

FACULTAD DE INGENIERÍA - UBA

LABORATORIO 66.02

PROYECTO FINAL

Radar Acústico

Integrantes:

Agustina BARBETTA 96528

Tutor:

Santiago LAZZARI 96735

Pablo MARINO

Eugenia MARIOTTI 96260

BELCAGUY

Manuel PORTO 96587

20 de diciembre de 2015

Índice

1. Resumen	3
2. Introducción	3
2.1. Objetivo y motivación del proyecto	3
2.2. Explicación del circuito y de sus funcionalidades	3
3. Requerimientos	4
3.1. Dados por la cátedra	4
3.2. Impuestos por nosotros mismos	5
4. Especificaciones del sistema	6
5. Preliminares	8
5.1. Elementos utilizados	8
5.2. Establecimiento de condiciones previas	8
5.2.1. Cronograma	8
5.2.2. Presupuesto de componentes	9
5.2.3. Aprendizajes	10
5.3. Análisis teórico	16
5.3.1. El sensor	16
5.3.2. El microcontrolador	18
5.3.3. El software	18
6. Armado del dispositivo	18
6.1. Programación del microcontrolador de la placa Arduino y su interfaz	25
6.1.1. Microcontrolador	25
6.1.2. Interfaz gráfica	26
6.2. Armado del banco de mediciones	27
7. Mediciones	28
7.1. Calibración	28
7.2. Prueba de campo	29
7.3. Mediciones propias del instrumento	30

8. Setups de ensayo y mediciones	31
8.1. Tensión de alimentación del instrumento	31
8.2. Consumo de corriente del instrumento	33
8.2.1. Salida de fuente de alimentación externa	34
8.3. Tensión del pin 'trigger'	35
8.4. Tensión del pin 'echo'	36
8.5. Visualización de señal de control del servomotor	37
8.6. Frecuencia y período de la señal de control del servomotor	38
9. Resultados	39
9.1. Calibración	39
9.1.1. Incerteza de medición	42
9.2. Prueba de campo	45
9.3. Mediciones propias del instrumento	50
9.3.1. Tensión de alimentación del instrumento	50
9.3.2. Consumo de corriente del instrumento	50
9.3.3. Tensión del pin 'trigger'	51
9.3.4. Tensión del pin 'echo'	52
9.3.5. Visualización de señal de control del servomotor	55
9.3.6. Frecuencia y período de la señal de control del servomotor .	58
10. Lecciones aprendidas	60
11. Conclusión	62
11.1. Verificación de requerimientos iniciales	63
12. Anexos	64
12.1. Anexo I: Fichas de datos de los componentes utilizados	64
12.2. Anexo II: Código del microcontrolador	70
12.3. Anexo III: Código de la interfaz gráfica	72
12.4. Anexo IV: Correcciones	77

1. Resumen

El presente trabajo consiste en construir un radar acústico con el cual se medirá la posición detectando distancia y ángulo de los diferentes obstáculos presentados frente al mismo. Este proyecto incluye la construcción del instrumento; compuesto por una placa Arduino UNO[1], un servomotor y un sensor de ultrasonido, el desarrollo de una interfaz gráfica para la visualización de obstáculos, la calibración del instrumento y la realización de mediciones con el mismo.

Acrónimos y palabras clave: radar, acústico, arduino, ultrasonido.

2. Introducción

2.1. Objetivo y motivación del proyecto

El objetivo del proyecto es aplicar los conocimientos aprendidos en la materia Laboratorio 66.02 mediante el desarrollo de un proyecto de nuestra elección. En nuestro caso, elegimos un radar acústico. El mismo fue ideado, construido, programado y calibrado durante el transcurso de la materia. En el presente informe se documenta el proceso de desarrollo con el objetivo de explicar y contextualizar los distintos eventos ocurridos y decisiones tomadas durante la experiencia.

2.2. Explicación del circuito y de sus funcionalidades

El diseño del radar se basa en el sensor ultrasónico HC-SR04. Los sensores de ultrasonidos son detectores de proximidad que trabajan libres de roces mecánicos y que detectan objetos a distancias que van desde pocos centímetros hasta varios metros. El sensor emite un sonido y mide el tiempo que la señal tarda en regresar. Este refleja en un objeto, el sensor recibe el eco producido y lo convierte una señal eléctrica. Este sensor trabajan solamente en el aire, y pueden detectar objetos con diferentes formas, colores, superficies y de diferentes materiales. Los materiales pueden ser sólidos, líquidos o polvorrientos, sin embargo han de ser deflectores de sonido. Este tipo de sensor trabaja según el tiempo de transcurso del eco, es decir, se valora la distancia temporal entre el impulso de emisión y el impulso del eco.

En nuestro caso, el sensor utilizado provee un rango de medición de 2 a 400 cm con una precisión que puede llegar a los 3 mm, según su ficha de datos.

Para darle movilidad al sensor, se lo montó en un micro servomotor. Este tipo de motores se puede controlar con mayor precisión que los motores de corriente estándar y posee tres cables; alimentación, tierra y control. Los servos no giran libremente como un motor de corriente estándar, sino que su ángulo de rotación se limita a 180 grados de ida y vuelta. Cuando es mandado a mover, llegará a la posición deseada y la mantendrá incluso si una fuerza externa empuja contra ella.

En nuestro caso, el radar fue ideado para barrer 180 grados. De esta forma no se enroscarán los cables del sensor ultrasónico.

Por último, como controlador del sensor y servomotor se utilizó una placa Arduino UNO. Arduino es una plataforma de hardware libre, basada en una placa con un microcontrolador y un entorno de desarrollo, diseñada para facilitar el uso de la electrónica en proyectos multidisciplinares. El hardware consiste en una placa con un microcontrolador Atmel AVR (En nuestro caso, fue el Atmega328) y puertos de entrada/salida. Por otro lado el software consiste en un entorno de desarrollo que implementa el lenguaje de programación C y el cargador de arranque que es ejecutado en la placa.

Una vez construido, el radar será capaz de detectar, la posición de un objeto cualquiera en coordenadas polares dentro de su rango.

3. Requerimientos

A continuación se presentan los requerimientos del proyecto.

3.1. Dados por la cátedra

- Que sea una idea que se pueda llevar a la realidad.
- Factible tecnológicamente.
- Factible económicoamente.
- Factible temporalmente.
- Que cumpla plazos.
- Medible con instrumentos electrónicos.

- Que posea alguna utilidad.
- Que funcione.

3.2. Impuestos por nosotros mismos

- Obtener insumos; placa Arduino UNO, sensor ultrasonico y servomotor.
- Aprender a programar el microcontrolador de la placa Arduino UNO.
- El aparato completo debe consumir, en funcionamiento nominal, una corriente menor a 1 A.
- Debe ser capaz de detectar y medir correctamente la distancia a la que se encuentra un objeto en un rango de 5 a 100 cm.
- Codificar un programa para el microcontrolador capaz dar órdenes e interpretar las señales de todos los componentes del hardware.
- Lograr que el movimiento del conjunto motor-sensor sea lo más continuo posible, sin perder funcionalidad.
- Ensamblar el aparato de forma que resista su uso y transportes durante la cursada.
- Diseñar una interfaz gráfica amigable al usuario y fácil de leer para la toma de mediciones.
- Optimizar la interfaz de forma que refleje el barrido efectuado por el sensor en tiempo real, es decir, que no haya desfasaje entre el sensor y lo mostrado en la computadora.
- Realizar sobre el radar, una medición con cada uno de los instrumentos aprendidos en la materia.
- Los errores cometidos por el radar deben poder ser analizados y cuantificados. También, se deben poder obtener valores de incertezas específicos asociados a cada uno de los resultados presentados durante el trabajo.

4. Especificaciones del sistema

El sistema incluye, tal como se explico en otras secciones, un sensor ultrasónico. El mismo funciona, según su hoja de especificaciones, a una distancia de entre 2 cm y 400 cm sin obstáculos entre el sensor y el objeto a medir. Sin embargo, se definió un rango de distancias entre las cuales se realizaron pruebas de campo. Las razones de dicha decisión son, la dificultad de construir un banco de mediciones a distancias mayores a los 100 centímetros y la presencia de errores de calibración en las mediciones del sensor. Este último punto se explicará con mayor detalle en la sección 9.1, pero a grandes rasgos consistió en determinar un rango de distancias en donde se pudo corregir la medición utilizando un factor de corrección único.

Inicialmente se midió sobre un intervalo de distancias de 2 a 60 cm, pero como se expondrá posteriormente trabajaremos sobre un rango de distancias reducido de entre 10 y 35 cm. Las razones de esto están relacionadas con el factor de corrección previamente mencionado.

Otro aspecto que es necesario explicar es la amplitud en la cual trabajaron el sensor y el servomotor. En la sección 2.2 se mencionó que el radar fue ideado para barrer un intervalo de 0 a 180 grados. En ciertos ángulos de los extremos el servomotor se atascaba, por lo que se probó el funcionamiento del mismo fuera del proyecto. Al encontrar que esta dificultad persistía, se decidió limitar el intervalo de funcionamiento exclusivamente desde los 15 hasta los 165 grados para poder utilizar el servo que tenemos.

Relacionado con el punto anterior, existe un error sistemático con la determinación del ángulo en el que se encuentra un objeto. Esto ocurre porque el ángulo de detección del radar es de 20° ¹, tomando como origen el centro del radar.

¹Este ángulo es de 15° según la ficha de datos del sensor, el valor presentado es el encontrado experimentalmente según la fuente citada.

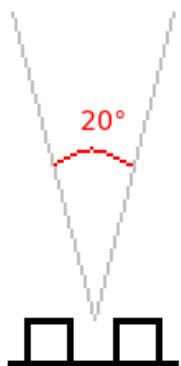


Figura 1: Angulo de detección del radar

La consecuencia de esto es que se incurre en un error de $\pm 10^\circ$ y en ciertos casos el objeto se ve mas ancho de lo normal.[3]

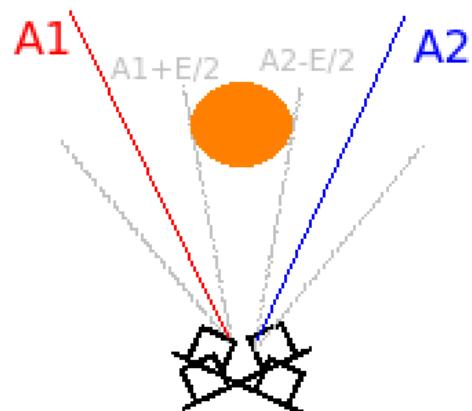


Figura 2: Ancho real del objeto vs ancho percibido por el radar

5. Preliminares

5.1. Elementos utilizados

Para la construcción del instrumento se utilizaron los siguientes componentes:

- Arduino UNO
- Sensor ultrasónico HC-SR04
- Servomotor SM-S2309S
- Protoboard de 400 puntos
- Cables macho-hembra
- Cables macho-macho

Las fichas de datos de los primeros tres se encuentran en el Anexo I.

Los instrumentos de laboratorio utilizados para realizar las mediciones fueron:

- Multímetro TRUE RMS Checkman TK-4002 como voltímetro
- Multímetro TRUE RMS Brymen BM837RS como voltímetro.
- Multímetro digital Protomax MS8221A como amperímetro
- Multímetro analógico Protomax SNM-YX36TRE-B
- Osciloscopio GOODWILL GOS-653G
- Contador GOODWILL GUC-2020

5.2. Establecimiento de condiciones previas

5.2.1. Cronograma

El desarrollo del proyecto fue basado en objetivos semanales acordados previamente con el cuerpo docente.

Semana del 05/10/2015

Elección de proyecto.

Semana del 12/10/2015

Compra de componentes necesarios y armado del circuito.

Semana del 19/10/2015

Desarrollo del programa de control del radar e interfaz gráfica.

Semana del 26/10/2015

Puesta a prueba del radar y realización de las primeras mediciones.

Semana del 2/11/2015

Mediciones y propagación de incertezas. Inicio de realización del informe.

Semana del 9/11/2015

Mediciones y propagación de incertezas. Realización del informe.

Semana del 16/11/2015

Mediciones y propagación de incertezas.
Realización del informe.

Semana del 23/11/2015

Entrega del proyecto.

5.2.2. Presupuesto de componentes

A fin de cumplir con el requerimiento de que el proyecto sea factible económicamente, elegimos la idea sabiendo que podíamos conseguir los componentes fácilmente y sin tener que comprar la gran mayoría de ellos. A continuación se detallan los precios de todos los componentes utilizados. La presente lista corresponde al periodo Octubre-Noviembre de 2015, por lo que los precios podrían no ser los mismos al momento de presentarse el proyecto.

- **Arduino UNO:** \$ 478
- **Sensor de ultrasonido HC-SR04:** \$ 65
- **Servomotor SM-S2309S:** \$ 127
- **Protoboard de 400 puntos:** \$ 82,5

5.2.3. Aprendizajes

A lo largo del proyecto se presentaron distintas problemáticas a raíz de las especificaciones que se necesitaba que el sistema poseyera y de las mediciones que se debían realizar. El propósito de esta sección es mencionar las dificultades afrontadas y explicar la solución encontrada.

Uno de los primeros problemas afrontados fue el de programar la placa para controlar a ambos, servomotor y sensor, como un conjunto. Además, se debió programar una interfaz gráfica para poder visualizar los resultados mas fácilmente. Para ello, se debió familiarizarse con los lenguajes de programación de Arduino y Processing (para la interfaz). Estos lenguajes están basados, y tienen una gran similitud, con C y Java respectivamente. Habiendo aprendido lo necesario de ambos lenguajes, lo siguiente fue la programación del sistema, proceso explicado con mayor detalle en la sección 6.1.

Luego de la programación del radar se realizaron mediciones para determinar la posición de un objeto. Al realizarlas, se advirtió un error de calibración en el radar que se debió corregir. Este proceso se explica detalladamente en la sección 9.1. A continuacion se presentan algunas imagenes de las mediciones mencionadas.

