Output Signal | Input TTL lever signal and the range in proportion |
| Dimension | 45*20*15mm |


**Vcc Trig Echo GND**

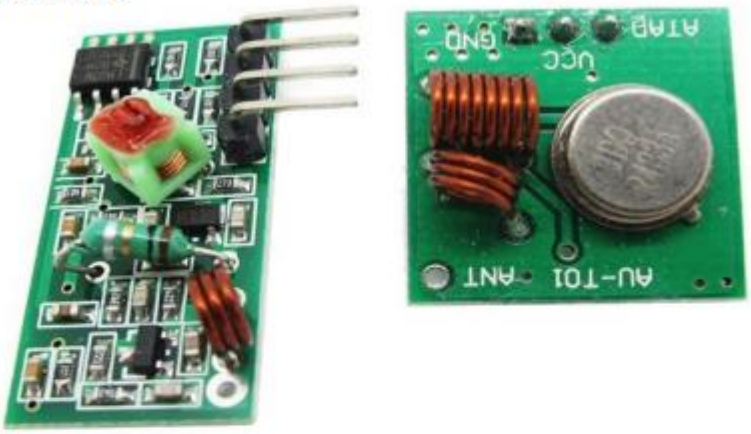
## Timing diagram

The Timing diagram is shown below. You only need to supply a short 10uS pulse to the trigger input to start the ranging, and then the module will send out an 8 cycle burst of ultrasound at 40 kHz and raise its echo. The Echo is a distance object that is pulse width and the range in proportion .You can calculate the range through the time interval between sending trigger signal and receiving echo signal. Formula: uS / 58 = centimeters or uS / 148 =inch; or: the range = high level time * velocity (340M/S) / 2; we suggest to use over 60ms measurement cycle, in order to prevent trigger signal to the echo signal.



# 2. Módulos RF 433 MhZ.

**RB-Ite-108**
**433 Mhz RF Link Kit**

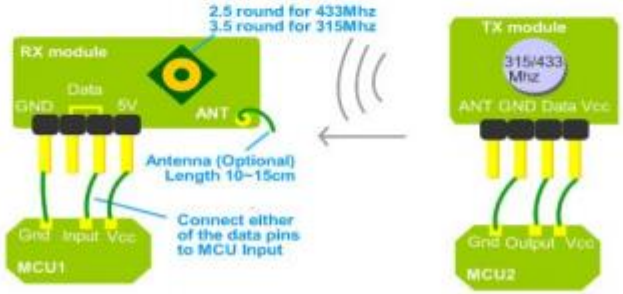



The 433MHz RF link kit is consisted of transmitter and receiver, popularly used for remote control.

**Features**

- Frequency: 433Mhz
- Modulation: ASK
- Receiver Data Output: High - 1/2 Vcc, Low - 0.7v
- Transmitor Input Voltage: 3-12V (high voltage = more transmitting power)

**Usage**




Demostration scheme of 433/315Mhz RF kit

The popular link is like this: MCU -> Encoder -> Transmitter ------ Receiver -> Decoder -> MCU

Encoder and Decoder are optional, their existence is to

## 3. SFH309.

Este componente fue utilizado en los inicios del proyecto, es decir, se encuentra en la parte de antecedentes. Podrá encontrar el datasheet utilizado en el siguiente enlace:
http://www.zdspb.com/media/manuals/SFH309FA_3mm.pdf

## 4. Transistor 2N2222.

Este componente fue utilizado en los inicios del proyecto, es decir, se encuentra en la parte de antecedentes. Podrá encontrar el datasheet utilizado en el siguiente enlace:

http://www.farnell.com/datasheets/296640.pdf

## 5. Inversor convencional.

Este componente fue utilizado en los inicios del proyecto, es decir, se encuentra en la parte de antecedentes. Podrá encontrar el datasheet utilizado en el siguiente enlace:

http://www.ti.com/lit/ds/symlink/sn7404.pdf

## 6. Schmitt-Triggers.

Este componente fue utilizado en los inicios del proyecto, es decir, se encuentra en la parte de antecedentes. Podrá encontrar el datasheet utilizado en el siguiente enlace:

http://www.ti.com/lit/ds/symlink/cd40106b.pdf

# Bibliografía


[1] Sensor SRF04. http://www.superrobotica.com/S320110.htm

[2] Sensor SRF04 Documentación técnica. https://www.robot-electronics.co.uk/htm/srf04tech.htm

[3] Conceptos teóricos sensor SRF04. http://docplayer.es/19432332-Implementacion-de-sensores-de-ultrasonidos-enun-sistema-autonomo-de-tiempo-real-m-a-esther-gilaberte-sanz.html

[4] Módulos de radio frecuencia. http://www.web-robotica.com/arduino/como-utilizar-los-modulos-rf-transmisor-y-receptor-315mhz-con-arduino

[5] Conexiones de los módulos de radio frecuencia. https://www.luisllamas.es/comunicacion-inalambrica-en-arduino-con-modulos-rf-433mhz/

[6] Librería VirtualWire. https://github.com/danielesteban/ArduinoLib/tree/master/VirtualWire

[7] Arduino y attachInterrupt. https://www.arduino.cc/en/Reference/AttachInterrupt

[8] Arduino y las interrupciones. http://www.prometec.net/interrupciones/

[9] Que son y como usar las interrupciones en Arduino. https://www.luisllamas.es/que-son-y-como-usar-interrupciones-en-arduino/

[10] Arduino Timer. https://oscarliang.com/arduino-timer-and-interrupt-tutorial/

[11] Librería TimerOne y TimerThree. https://github.com/PaulStoffregen/TimerThree

[12] Trilateración. https://es.wikipedia.org/wiki/Trilateraci%C3%B3n

[13] Técnicas de localización. http://www.dsi.uclm.es/personal/EvaMariaGarcia/docs/2008-Curso%20Verano.pdf

[14] Ecuaciones de la trilateración y como sarlas. https://math.stackexchange.com/questions/884807/find-x-location-using-3-known-x-y-location-using-trilateration

[15] Tutoriales de Rosserial-arduino. http://wiki.ros.org/rosserial_arduino/Tutorials

[16] Manuel de commandos en ROS. https://moodle2015-16.ua.es/moodle/mod/book/view.php?id=82546&chapterid=2346

[17] Arduino UNO. http://arduino.cl/arduino-uno/

[18] Arduino MEGA. http://arduino.cl/arduino-mega-2560/

[19] RaspBerry Pi 3 model b.
https://raspberryparatorpes.net/hardware/raspberry-pi-3-model-b/