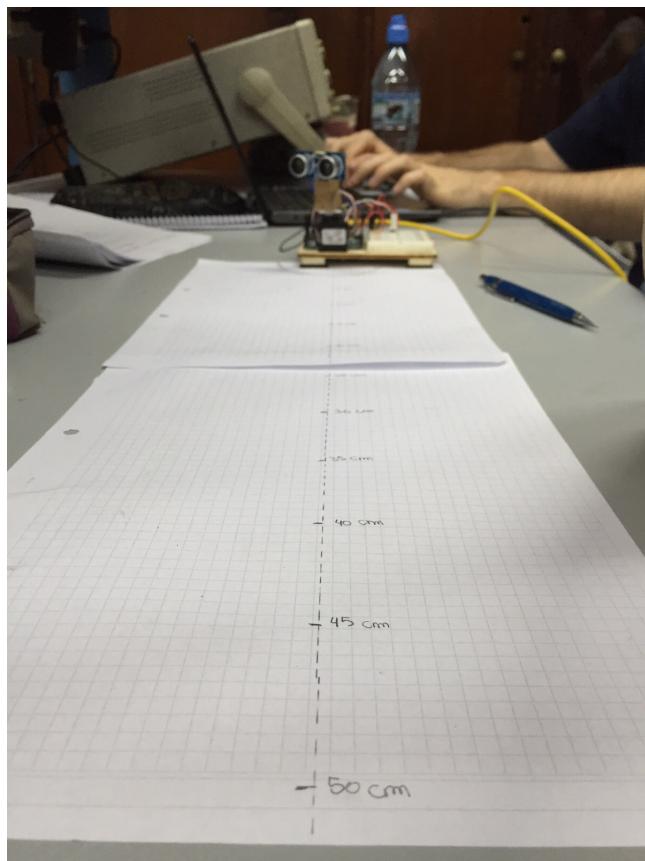


Figura 3: Mediciones de calibración

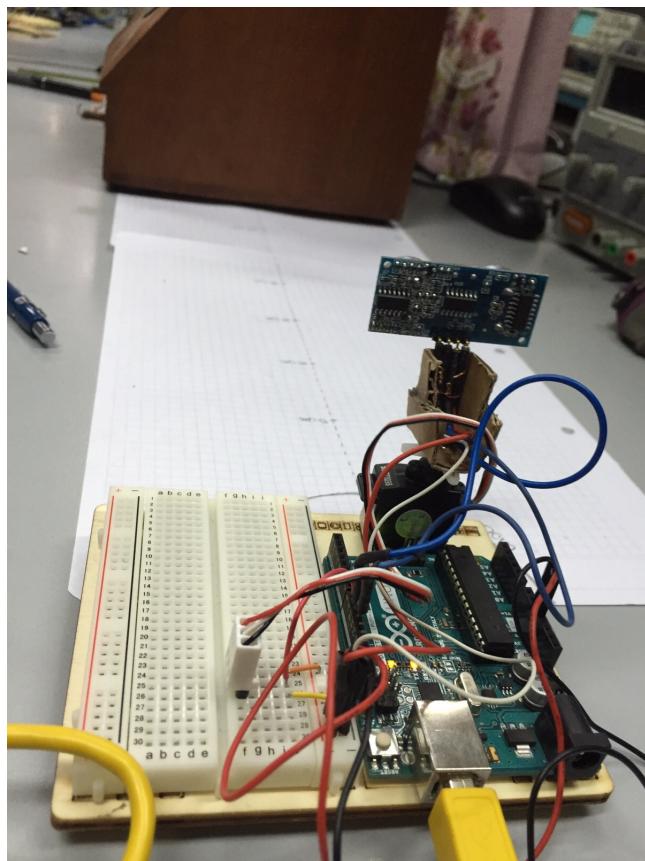


Figura 4: Mediciones de calibración

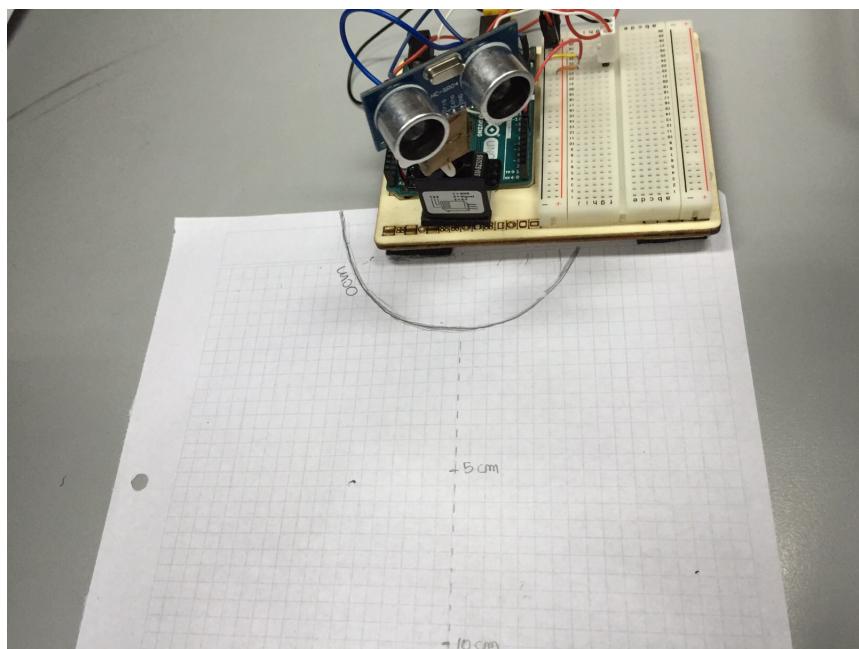


Figura 5: Mediciones de calibración

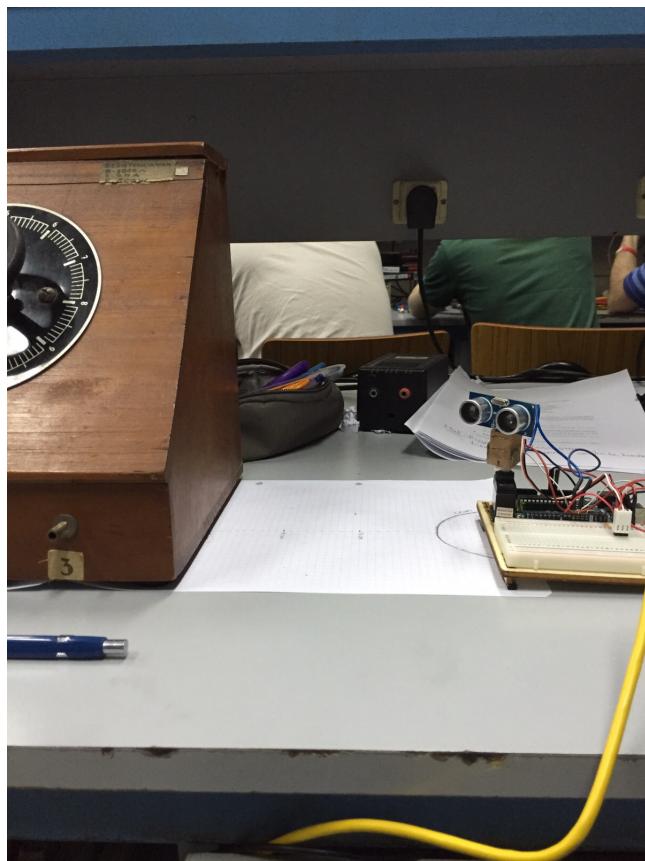


Figura 6: Mediciones de calibración

Otra de las dificultades encontradas fue determinar como realizar las mediciones sobre el radar utilizando los instrumentos de laboratorio. Dado que se buscaba medir, entre otras cosas, el consumo total del radar, de los distintos pines utilizados y la señal del servomotor se debió, primero, lograr conectar los componentes del radar con los instrumentos necesarios para realizar cada medición. Una vez logrado esto, se debieron aplicar los conocimientos aprendidos en la materia para poder utilizar cada instrumento de manera correcta y así lograr mediciones fidedignas. También se presentan una serie de imágenes que ilustran algunos de los resultados obtenidos. Se debe tener en cuenta de que todas estas imágenes junto con los resultados se explican con mayor detalle en la sección de mediciones.

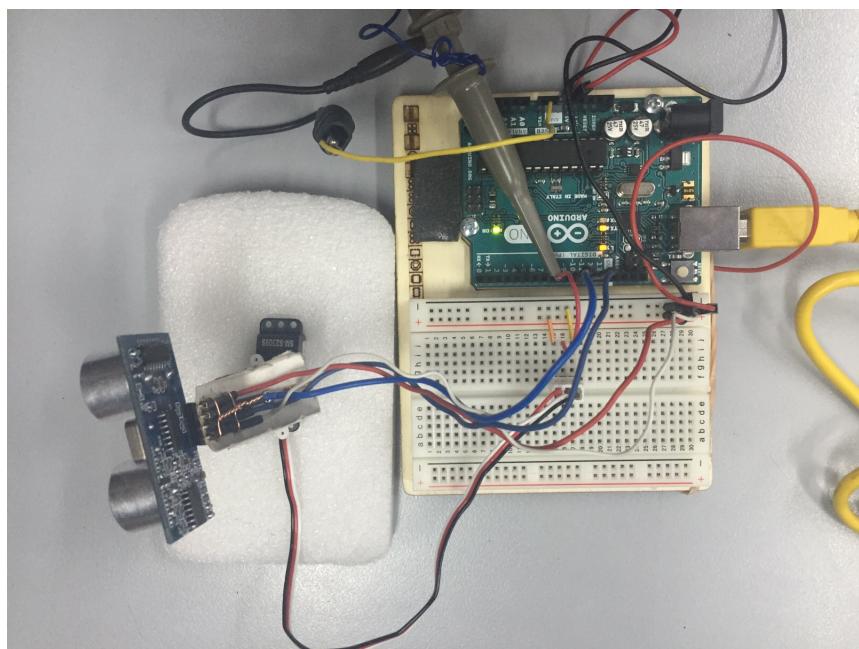


Figura 7: Mediciones sobre el radar

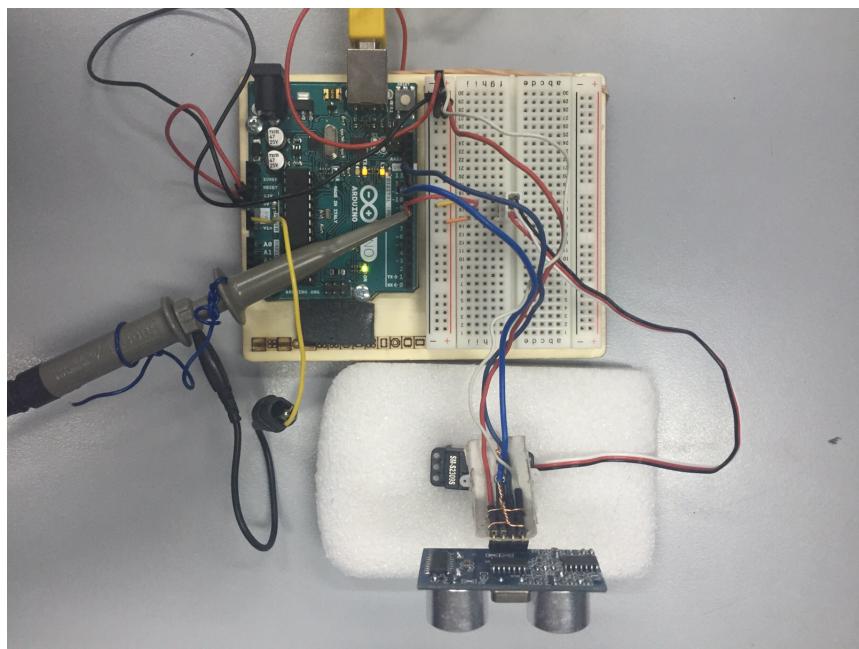


Figura 8: Mediciones sobre el radar

5.3. Análisis teórico

El dispositivo funciona en base a un sensor de ultrasonido. Para comprender el funcionamiento de este dispositivo, por lo tanto, es necesario comprender cómo funciona el sensor y cómo interactúa con el microcontrolador. Esta sección se dedicará exclusivamente a la introducción y explicación de los conceptos básicos que el lector debe poseer para entender el funcionamiento del dispositivo en conjunto con el software.

5.3.1. El sensor

El sensor ultrasónico utilizado puede verse en la siguiente figura:

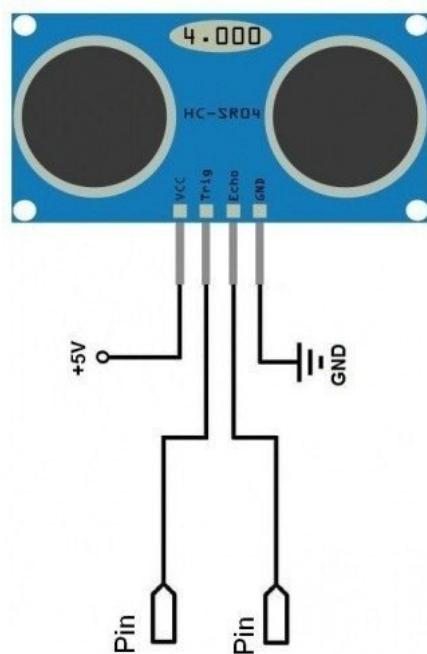


Figura 9: Sensor ultrasónico

El lector observará que en la parte inferior del sensor hay 4 entradas/salidas diferentes. La primera salida recibe el nombre VCC, y es el canal por el cual el sensor recibe una tensión de 5V.

Las entrada Trig y salida Echo son las que le permiten al microcontrolador manejar el funcionamiento del sensor. Observemos con detenimiento el siguiente diagrama:

La conocida GND es la conexión a tierra.

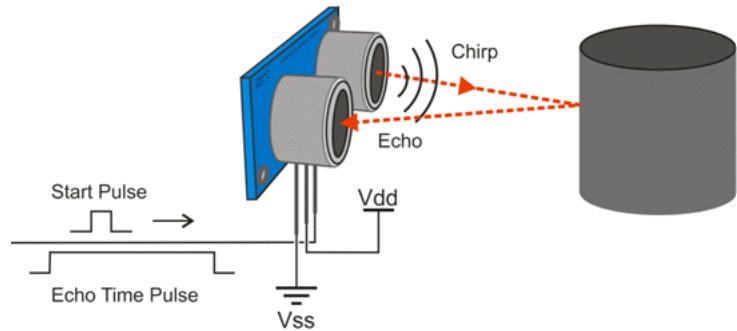


Figura 10: Principio de funcionamiento sensor ultrasónico

Lo que ilustra la figura es lo que sucede cuando el sensor detecta la presencia de un objeto frente a él. El emisor del sensor es la parte encargada de emitir una onda de sonido. El pin denominado Trig es el canal por el cual el microcontrolador le indica al emisor (a través de un pulso) que puede disparar la onda. Una vez que se emitió la onda, se activa el receptor del sensor, que se mantiene activo hasta que recibe el rebote de la onda emitida o hasta que se cumple el tiempo máximo de espera. Este tiempo máximo está determinado por la distancia máxima a la cual el sensor puede detectar objetos en forma confiable. En el caso del sensor en cuestión la distancia máxima especificada es de 4 m. El pin Echo será el canal por el cual se envíe el pulso cuyo ancho dependerá del tiempo que tarde la onda en volver, es decir, cuando el sensor capta el rebote de la onda, se produce un flanco descendente en el pulso transmitido por el pin Echo. El ancho de pulso permite conocer el tiempo que tardó la onda en ir y volver, y así se puede calcular la distancia a la cual se encuentra el objeto detectado según la ecuación:

$$Distancia = \frac{(TiempoTrig - TiempoEcho) * (VelocidadSonido)}{2} \quad (1)$$

Para mayor detalle sobre el funcionamiento del sensor adjuntamos la hoja de datos del mismo donde además el lector podrá apreciar un diagrama de tiempos que ilustra el comportamiento descripto.

5.3.2. El microcontrolador

Un microcontrolador es un circuito integrado programable capaz de ejecutar las instrucciones grabadas en su memoria. El detalle sobre el código del microcontrolador se encuentra en la sección **6.1.1**. En esta sección sólo se busca dar al lector una idea general de las tareas realizadas por el mismo. Para ello le solicitamos al lector que observe con detenimiento el siguiente diagrama:

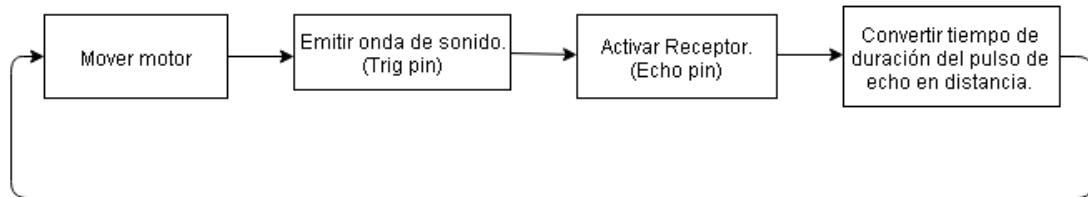


Figura 11: Ciclo de tareas microcontrolador.

5.3.3. El software

El software provisto para este dispositivo tiene la función de mostrar las distancias captadas por el sensor en forma gráfica. Se proporcionan detalles sobre su implementación y uso en la sección **6.1.2**.

6. Armado del dispositivo

Se diseño el siguiente circuito con el objetivo de poner en funcionamiento el servomotor y sensor ultrasónico que conformarán al radar.

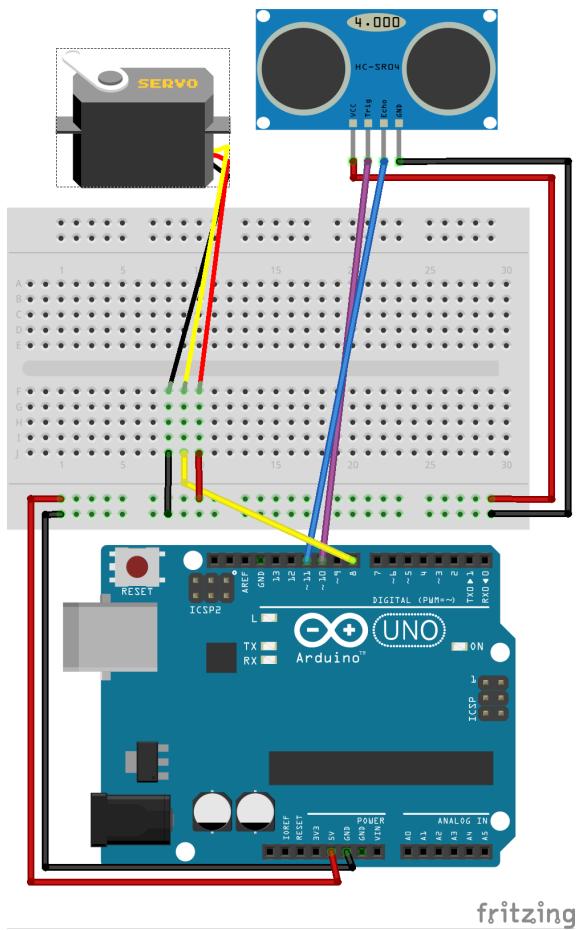


Figura 12: Esquema de las conexiones del radar

Se utilizaron cables macho-macho para conectar los pines 5 V y GND de la placa a los extremos del protoboard, creando así dos hileras en las que cómodamente se insertaron las salidas V_{cc} y GND del servomotor y sensor. Tanto el cable de control del motor, como los pines 'echo' (INPUT) y 'trigger' (OUTPUT) del sensor fueron conectados a pines PWM de la placa.

El siguiente paso fue diseñar un soporte para colocar al sensor arriba del motor. El material elegido fue cartón, y se confeccionó con el mismo una cajita para apoyar el sensor. Se utilizaron hilos de cobre como forma de conectar el cabezal del servomotor con el soporte, además de mantener al sensor fijo en su lugar. Por último, se atornilló el soporte al servo.

A continuación se ilustra el procedimiento:

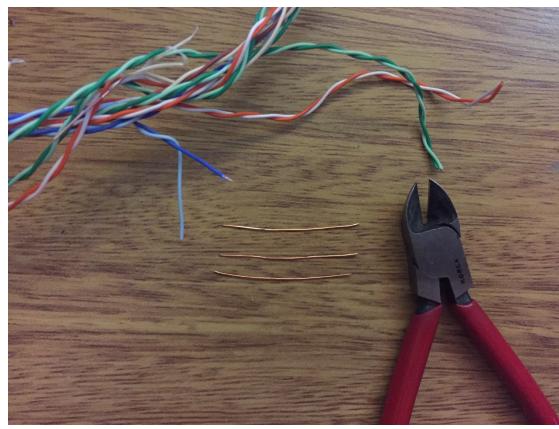


Figura 13: Hilos de cobre utilizados

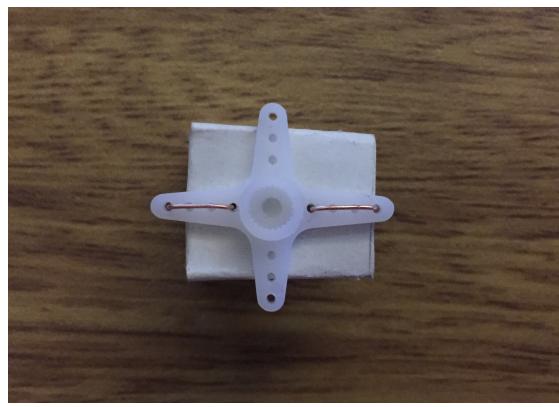


Figura 14: Cabezal del servomotor fijo al soporte, cara inferior

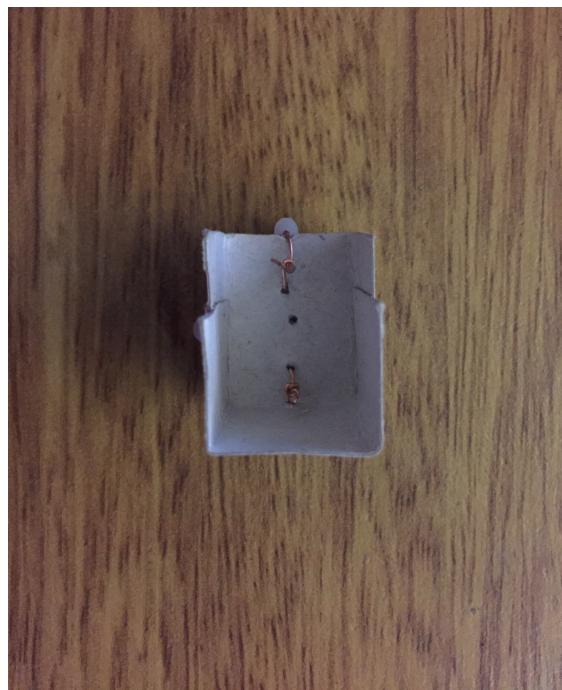


Figura 15: Cabezal del servomotor fijo al soporte, vista interior

Se realizó una prueba para verificar que el cabezal se podía atornillar al servo sin problemas.

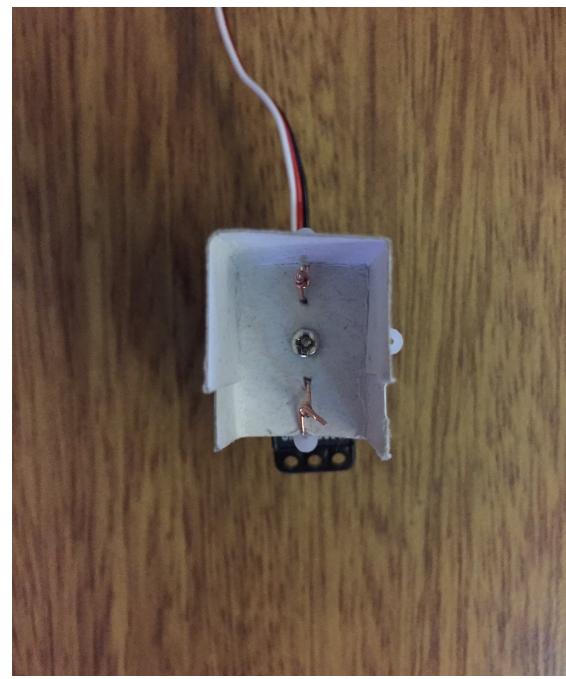


Figura 16: Servomotor con soporte visto de arriba

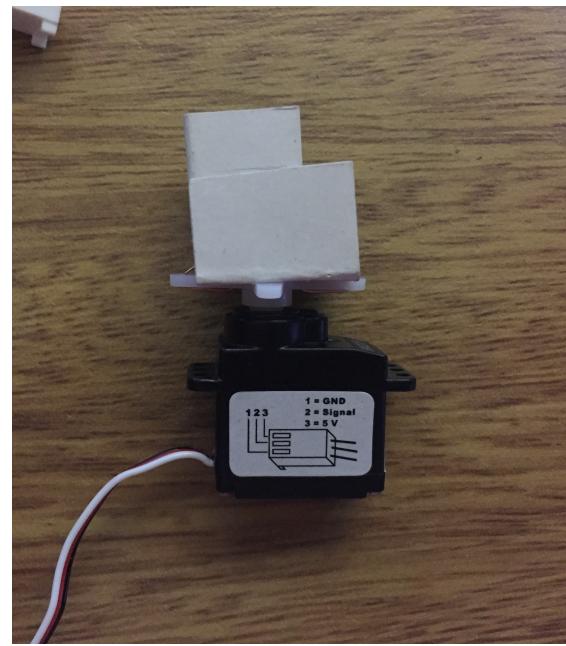


Figura 17: Servomotor con soporte, vista lateral

Se procedió desatornillando el soporte y fijando el sensor al mismo, con los mismos hilos de cobre, por medio de sus cables.



Figura 18: Sensor fijo al soporte, vista frontal



Figura 19: Sensor fijo al soporte, vista trasera

Luego, se atornilló el soporte al motor.

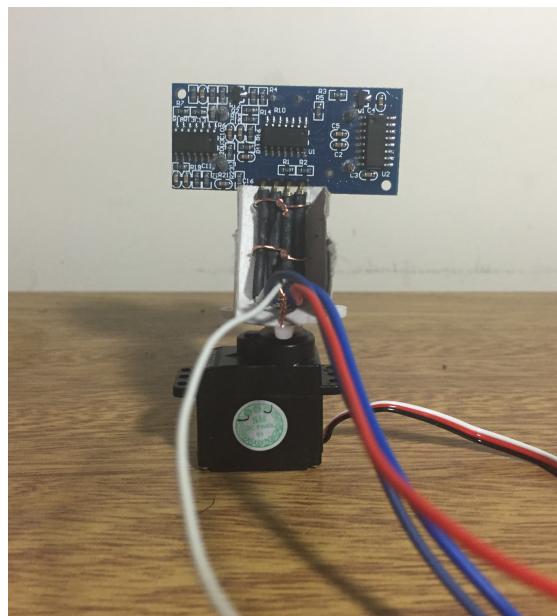


Figura 20: Conjunto de motor y sensor, vista trasera

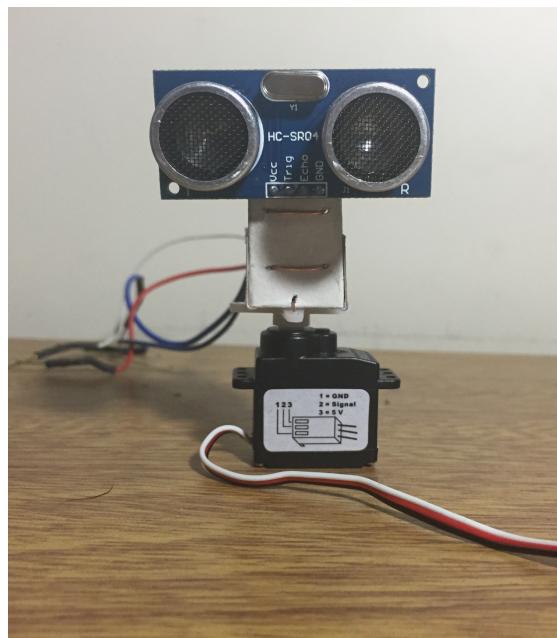


Figura 21: Conjunto de motor y sensor, vista frontal

Para terminar con el armado; se colocó el protoboard, la placa y el conjunto

formado por el motor y sensor en una base de madera.

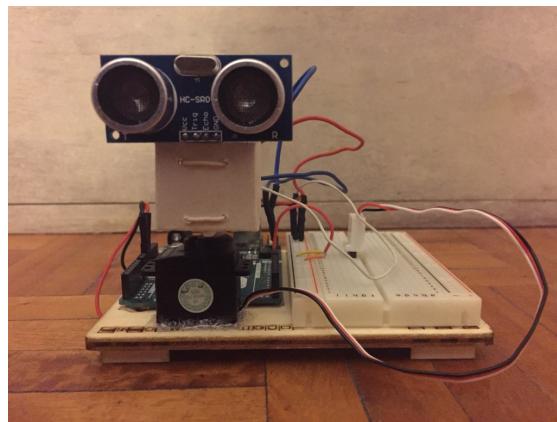


Figura 22: Instrumento final

6.1. Programación del microcontrolador de la placa Arduino y su interfaz

6.1.1. Microcontrolador

La estructura básica del lenguaje de programación de Arduino es bastante simple y se compone de al menos dos funciones. En primer lugar, la función **setup()** es la encargada de recoger la configuración y en segundo, **loop()** es la que contiene el programa que se ejecutará cíclicamente (de ahí el término **loop** –bucle–). Ambas funciones son necesarias para que el programa trabaje.

Previo a la declaración de estas funciones, se deben declarar las constantes globales a utilizar durante el programa (Número de pins a los que se encuentran conectados los sensores, etc).

La función de configuración **setup()** es la primera que se ejecuta al conectar el dispositivo, lo hace sólo una vez, y se utiliza para configurar o inicializar el modo de los pines por medio de la función **pinMode** (modo de trabajo de las entradas/salidas), la configuración de la comunicación en serie y otros.

La función bucle **loop()** contiene el código que se ejecutará continuamente (lectura de entradas, activación de salidas, etc). Esta función es el núcleo de todos los programas de Arduino y la que realiza la mayor parte del trabajo.

Se codificó un archivo **.ino** con las instrucciones a seguir por el controlador.

El código completo se encuentra adjunto en el Anexo II y en el repositorio público creado para el proyecto <https://github.com/abrdn/Acoustic-Radar>.

En el mismo, se encuentra la función **setup()** la cual inicializa el pin de Trigger con modo OUTPUT² y el pin Echo con modo INPUT³. A continuación, se identifica el pin correspondiente al motor e inicializa la comunicación serial. La función **loop()** se encarga de ejecutar el ciclo principal del programa por medio de la llamada a la función **moveServo()** para los distintos ángulos que componen el arco de giro del radar. La función **moveServo()** recibe un ángulo y se encarga de llevar al motor de la posición actual al ángulo indicado, luego de lo cual utiliza la función **calculateDistance()**. La función **calculateDistance()** proporciona al pin Trigger con un pulso de 10 μ s, luego de lo cual se utiliza la función **pulseIn()** con la cual se toma el tiempo que tarda en volver a Echo la onda emitida por el Trigger. Se transforma el tiempo del pulso que devuelve la función **pulseIn()** en la distancia entre el radar y el objeto detectado usando la función **microseconds-ToCentimeters()**. Una vez que se calculó la distancia del obstáculo (si es que hubiera alguno), la función **loop()** pasa al siguiente ángulo para iniciar el ciclo nuevamente.

6.1.2. Interfaz gráfica

Para presentar las mediciones realizadas por el instrumento fue necesario desarrollar una interfaz gráfica para la fácil visualización de la posición actual del barrido del radar y los obstáculos encontrados. La interfaz fue programada en Processing, un lenguaje de programación y entorno de desarrollo integrado de código abierto basado en Java.

Esencialmente, el programa desarrollado toma la salida de datos del puerto serie de Arduino, los cuales son el grado actual de barrido y la distancia al obtáculo detectada (Si es que lo hay) y los grafica.

A continuación se muestra una imagen de la interfaz

²Un pin configurado en modo OUTPUT se encuentran en un estado de baja impedancia, por lo que pueden proveer una cantidad substancial de corriente a otros circuitos. Fuente: <https://www.arduino.cc/en/Reference/Constants>

³Un pin configurado en modo INPUT se encuentran en un estado de alta impedancia, y equivalen a una resistencia en serie con el pin de 100M Ω s. Fuente: <https://www.arduino.cc/en/Reference/Constants>

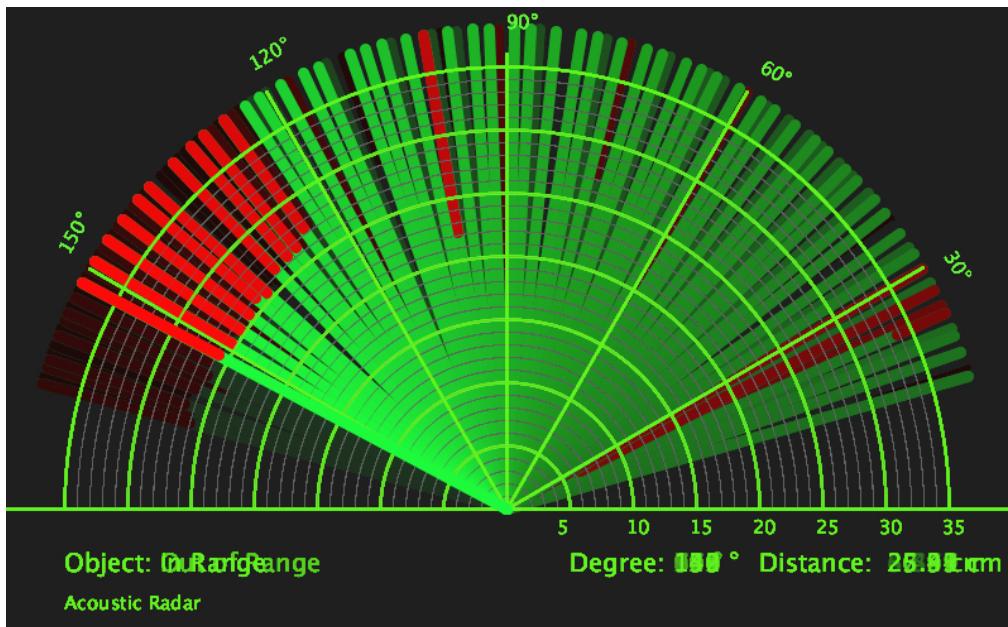


Figura 23: Captura de pantalla de interfaz gráfica

El comienzo de la línea roja indica el comienzo de un obstáculo frente al instrumento, mientras que el color verde indica la ausencia de ellos. Se agregó también el dato del grado y distancia actual debajo del área en la que se ilustra el barrido.

El código completo se encuentra en los módulos `radar_view.pde` y `Radar.pde` adjuntos en el Anexo III y en el repositorio público creado para el proyecto

<https://github.com/abrdn/Acoustic-Radar>.

6.2. Armado del banco de mediciones

Una vez que el radar estuvo correctamente ensamblado y configurado, el paso siguiente fue armar un banco de mediciones para poder probar el correcto funcionamiento del instrumento. Se eligió como cero al extremo frontal del sensor ultrasónico. Se colocó el radar sobre una hoja en donde se marcaron distintas distancias al mismo con sus magnitudes. A continuación se presentan una serie de imágenes retratando el proceso.

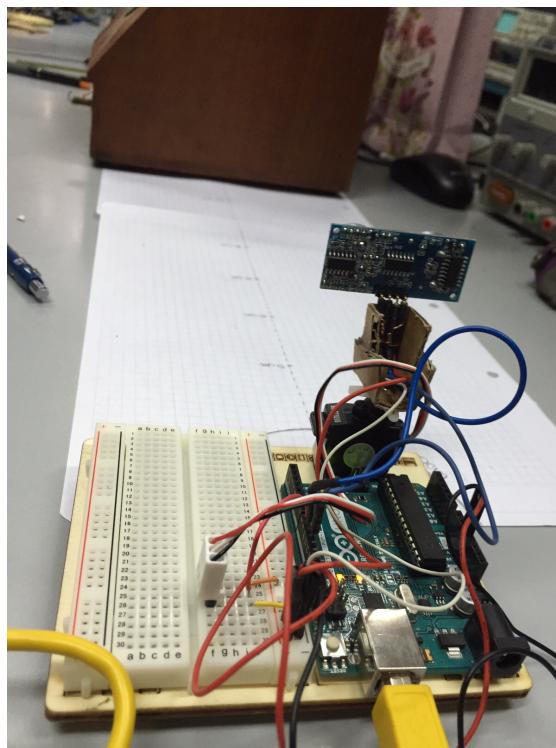


Figura 24: Primer banco de mediciones del radar

7. Mediciones

7.1. Calibración

Para calibrar el instrumento, se eligió un obstáculo y se fue posicionando a distintas distancias frente al sensor. Se tomó nota de cada distancia real y experimental.

Al observar los resultados se advirtió un error proporcional a la distancia. Se procedió a buscar dicha constante de proporcionalidad probando con diferentes números.

Los resultados de la calibración, junto con la constante de corrección, se encuentran en la subsección de resultados correspondiente.

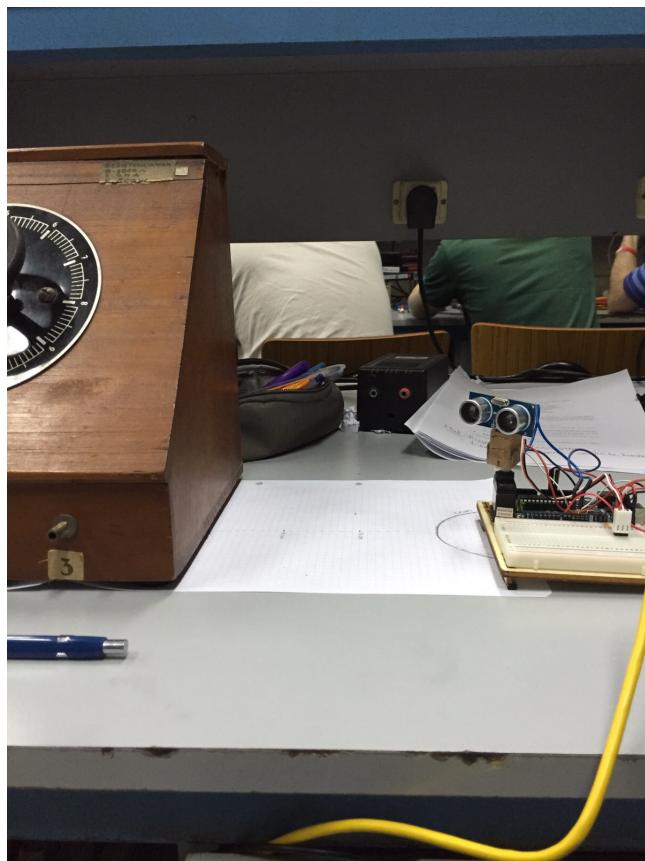


Figura 25: Toma de mediciones para calibración

7.2. Prueba de campo

Una vez calibrado el radar, se realizó una prueba de campo con diferentes obstáculos, a diferentes distancias y ángulos. A continuación se presentan algunas fotos.

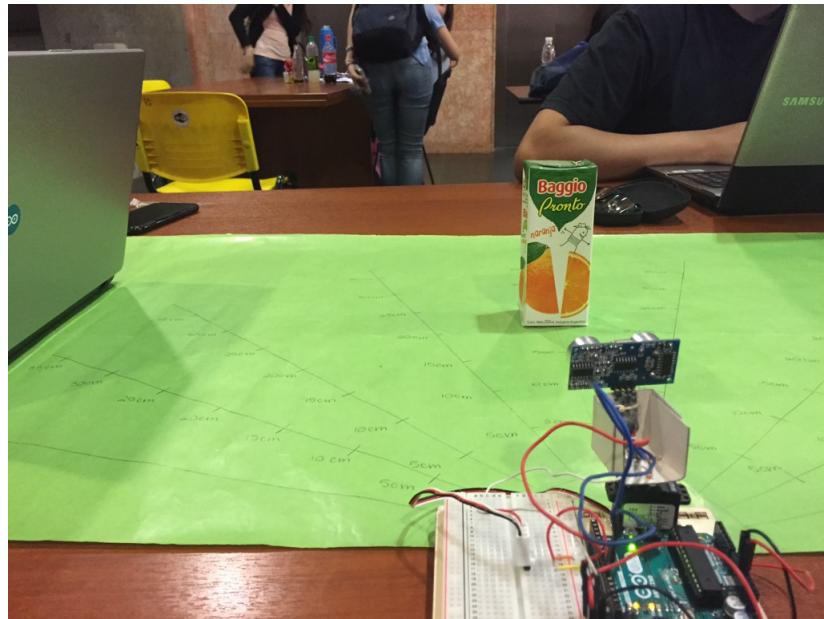


Figura 26: Toma de mediciones para prueba de campo

Los resultados se encuentran en la subsección de resultados correspondiente.

7.3. Mediciones propias del instrumento

Como parte del proyecto se nos pidió realizar una medición por cada instrumento con el que trabajamos durante la materia. Las principales mediciones realizadas fueron las siguientes:

- Visualización de la tensión del pin de control del servomotor en osciloscopio.
- Medida de frecuencia y período del pin de control del servomotor en contador.
- Medida de tensión eficaz y media de la onda del pin de control del servomotor con voltímetro digital.
- Medida de tensión con la que se alimenta al instrumento completo con voltímetro analógico.
- Medida de corriente consumida por el instrumento completo con amperímetro digital.

Lamentablemente, las señales Trigger y Echo que más nos interesaba medir fueron codificadas por la placa Arduino y resultaron imposibles de visualizar de forma clara en el osciloscopio.

Los resultados de las mediciones se encuentran en la subsección de resultados correspondiente.

8. Setups de ensayo y mediciones

En esta sección se presentan los esquemas de las conexiones realizadas en el laboratorio con el objetivo de llevar a cabo las mediciones sobre el radar. Las especificaciones de cada setup (marca, modelo, configuraciones de instrumentos utilizados, etc) se repiten en la sección de resultados.

8.1. Tensión de alimentación del instrumento

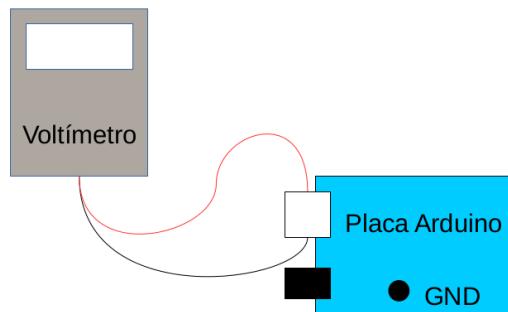


Figura 27: Diagrama de conexiones: tensión entrada de alimentación USB

- Multímetro analógico Konstar, modelo KS-803

- Alcance: 12 V
- Número de divisiones: 60
- Conectado a pin de alimentación de entrada USB.

El cable USB posee 4 buses internos, uno es de alimentación, otro es tierra y los otros dos son buses de datos. Gracias a ello, nos fue posible encontrar en la parte inferior de la placa 4 terminales correspondientes a estos buses. Debido a que las mismas no tienen identificación alguna, para identificar cuáles eran las dos terminales correspondientes a los datos y cuáles eran las terminales correspondientes a GND y tensión probamos colocando las terminales del voltímetro sobre cada una de ellas hasta encontrar el par sobre el cual la tensión se mantenía estable. Una vez hecho esto nos dispusimos a registrar el valor mostrado por el voltímetro. El valor promedio medido fue de 5,05 V con el voltímetro digital y 5,6V con el voltímetro analógico. Dado que el error del voltímetro analógico es de $\pm 0,4$ V podemos decir que el valor obtenido se aproxima al valor teórico de alimentación de la placa, según su manual, de 5 V.

8.2. Consumo de corriente del instrumento

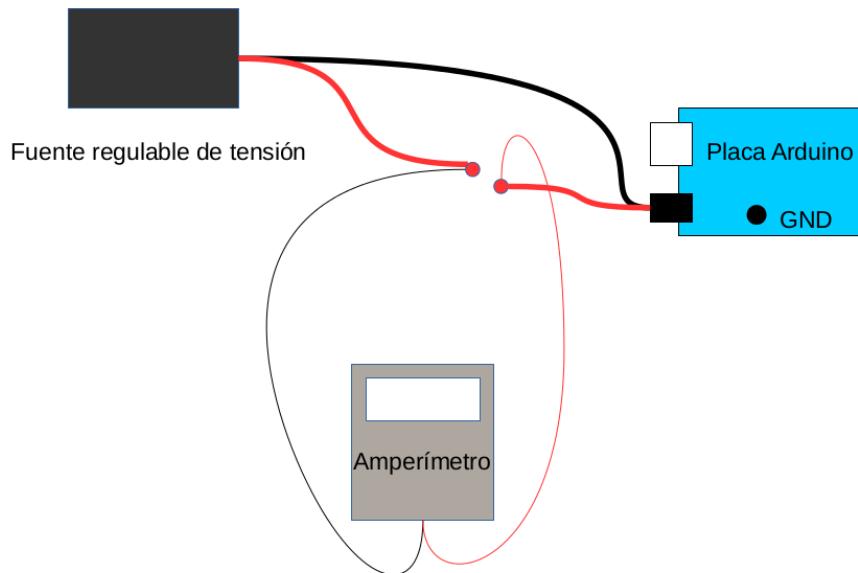


Figura 28: Diagrama de conexiones: corriente salida fuente de alimentación externa

- Multímetro digital (DVM) Protomax, modelo MS8221A.
- Alcance: 20 mA
- Número de cuentas: 2000
- Conectado en serie a entrada jack de alimentación.

Para esta medición tuvimos la dificultad de necesitar conectar un amperímetro en serie para poder medir la corriente consumida por el aparato completo, así es que conseguimos un cable con la terminación apropiada para la entrada de alimentación externa (jack), lo conectamos al amperímetro y éste, a su vez, a la fuente de tensión. Una vez realizada la conexión le proporcionamos a la placa una tensión de aproximadamente 5,4 V por ser este el valor medido previamente

(Aquí se cometió un error, pues la tensión que se le debe proporcionar a la placa por USB no es la misma que la del jack, se explica detalladamente en el Anexo IV: Correcciones). Se encontró que los valores de corriente medidos fueron muy variables. En promedio el valor fue de 93,5 mA. Teniendo en cuenta que el valor de consumo máximo es de 1 A cuando la placa es alimentada por una fuente externa, consideramos que los resultados fueron satisfactorios.

8.2.1. Salida de fuente de alimentación externa

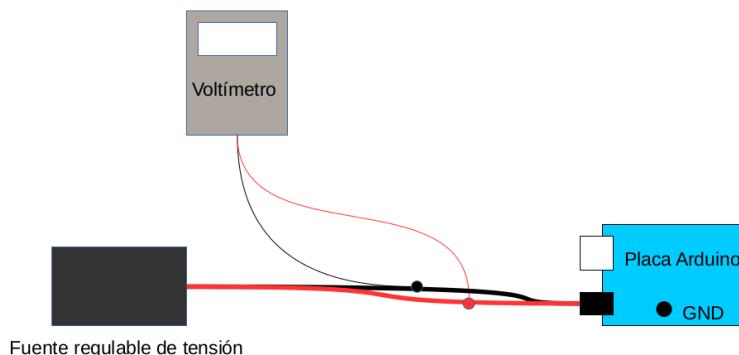


Figura 29: Diagrama de conexiones: tensión salida a fuente de alimentación externa

- Multímetro True RMS Checkman, modelo TK-4002
- Alcance: 20 V
- Número de cuentas: 2000
- Conectado en paralelo a jack de alimentación.

Para esta medición utilizamos el mismo cable (jack) de la sección anterior, sólo que esta vez conectamos un voltímetro en paralelo para medir la tensión de

entrada. La tensión medida, en este caso, es de aproximadamente 5,41 V, lo cual supera el valor esperado aún considerando el error sistemático introducido por el voltímetro.

8.3. Tensión del pin 'trigger'

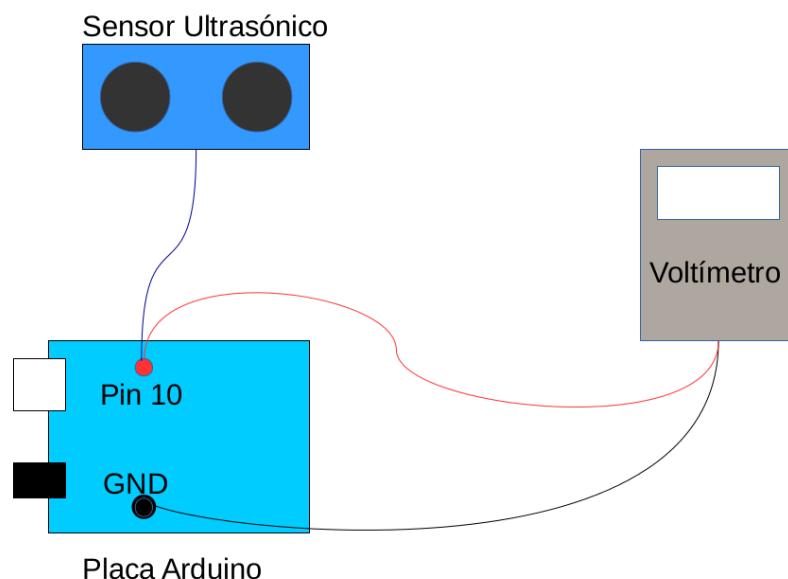


Figura 30: Diagrama de conexiones: tensión sobre el pin 'trigger'.

- Multímetro True RMS Checkman, modelo TK-4002
- Alcance: 20 mV
- Número de cuentas: 2000
- Pin Trigger (10)

Para esta medición lo que hicimos fue usar cables para poder realizar la conexión entre los pines y las terminales del voltímetro (porque los pines son pequeños y resulta imposible conectar una terminación cocodrilo directamente sobre el pin).

Una vez realizada la conexión se midieron múltiples valores de tensión y se registró que la tensión promedio de este pin fue de aproximadamente 15,74 mV.

8.4. Tensión del pin 'echo'

Las conexiones fueron iguales a las de la medición de tensión de entrada sobre el pin 'trigger', excepto que debió conectar el voltímetro al pin 11, que corresponde al pin 'echo'. Una vez realizada la conexión se midieron múltiples valores de tensión (más que para el pin 'trigger' porque los valores medidos resultaron ser muy variables) y se registró que la tensión promedio de este pin era de aproximadamente 75,03 mV.

- Multímetro True RMS Brymen, modelo BM837RS
- Alcance: 400 mV
- Número de cuentas: 4000
- Pin Trigger (11)

Tensión del pin de control del servomotor

Las conexiones fueron iguales a las de la medición de tensión de entrada sobre el pin 'trigger', excepto que debió conectar el voltímetro al pin 8, que corresponde al pin de control del servomotor. Una vez realizada la conexión se midieron múltiples valores de tensión media y eficaz para esta señal cuadrada colocando el voltímetro en los modos DC y AC respectivamente. Se registró un valor medio de 234.9 mV y eficaz de 1.118 V.

Para tensión media:

- Multímetro True RMS Brymen, modelo BM837RS.
- Alcance: 400.0 mV.
- Número de cuentas: 40000.
- Modo DC
- Pin de control del servomotor (8).

Para tensión eficaz:

- Multímetro True RMS Brymen, modelo BM837RS.
- Alcance: 4.000 V.
- Número de cuentas: 40000.
- Modo AC
- Pin de control del servomotor (8).

8.5. Visualización de señal de control del servomotor

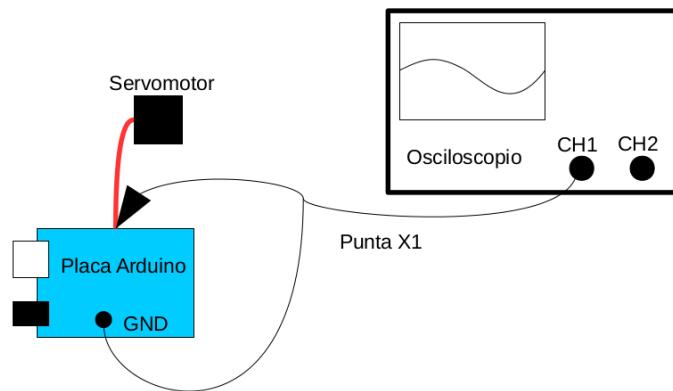


Figura 31: Diagrama de conexiones: señal de entrada del servomotor con Osciloscopio.

- Osciloscopio GoodWill, modelo GOS-653G.
- Volts por división: 2 V/div
- Base de tiempo: 5 ms/div
- Punta X1
- Punta conectada a pin de control del servomotor (8)

Para realizar esta medición se conectó la punta del osciloscopio al pin correspondiente al motor. Se visualizó una señal cuadrada, lo cual era esperable, dado que el movimiento del motor es discreto. De la señal visualizada en la pantalla pudimos medir un período de 20ms y una tensión pico-pico de 4,6 V.

8.6. Frecuencia y período de la señal de control del servomotor

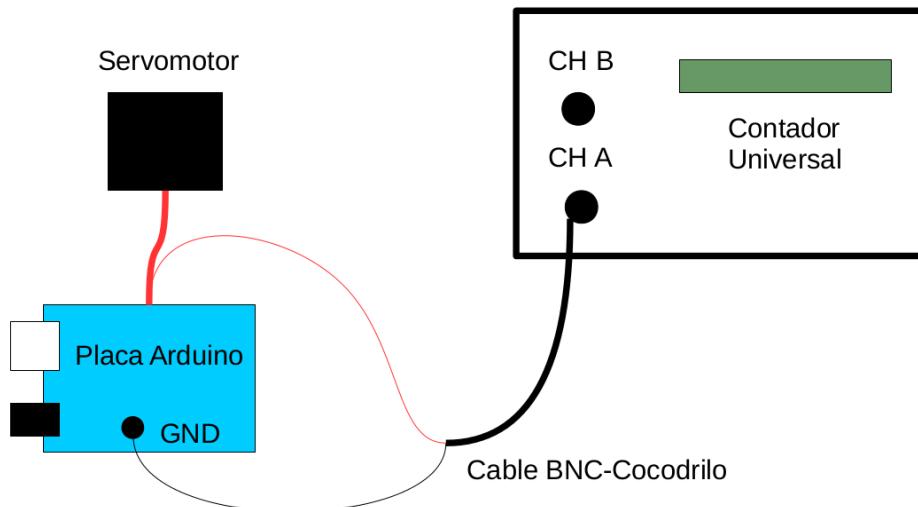


Figura 32: Diagrama de conexiones: frecuencia y período de señal del servomotor con Contador Universal.

- Contador universal GoodWill, modelo GUC-2020
- Gate Time: 1s
- Modo: FREC o PRD, según el caso.
- Cocodrilo conectado a pin de control del servomotor (8)

Para esta medición simplemente se conectó la señal de salida del motor al canal A del contador, como se muestra en la figura. Para medir utilizamos tanto el modo

período como el modo frecuencia con un Gate Time de 1s. La frecuencia medida fue de 0,05 kHz mientras que el período medido fue de 0,02 s. Los resultados coinciden con las mediciones obtenidas con el osciloscopio.

9. Resultados

9.1. Calibración

Esta sección corresponde a los resultados obtenidos en la sección 7.1. Se realizó una serie de mediciones con el objetivo de probar la exactitud del instrumento. Las mismas se realizaron posicionando un objeto a 90 grados y variando su distancia al 0 del sensor. Se encontró que la distancia medida fue siempre superior a la real, y se intentó corregir este error proponiendo diversas constantes simbolizando qué porción de esta medición es la distancia real. A continuación se presenta una serie de tablas exponiendo distancia real, medida y corregida (según la constante final) de un objeto frente al radar. Cabe aclarar que ambos valores, el medido sin corrección y con corrección fueron leídos directamente del puerto Arduino. Se presentan los datos con todos los decimales devueltos.

Objeto a 5 cm.

Nro.	Valor medido sin corrección (cm)	Valor medido con corrección (cm)
1	7.48	5.61
2	7.55	5.66
3	7.62	5.72
4	7.55	5.66
5	7.38	5.53

Objeto a 10 cm.

Nro.	Valor medido sin corrección (cm)	Valor medido con corrección (cm)
1	13.29	9.97
2	13.90	10.42
3	13.28	9.96
4	13.88	10.41
5	13.28	9.96

Objeto a 15 cm.

Nro.	Valor medido sin corrección (cm)	Valor medido con corrección (cm)
1	20.38	15.28
2	20.29	15.22
3	20.38	15.28
4	20.31	15.23
5	20.52	15.39

Objeto a 20 cm.

Nro.	Valor medido sin corrección (cm)	Valor medido con corrección (cm)
1	26.78	20.08
2	26.84	20.13
3	25.84	19.38
4	26.53	19.90
5	26.81	20.11

Objeto a 25 cm.

Nro.	Valor medido sin corrección (cm)	Valor medido con corrección (cm)
1	33.21	24.91
2	33.26	24.94
3	33.21	24.91
4	33.34	25.03
5	33.33	25.00

Objeto a 30 cm.

Nro.	Valor medido sin corrección (cm)	Valor medido con corrección (cm)
1	39.53	29.65
2	40.02	30.01
3	40.09	30.06
4	40.02	30.01
5	39.53	29.65

Objeto a 35 cm.

Nro.	Valor medido sin corrección (cm)	Valor medido con corrección (cm)
1	46.36	34.77
2	46.36	34.77
3	46.36	34.77
4	46.36	34.77
5	46.45	34.84

Objeto a 40 cm.

Nro.	Valor medido sin corrección (cm)	Valor medido con corrección (cm)
1	52.45	39.34
2	52.45	39.34
3	52.40	39.30
4	52.98	39.74
5	52.78	39.58

Objeto a 45 cm.

Nro.	Valor medido sin corrección (cm)	Valor medido con corrección (cm)
1	59.21	44.41
2	59.26	44.44
3	59.26	44.44
4	59.26	44.44
5	59.12	44.34

Objeto a 50 cm.

Nro.	Valor medido sin corrección (cm)	Valor medido con corrección (cm)
1	65.67	49.25
2	65.74	49.31
3	65.66	49.24
4	65.72	49.29
5	64.79	48.59

Es posible apreciar que existe un error de offset en las mediciones realizadas. Se observó que la incidencia del error fue disminuyendo a medida que las distancias aumentaron. Para continuar con el trabajo se eligió un rango de distancias en el cual los errores cometidos fueron aproximadamente iguales y se corrigió ese error. Es importante remarcar que este proceso de corrección se puede hacer con múltiples intervalos dentro del rango del sensor ultrasónico (2 a 400 cm), nosotros simplemente elegimos uno sobre el cual nos resultó fácil medir y trabajar en profundidad.

Una de las fuentes de error posible es que la referencia al 0 de las distancias fue impuesta por nosotros mismos, al no estar especificada en la ficha de datos del sensor ultrasónico. Esta referencia fue determinada como la circunferencia descrita por la proyección del sensor en movimiento, en el plano donde se colocan los obstáculos. Este último pudo haber sido uno de los factores determinantes de la necesidad de una calibración del instrumento, en conjunto con otras fuentes de error razonables que consideraremos al expresar la incerteza de la medición.

Dado que pudimos observar que el error producido mantenía una relación directamente proporcional a la distancia medida, decidimos realizar una corrección sobre la medición devuelta por el sensor.

En forma experimental determinamos que para el rango de 10 cm a 35 cm el error se puede corregir de la siguiente forma: $X_{medido} = K_L * X_{devuelto-por-sensor}$ donde $K_L = \frac{3}{4}$.

9.1.1. Incerteza de medición

Para obtener la incerteza de las mediciones realizadas deben tenerse en cuenta distintos aspectos. Por un lado, el error cometido debe considerar los errores introducidos por el sensor (error sensor). Por otro lado, debemos tener en cuen-

ta el error que puede surgir del armado del banco de medición (error banco), así como también de la comparación de la medición del sensor con la magnitud que se supone es la correcta respecto del banco de medición (error de apreciación). Proponemos entonces la siguiente aproximación:

$$\Delta X = \Delta X_s + \Delta X_{banco} + \Delta X_{ap} \quad (2)$$

Para el error del sensor tomaremos la ecuación para obtener la distancia medida utilizada en el código del controlador:

$$Distancia = \frac{t}{v_{sonido}} * \frac{1}{2} * \frac{3}{4}$$

Siendo t el tiempo que tarda el volver al receptor la onda rebotada (Medido en microsegundos por la función **pulseIn()**) y v_{sonido} el valor de la velocidad del sonido en microsegundos por centímetros (Tomada como $29cm/\mu s$).

Según la documentación de la función **pulseIn()**⁴, la misma trabaja correctamente para un intervalo de tiempo entre pulsos de duración de $10\mu s$ a 3 minutos. Procedemos a calcular cual sería la duración de los pulsos para nuestro intervalo de distancias de 10 a 35 cm.

Despejando la ecuación obtenemos:

$$t = \frac{Distancia * 2 * v_{sonido}}{3}$$

Para 10 cm, el tiempo de espera hasta la llegada de la onda rebotada será de $193,33\mu s$, mientras que para 35 cm será de $676,67\mu s$.

$$t_{max} = \frac{2 * 35cm * 29 \frac{\mu s}{cm}}{3} = 676,67\mu s$$

$$t_{min} = \frac{2 * 10cm * 29 \frac{\mu s}{cm}}{3} = 193,33\mu s$$

Podemos observar, entonces que las distancias medidas se mantienen dentro del intervalo de trabajo de la función **pulseIn()**, por lo que no se considerará la incertezza correspondiente a la medición del intervalo de tiempo.

⁴<https://www.arduino.cc/en/Reference/PulseIn>

Para determinar la incerteza asociada a la velocidad del sonido, tendremos en cuenta su variación de acuerdo con la temperatura. La velocidad del sonido varía según:

$$v_{Sonido} = \sqrt{\frac{\gamma * R * T}{M}}$$

Donde:

- $\gamma = 1.4$ (para el aire).
- $R = 8.314 \frac{J*K}{mol}$.
- $M = 0.029 \frac{kg}{mol}$ (para el aire).

Para la temperatura, consideraremos un rango de variación de 20°C (o 293.15K) a 25°C (o 298.15K), con el cual calcularemos un rango para la velocidad del sonido.

A T=293.15K:

$$v_{Sonido} = \sqrt{\frac{1,4 * 8,314 \frac{J*K}{mol} * 293,15K}{0,029 \frac{kg}{mol}}} = 343,02 \frac{m}{s}$$

A T=298.15K:

$$v_{Sonido} = \sqrt{\frac{1,4 * 8,314 \frac{J*K}{mol} * 298,15K}{0,029 \frac{kg}{mol}}} = 345,93 \frac{m}{s}$$

Por lo tanto definimos $\Delta v_{Sonido} = v_{Sonido}(T = 298,15K) - v_{Sonido}(T = 293,15K) = 345,93 \frac{m}{s} - 343,02 \frac{m}{s} = 2,91 \frac{m}{s}$.

Sabiendo que $\frac{1m}{s} = \frac{10000\mu s}{cm}$ utilizaremos esta incerteza como $\Delta V_{Sonido} = 2,91 \times 10^{-4} \frac{cm}{\mu s}$.

Tomando como incerteza de la velocidad del sonido, propagamos el error para el cálculo de la distancia y tomamos la incerteza del sensor como:

$$\Delta X_s = \frac{3 * t}{2 * v_{sonido}^2} * \Delta v_{sonido}$$

Por lo tanto

$$\Delta X_s = \frac{3 * t}{2 * (\frac{29cm}{\mu s})^2} * 3 \times 10^{-4} \frac{cm}{\mu s} \quad (3)$$

Para el error cometido por el armado del banco de medición, decidimos tomar como error un valor de 1 mm, debido a que se utilizó una regla para su confección y debemos considerar la posibilidad de que haya existido algún error en el marcado de los puntos de referencia o en las mediciones realizadas para tal fin.

$$\Delta X_{banco} = 0,1cm$$

Por último asignamos como incerteza de apreciación de forma análoga a como hacemos con los multímetros analógicos:

$$\Delta X_{ap} = \frac{35cm}{4 * 35div} = 0,25cm$$

Obtenemos entonces el siguiente valor de incerteza:

$$\Delta X = \Delta X_s + \Delta X_{banco} + \Delta X_{ap} = \frac{3 * t}{2 * (\frac{29cm}{\mu s})^2} * 3x10^{-4}\frac{cm}{\mu s} + 0,1cm + 0,25cm$$

Si consideramos $t_{max} = 676,67\mu s$ obtenemos $\Delta X = \pm 0,35cm$.

9.2. Prueba de campo

Se realizaron múltiples mediciones con el instrumento tomando un obstáculo y posicionandolo, sobre el banco de mediciones confeccionado, a diferentes distancias y ángulos del radar. Para cada posición se compararon las mediciones con las magnitudes reales escritas en nuestro banco de mediciones. Se tomó como ángulo medido al valor central del intervalo de ángulos en el que la interfaz muestra el objeto de prueba. A continuación se presenta un cuadro con los resultados:

Ángulo (°)	Distancia (cm)	Ángulo medido (°)	Distancia medida (cm)
90	20	90	19.8
135	20	140	20.5
70	20	80	19.8
30	10	37	10.2
150	10	149	11.0
110	10	113	10.9
150	35	154	34.4
70	35	79	34.4
110	35	98	35.0

Cuadro 1: Resultados de prueba de campo

Los resultados fueron acortados de acuerdo a la cota de incertezza previamente calculada de 0.4 cm.

Consideramos satisfactorios a los resultados obtenidos, debido a que la diferencia entre los valores teóricos y reales fue muy pequeña si tenemos en cuenta los errores que se pudieron haber cometido en el armado del banco y la toma de mediciones.

Por último, se muestran alguna de las imágenes tomadas durante la prueba

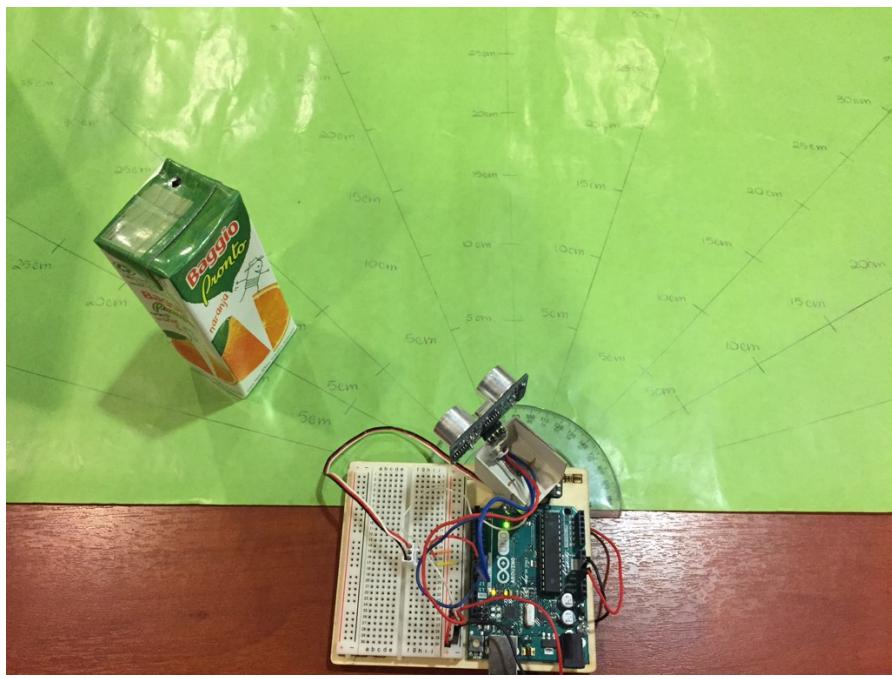


Figura 33: Obstáculo a 30 grados y 10 cm

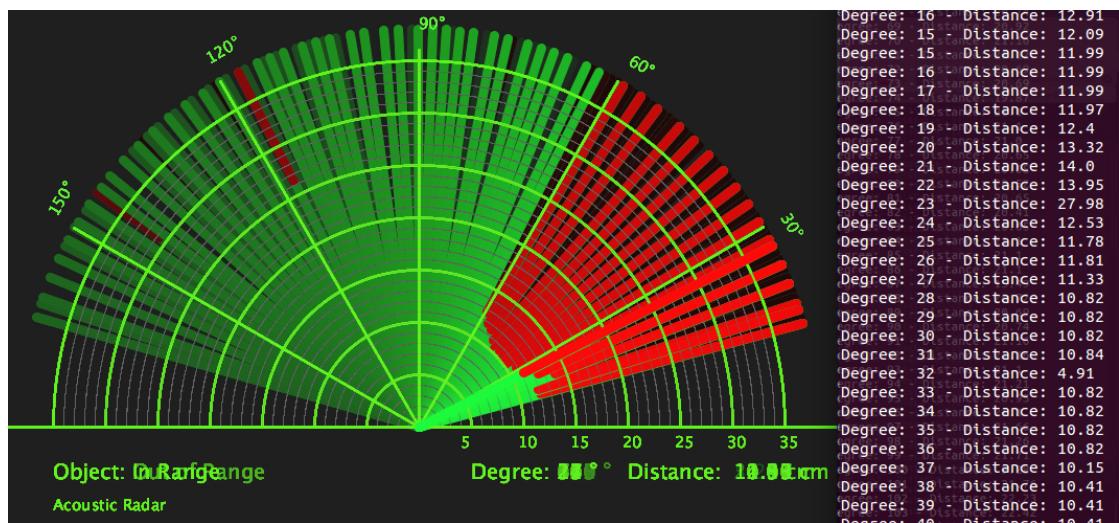


Figura 34: Medición de obstáculo a 30 grados y 10 cm

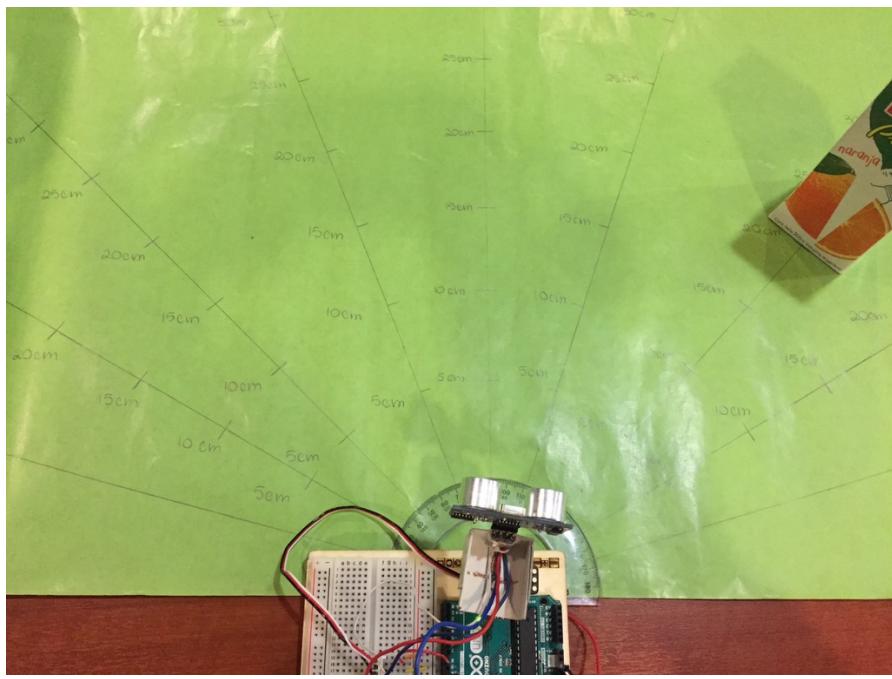


Figura 35: Obstáculo a 70 grados y 20 cm

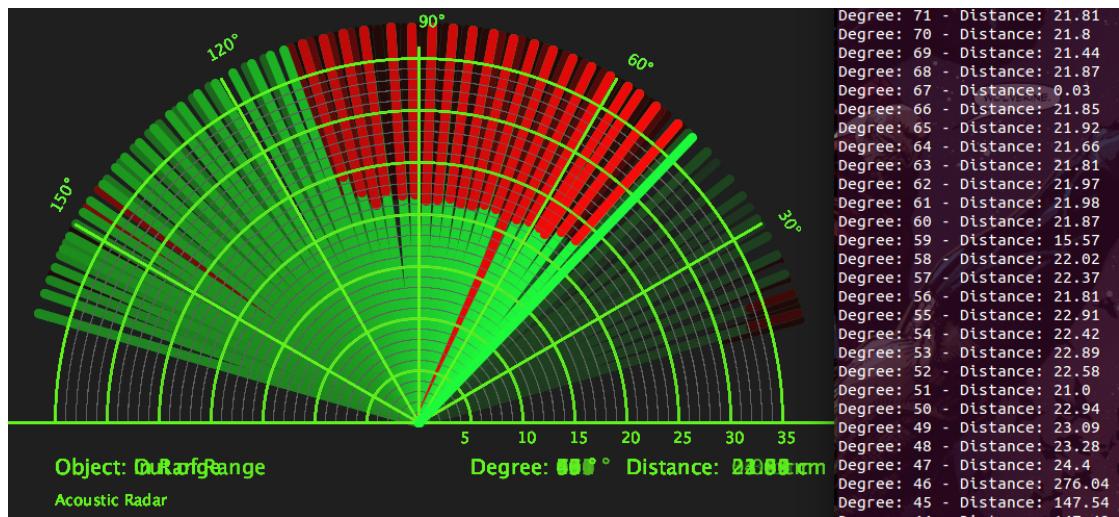


Figura 36: Medición de obstáculo a 70 grados y 20 cm

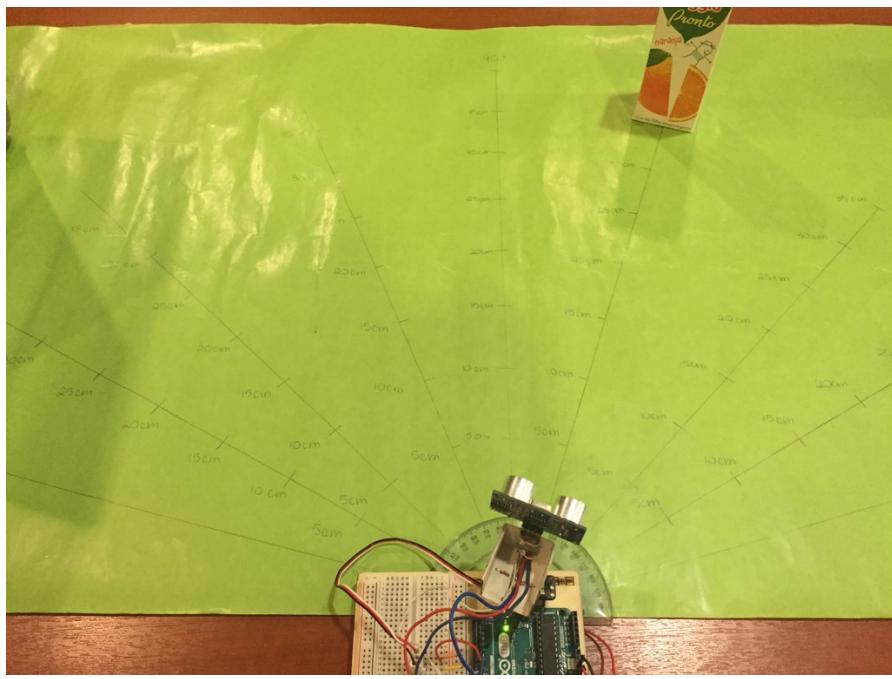


Figura 37: Obstáculo a 110 grados y 20 cm

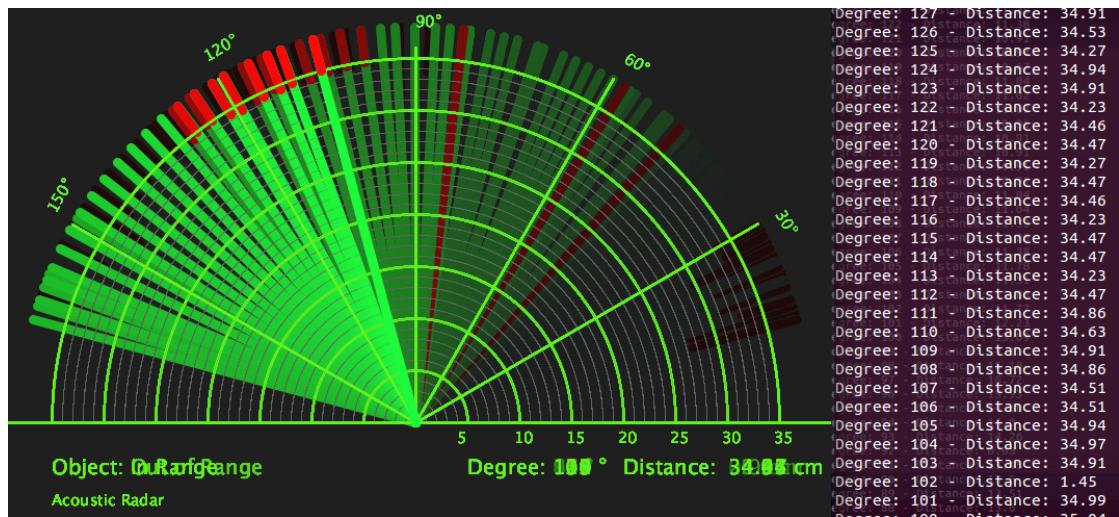


Figura 38: Medición de obstáculo a 110 grados y 35 cm

9.3. Mediciones propias del instrumento

Aquí se presentan los resultados de las mediciones realizadas en la sección 7.3. A continuación, presentaremos las mediciones referidas a la tensión de alimentación y corriente de los distintos componentes del radar. Asimismo se incluirán detalles sobre el instrumento con el que fue realizada cada medición, así como también las incertezas correspondientes al caso.

9.3.1. Tensión de alimentación del instrumento

Se midió la tensión de alimentación del instrumento a través del pin de alimentación de la entrada USB. Para ello, utilizamos el multímetro analógico **Konstar** modelo **KS-803**. Utilizando el voltímetro en la siguiente configuración:

- Alcance: 12 V
- Error de Clase: 3 %
- Divisiones: 60

El valor medido fue: 5.6 V

Calculamos la incertezza de medición según la fórmula:

$$\xi_r = \xi_m + \xi_{ap}$$
$$\xi = \frac{Alcance}{X_{medido}} * \xi_{clase} + \frac{1}{4} * \frac{Alcance}{divisiones * X_{medido}} * 100 \% \quad (4)$$

Obtenemos entonces la tensión expresada con su correspondiente incertezza:

Tensión de entrada medida con voltímetro analógico: $5.6 \text{ V} \pm 0.41 \text{ V}$.

9.3.2. Consumo de corriente del instrumento

Las siguientes mediciones fueron realizadas con un multímetro digital (DVM) **Protomax** modelo **MS8221A**. Se midió la corriente sobre la entrada de alimentación externa que, como se explicó en la sección anterior, fue alimentada por una fuente de tensión regulable del laboratorio.

Medición	1	2	3	4	5
Valor medido (mA)	90.60	87.40	86.70	98.90	103.9

Cuadro 2: Consumo de placa Arduino UNO medido con DVM

A continuación, se presentan las incertezas correspondientes a las mediciones presentadas.

Expresión de incerteza asociada a una medición realizada con el multímetro digital (DVM):

$$\Delta X = 0,5\% * X_{medido} + 2dg$$

Para conocer el valor de $2dg$ consideramos el número de cuentas del multímetro y el alcance:

A continuación, se presentan las incertezas correspondientes a las mediciones presentadas:

- Alcance: 20 mA
- Número de cuentas: 2000

$$2dg = 2 * Resolucion = 2 * \frac{Alcance}{ncuentas} = 0,2mA$$

Con estos datos elaboramos la siguiente tabla de incertezas para cada medición tomada.

Medición	1	2	3	4	5
Corriente medida (mA)	90.60 ± 0.65	87.40 ± 0.64	86.70 ± 0.63	98.90 ± 0.7	103.9 ± 0.7

Cuadro 3: Resultados mediciones corriente de entrada de placa Arduino UNO realizadas con DVM

9.3.3. Tensión del pin 'trigger'

Las siguientes mediciones fueron realizadas con un multímetro True RMS **Check-man** modelo **TK-4002**. Presentamos las mediciones realizadas con sus respectivas incertezas, las cuales fueron calculadas con la expresión correspondiente a la ecuación (4).

Configuración del multímetro:

- Alcance: 20 mV
- Número de cuentas: 2000

Por lo tanto el valor de 2dg para calcular la incerteza será:

$$2dg = 2 * Resolucion = 2 * \frac{Alcance}{ncuentas} = 00,02mV$$

Podemos presentar entonces los siguientes resultados:

Medición	Tensión (mV)
1	15.40 ± 0.097
2	15.50 ± 0.098
3	15.60 ± 0.098
4	15.80 ± 0.099
5	16.40 ± 0.102

Cuadro 4: Medida de tensión del pin 'trigger'

9.3.4. Tensión del pin 'echo'

Las siguientes mediciones fueron realizadas con un multímetro True RMS **Brymen** modelo **BM837RS**. Presentamos las mediciones realizadas con sus respectivas incertezas.

Configuración del multímetro:

- Alcance: 400.0 mV
- Número de cuentas: 4000

La expresión de incerteza asociada a una medición realizada con este multímetro digital, para este alcance, es:

$$\Delta X = 0,08 \% * X_{medido} + 1dg$$

Por lo tanto el valor de 1dg para calcular la incerteza será:

$$1dg = 1 * Resolucion = 1 * \frac{Alcance}{ncuentas} = 0,1mV$$

Podemos presentar entonces los siguientes resultados:

Medición	Tensión (mV)
1	72.9 ± 0.16
2	84.8 ± 0.17
3	75.4 ± 0.16
4	92.9 ± 0.17
5	66.9 ± 0.15
6	63.3 ± 0.15
7	39.1 ± 0.13
8	90.1 ± 0.17
9	98.0 ± 0.18
10	56.8 ± 0.15
11	42.1 ± 0.13
12	44.7 ± 0.14
13	40.7 ± 0.13
14	35.1 ± 0.13
15	100.9 ± 0.18
16	101.2 ± 0.18
17	116.9 ± 0.19
18	128.8 ± 0.20

Cuadro 5: Medida de tensión del pin echo sin una distancia predeterminada

Tenemos motivo para dudar de la confiabilidad de los datos obtenidos, ya que los datos resultaron ser dispersos en un intervalo de 39.1 mV a 128.8 mV sin mostrar una tendencia definida. Atribuimos la variabilidad de los datos al hecho de que la señal recibida por el pin en cuestión tiene picos de tensión dependiendo de los objetos que se encuentren frente al sensor, debido a que el pin 'echo' cambia de LOW a HIGH cada vez que recibe una onda de sonido rebotada.

Tensión del pin de control del servomotor

Se midió el valor medio y eficaz para la tensión de la señal del pin de control del servomotor. Las siguientes mediciones fueron realizadas con un multímetro True RMS **Brymen** modelo **BM837RS**. Presentamos las mediciones realizadas con sus respectivas incertezas.

Para la medición del valor eficaz:

Configuración del multímetro: Para tensión eficaz:

- Alcance: 4.000 V.
- Número de cuentas: 40000.
- Modo AC
- Pin de control del servomotor (8).

Se calcularon las incertezas con la siguiente fórmula provista por el manual del instrumento:

$$\Delta X = 0,8\% * X_{medido} + 8dg$$

Las mediciones obtenidas, junto a sus incertezas mayoradas se presentan en el siguiente cuadro:

Medición	V_{RMS} (V)	Incerteza (V)
1	1,48	$\pm 0,02$
2	1,23	$\pm 0,02$
3	1,07	$\pm 0,01$
4	0,871	$\pm 0,02$
5	0,933	$\pm 0,01$

Cuadro 6: Resultados para la medición de valor eficaz de tensión para el pin de control del servomotor

Para la medición del valor medio: Configuración del multímetro:

- Alcance: 400.0 mV.
- Número de cuentas: 40000.
- Modo DC
- Pin de control del servomotor (8).

Se calcularon las incertezas con la siguiente fórmula provista por el manual del instrumento:

$$\Delta X = 0,08\% * X_{medido} + 1dg$$

Las mediciones obtenidas, junto a sus incertezas mayoradas se presentan en el siguiente cuadro:

Medición	V_{medio} (mV)	Incerteza (mV)
1	215,9	$\pm 0,2$
2	285,6	$\pm 0,3$
3	232,2	$\pm 0,2$
4	127,1	$\pm 0,2$
5	314,9	$\pm 0,3$
6	342,2	$\pm 0,3$
7	130,2	$\pm 0,2$
8	187,9	$\pm 0,2$
9	326,0	$\pm 0,3$
10	186,8	$\pm 0,2$

Cuadro 7: Resultados para la medición de valor medio de tensión para el pin de control del servomotor

Se realizó el cálculo del factor de cresta de la señal en el Anexo IV: Correcciones, para compararlo con el del multímetro utilizado y así verificar si las mediciones son buenas.

9.3.5. Visualización de señal de control del servomotor

Utilizando un osciloscopio **GOODWILL** modelo **GOS-653G**, pudimos visualizar la señal en la salida correspondiente al servomotor. Dado que el motor permanece quieto durante un tiempo determinado⁵ antes de volver a moverse, era esperable que la señal observada fuera de tipo cuadrada. La señal observada poseía las características enumeradas a continuación:

⁵Tiene un delay de $30\mu s$ para que el motor tenga tiempo suficiente para llegar al nuevo ángulo además del tiempo que tarda el sensor en enviar y recibir la onda de sonido, tiempo que luego se utilizará para calcular la distancia del obstáculo.

- Período: 20 ms
- Ancho de pulso mínimo: 0.9 ms
- Ancho de pulso máximo: 2 ms
- Tensión pico-pico: 4.6 V

Estos resultados fueron obtenidos con la siguiente configuración del osciloscopio:

- Volts por división: $2 \frac{V}{div}$
- Base de tiempos: $5 \frac{ms}{div}$

Calculamos el error asociado a las mediciones tomadas utilizando la siguiente expresión:

$$\xi_{total} = \xi_{ap} + \xi_{al} + \xi_{ex} \quad (5)$$

Para el canal horizontal:

$$\xi_{total} = \frac{\text{minimadivision}}{\text{divisionesleidas}} * 100 \% + 3 \% + 3 \% = \frac{0,1}{\text{divisionesleidas}} * 100 \% + 3 \% + 3 \%$$

Para el canal vertical la expresión es similar, sólo que no consideramos error de alinealidad:

$$\xi_{total} = \frac{\text{minimadivision}}{\text{divisionesleidas}} * 100 \% + 3 \% = \frac{0,1}{\text{divisionesleidas}} * 100 \% + 3 \%$$

Podemos entonces expresar los resultados con sus respectivas incertezas:

- Período: $20 \text{ ms} \pm 2.3 \text{ ms}$
- Ancho de pulso mínimo: $0.9 \text{ ms} \pm 0.504 \text{ ms}$
- Ancho de pulso máximo: $2 \text{ ms} \pm 0.62 \text{ ms}$
- Tensión pico-pico: $4.6 \text{ V} \pm 0.338 \text{ V}$

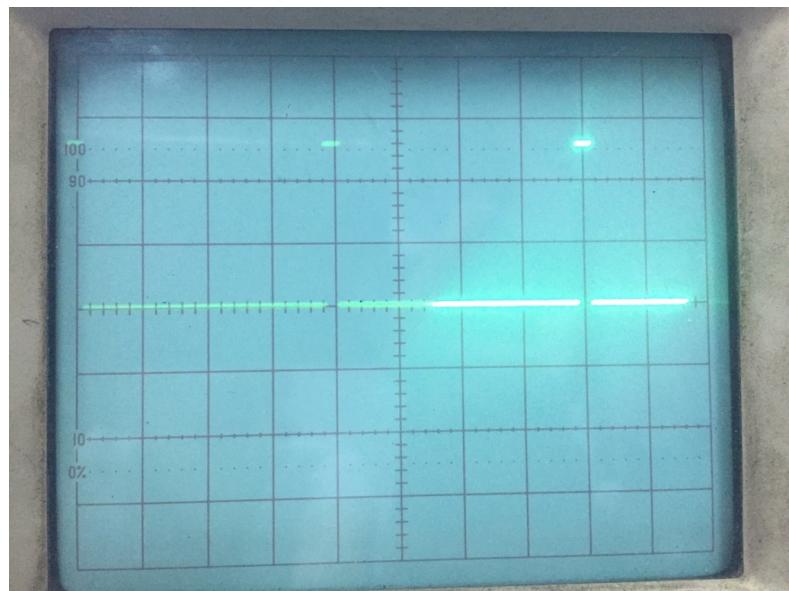


Figura 39: Visualización de la señal del motor en osciloscopio analógico.

Lo que pudimos observar en el osciloscopio es que el duty cycle de la señal varía con el tiempo, mientras que el período se mantiene constante. No sólo eso sino que pudimos observar que la variación del ancho de pulso respetaba un patrón según el cual crecía cuando el motor giraba en un sentido y decrecía cuando este cambiaba de sentido. Pudimos deducir entonces, que el cambio de sentido de giro del motor se relacionaba con la derivada de la función de crecimiento del ancho de pulso.

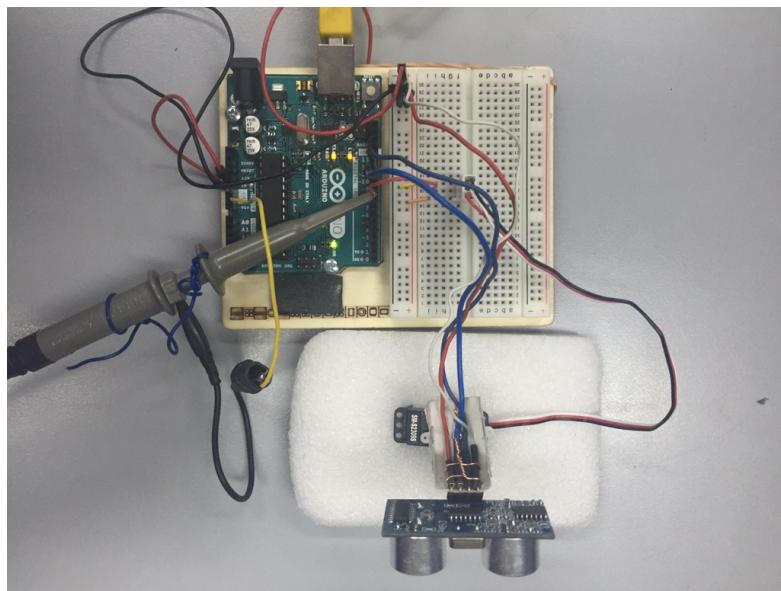


Figura 40: Banco de medición señal del servomotor con oscilloscopio.

9.3.6. Frecuencia y período de la señal de control del servomotor

Utilizando un contador universal **GOODWILL** modelo **GUC-2020** medimos tanto la frecuencia como el período de la señal del motor. A continuación presentamos los resultados obtenidos:

- Frecuencia: 0.05kHz
- Período: 002s

Estos resultados fueron obtenidos con la siguiente configuración del contador:

- Gate Time: 1s
- Modo: FREC. o PRD. según el caso.

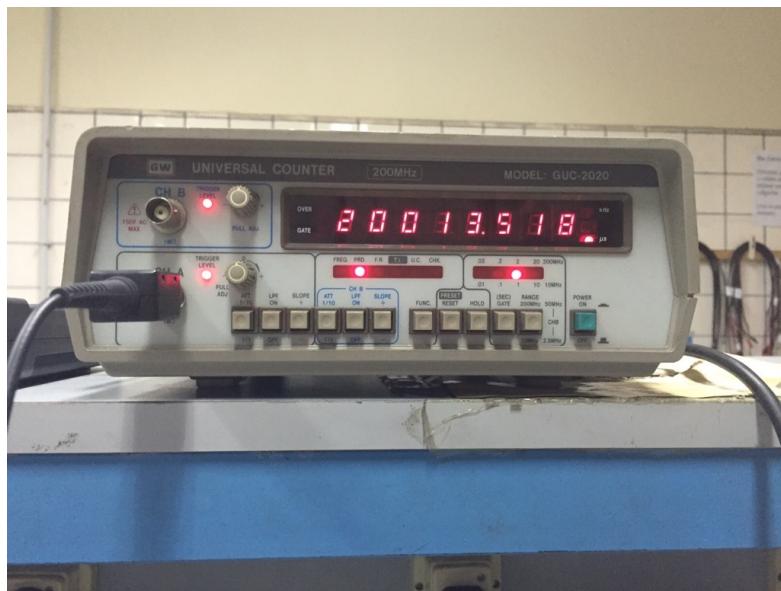


Figura 41: Medición período señal del servomotor con Contador Universal.

Para el contador universal:

- Error por variación de la tensión de línea = ± 1 ppm (considerando variación de tensión menor al 10 %)
- Error por estabilidad térmica = ± 5 ppm (en rango de trabajo 25-5 °C)
- Error por envejecimiento = ± 1 ppm/mes (± 60 ppm si suponemos 5 años desde última calibración)

El error de la base de tiempo queda establecido como:

$$\begin{aligned}\varepsilon_{BT} &= \varepsilon_{Temperatura} + \varepsilon_{Envejecimiento} + \varepsilon_{Tensiondelinea} \\ \varepsilon_{BT} &= 5 * 10^{-6} + 60 * 10^{-6} + 1 * 10^{-6} = \pm 6,6 * 10^{-5}\end{aligned}$$

Para este contador, al medir en modo frecuencia con un tiempo de compuerta de 1 s se midieron 0.05 kHz. Esto implica un total de 5 cuentas, con la incertezza determinada de la siguiente manera:

$$\varepsilon = \varepsilon_{BT} + \varepsilon_{\pm 1} = 61 \text{ ppm} + \frac{1}{50} \text{ ppm} = 0,1 \%$$

En modo período, el error para incluir una componente proveniente de la incertezza del disparador, que tiene que ver con el ruido de la señal en la entrada que se emplea para abrir la compuerta principal. Como la compuerta está en tiempo 1s, se promedian 50 períodos. La medición fue $20000 \mu\text{s}$, por lo que la incertezza será:

$$\varepsilon = \varepsilon_{BT} + \varepsilon_{\pm 1} + \varepsilon_{Disparador} = 61 \text{ ppm} + \frac{1}{20000} \text{ ppm} + \frac{3000}{100} \text{ ppm} = 0,01 \%$$

Obtenemos entonces los siguientes resultados con sus correspondientes incertezas:

- Frecuencia: $50\text{Hz} \pm 0.005\text{Hz}$
- Período: $20000 \mu\text{s} \pm 20\mu\text{s}$

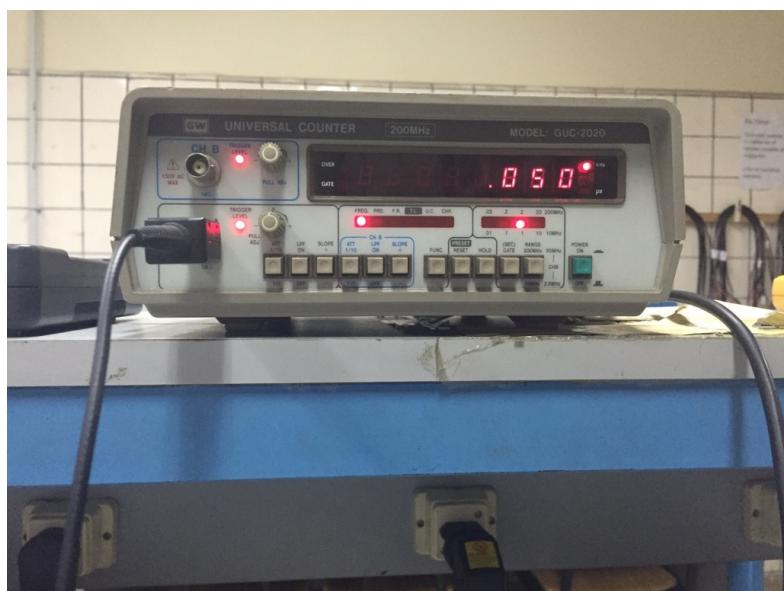


Figura 42: Medición frecuencia señal del servomotor con Contador Universal.

10. Lecciones aprendidas

Lo mejor que nos llevamos de este proyecto es haber aprendido a armar y programar un pequeño instrumento con Arduino. Cabe aclarar que programar

hardware fue un cambio significativo para nosotros que requirió tomar en cuenta cosas como darle tiempos de espera a los sensores para que ejecutaran las ordenes dadas a través del código, hecho que parece trivial pero nunca nos habíamos detenido a pensar. En segundo lugar, aprendimos cómo tomar la salida de datos de la placa por medio de su puerto serial y utilizarla en tiempo real para poder graficar resultados en una interfaz gráfica.

Aprendimos a utilizar Processing, una plataforma simple para confeccionar vistas programando en Java. Aquí nos encontramos con nuestro primer problema, los puertos no son un medio estable para transmitir y recibir información. Desde el comienzo, Processing presentó dificultades para leer el output que el sensor dejaba en el puerto de Arduino (A veces los datos venían por la mitad, la lectura se salteaba algunos ángulos de barrido, dejando a este incompleto, o incluso se producían lecturas con caracteres inválidos). Investigamos el problema en foros de ambas plataformas y encontramos solución. Se requirieron llamadas a la función `delay(int)`, debido a que cada vez que se iniciaba la interfaz de Processing, se reiniciaba el programa cargado en la placa Arduino, y cuando esto ocurre, la misma espera la carga de nuevo código. Como esto no ocurría, se producía el error reflejado en el puerto. Otra dificultad con la que nos encontramos siempre al trabajar con el instrumento es que Arduino IDE y Processing no se pueden mantener abiertos al mismo tiempo, el puerto dedicado a la placa puede ser utilizado por un entorno a la vez y figurará como ocupado para el otro.

Con respecto al hardware, nos encontramos con que el sensor y motor utilizados no venían con fichas de datos tan completas como nos hubiera gustado. El principal problema con el que nos encontramos en este aspecto fue buscar una referencia para el 0 de las distancias del sensor. Al no encontrarlo en la ficha del componente decidimos imponer uno y luego enfrentarnos con las consecuencias. Se tomó como 0 a la proyección del barrido del borde frontal del sensor en el banco de mediciones elaborado. Definida la referencia se realizó una calibración exhaustiva del instrumento como se relató en las secciones correspondientes. Otro detalle quizás no tan importante fue el soporte utilizado. Se pasó por varios diferentes, antes de encontrar una alternativa resistente a los transportes. Se probaron diferentes materiales y técnicas para fijar tanto el sensor como el cabezal del servomotor al mismo. Finalmente se encontró una combinación resistente, se utilizó cartón reforzado con dos capas, suprabond para darle su forma e hilos de cobre para agarrar los com-

ponentes a él. Con respecto a las conexiones, nos hemos equivocado el lugar de los cables en los pines, o incluso hemos enchufado el instrumento completo antes de darnos cuenta que había cables desconectados. Para todos estos casos observamos lo mismo, el conjunto formado por el motor y sensor no trabajan o lo hacen muy lentamente. El motor no se mueve y el sensor arroja mediciones nulas. Adjudicamos este problema a la ausencia alimentación de los componentes provocada por las conexiones erróneas.

Sobre las mediciones del instrumento, nos hubiera gustado visualizar la tensión en función de tiempo del pin 'trigger' del sensor ultrasónico, ya que este es quizás el más importante en el código que desarrollamos para el controlador. Probamos con diferentes configuraciones del osciloscopio y puntas pero no pudimos obtener una gráfica clara de la señal. En su reemplazo, medimos la tensión del pin de control del servomotor, cuyos resultados se encuentran asentados en la sección correspondiente.

Las dificultades encontradas fueron numerosas, sin embargo se pudo cumplir con la gran mayoría de los requisitos planteados. El proyecto arroja resultados satisfactorios, nos ha enseñado el uso de dos nuevas plataformas de diseño y desarrollo de hardware de código abierto y nos dio la oportunidad de poner en práctica todos nuestros conocimientos.

11. Conclusión

En primera instancia, y en referencia a los objetivos iniciales del trabajo, consideramos que logramos diseñar un proyecto que nos permitió aplicar los conceptos incorporados a lo largo de la materia. El motivo por el cual podemos realizar esta afirmación es que logramos diseñar un instrumento funcional sobre el cual pudimos realizar mediciones con todos los instrumentos estudiados, obteniendo resultados satisfactorios y acordes con lo esperado. El instrumento cumplió satisfactoriamente todos los requisitos impuestos por el enunciado. Para mejor visualización de los objetivos propuestos y alcanzados en la realización de este trabajo, se adjunta una tabla de objetivos (ver cuadro en sección 11.1) donde se presentan los objetivos iniciales del trabajo, marcando aquellos que fueron cumplidos una vez finalizado el mismo.

El proyecto en sí es muy sencillo y tiene mucho potencial. Agregando unas

pocas líneas de código se podría añadir el tiempo y exportar todas las mediciones realizadas por el sensor a una planilla de cálculo o incluso se podría integrar a un sistema más grande y complejo. El conjunto de sensor ultrasónico y servomotor estudiado podría cumplir la función de unos 'ojos' para un robot.

11.1. Verificación de requerimientos iniciales

Para finalizar, se realizó un cuadro para asentar los requerimiento cumplidos y no cumplidos.

Requerimiento	Cumplido	Comentarios
Dados por la cátedra	-	-
Realizable	✓	
Factible económicamente	✓	
Factible tecnológicamente	✓	
Factible temporalmente	✓	
Medible con instrumentos de laboratorio	✓	
Que posea una utilidad	✓	
Impuestos por nosotros	-	-
Obtener insumos	✓	
Aprender a manejar plataformas Arduino y Processing	✓	
Lograr consumo de corriente menor a 1 A para aparato completo	✓	
Lograr que detecte y mida correctamente la distancia a la que se encuentra un objeto en un rango de 5 a 100 cm	✗	Puede realizarse buscando las constantes de corrección para lo que resta del rango
Codificar programa para el microcontrolador	✓	
Lograr movimiento continuo del conjunto motor-sensor sin perder funcionalidad	✓	
Ensamblar aparato resistente a uso y transportes	✓	
Diseñar interfaz de fácil lectura	✓	
Optimizar interfaz de forma que refleje el barrido del sensor en tiempo real, sin desfasaje	✓	
Realización de mediciones con cada uno de los instrumentos aprendidos en la materia	✓	
Ánalysis de errores de mediciones del aparato y de instrumental de laboratorio sobre el mismo	✓	

Referencias

- [1] Arduino UNO, <https://www.arduino.cc/>
- [2] <https://www.arduino.cc/en/Main/ArduinoBoardUno>

[3] Radar robot con Arduino, http://10mp.net/blog/radar_robot_con_arduino.

12. Anexos

12.1. Anexo I: Fichas de datos de los componentes utilizados



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

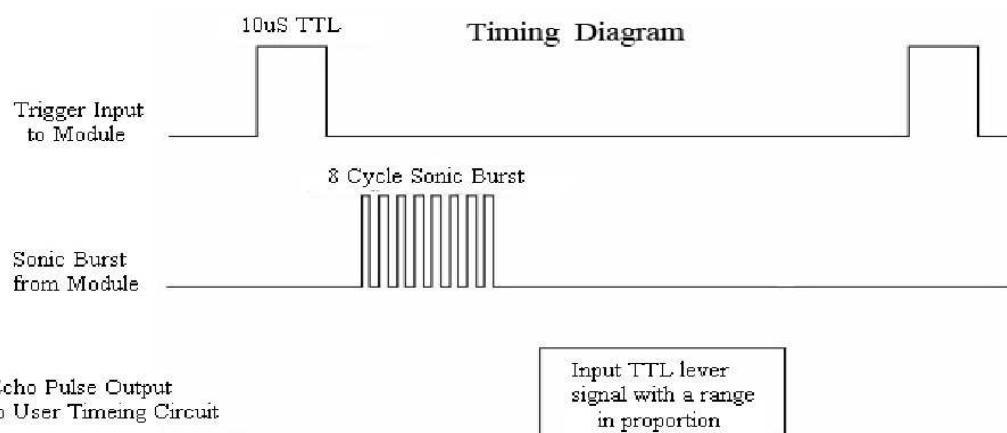
Electric Parameter

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



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: $uS / 58 = \text{centimeters}$ or $uS / 148 = \text{inch}$; 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.



Attention:

- The module is not suggested to connect directly to electric, if connected electric, the GND terminal should be connected the module first, otherwise, it will affect the normal work of the module.
- When tested objects, the range of area is not less than 0.5 square meters and the plane requests as smooth as possible, otherwise ,it will affect the results of measuring.

www.ElecFreaks.com



SM-S2309S MOTOR

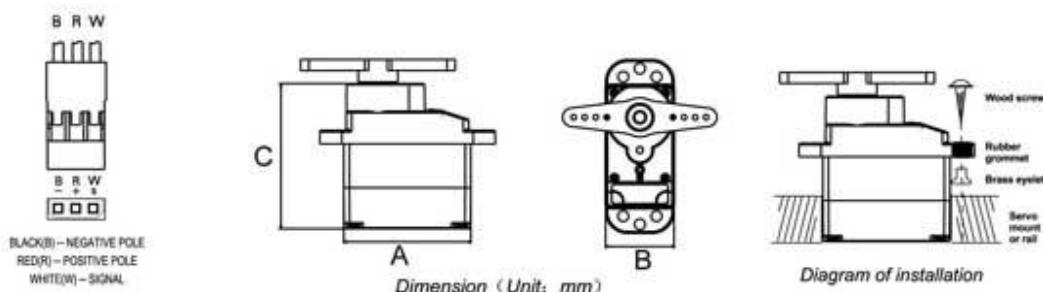


No. : SM-S2309S

Size: 22.9x12.3x22.2mm / 0.9x0.49x0.87in

Weight: 0.35oz

Specifications: Micro analog servo, 4 plastic gears + 1 metal gear



Products specification							Technical parameters							Rotation angle
Size(mm)					Weight		Wire	4.8V			6V			Rotation angle
A	B	C	D	E	g	oz		sec/60°	kg·cm	oz·in	sec/60°	kg·cm	oz·in	
22.9	12.3	22.2	-	-	9.9	0.35	20.0	0.11	1.1	15.3	0.09	1.3	18.1	±60°

(Specifications are subjected to change without notice.)

Product brochure

Micro analog servo, 4 plastic gear+ 1 metal gear.

Products packing

◦Packing with elevators (Elevators+PE bag)

Packaging content: Servo×1PCS、Servo arm×1bag、Manual×1PCS

Packaging specifications : Size-120×85mm、PE bag : 120×85×0.07mm、Net weight-9.9g、Gross weight-11.5g

◦General packing (PE bag)

Packaging content : Packing with PE bag

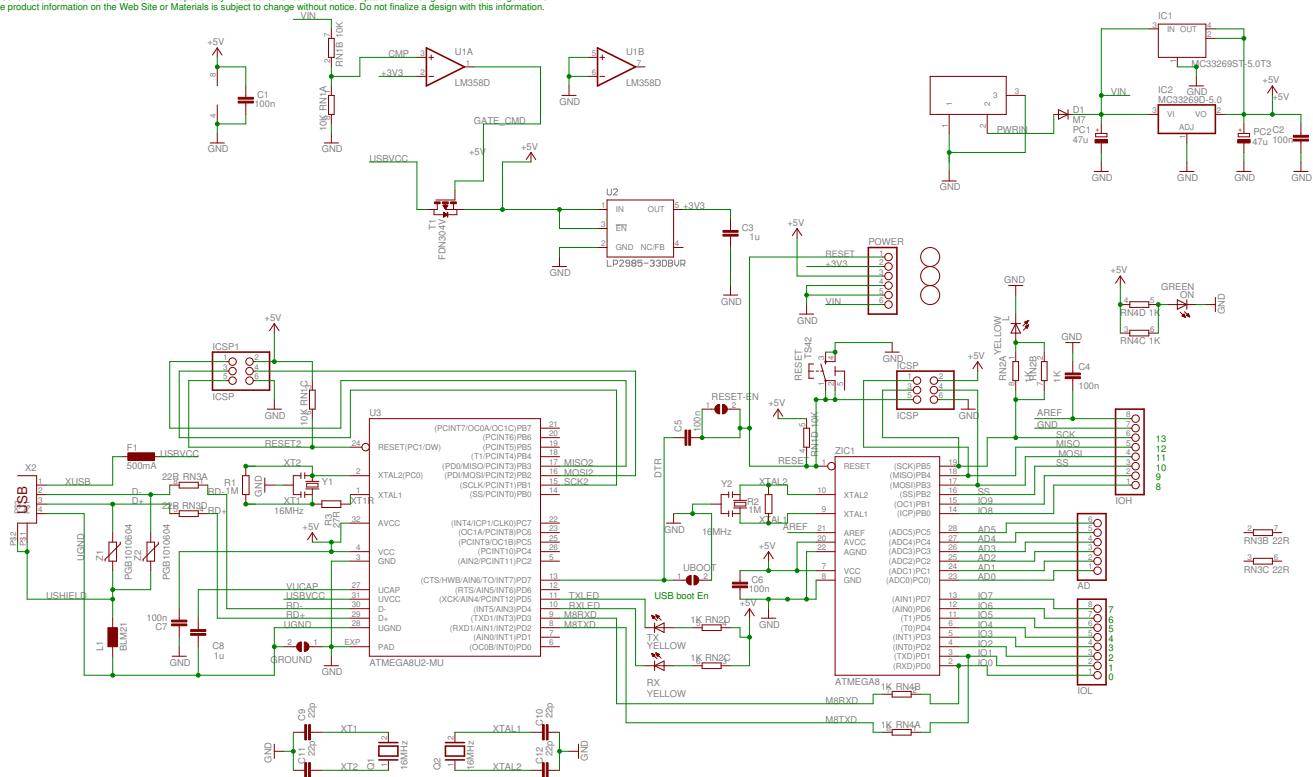
Packaging specifications : PE bag size-90×85×0.07mm、Net weight-9.9g、Gross weight-11.5g

Arduino™ UNO Reference Design

Reference Designs ARE PROVIDED "AS IS" AND "WITH ALL FAULTS." Arduino 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.

Arduino 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." Arduino reserves these for future definition and shall have no responsibility whatsoever for conflicts or inconsistencies 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.



12.2. Anexo II: Código del microcontrolador

```
1 #include <Servo.h>

3 // Pins
4 const int trigPin = 10;
5 const int echoPin = 11;
6 const int servoPin = 8;

7 // Servo's degree range
8 const int minDegree = 15;
9 const int maxDegree = 165;

11 long duration;
12 double distance;
13 Servo myServo;

15 void setup() {
16
17     pinMode(trigPin, OUTPUT); // Sets the trigPin as an output
18     pinMode(echoPin, INPUT); // Sets the echoPin as an input
19     myServo.attach(servoPin); // Defines on which pin is the servo
20         motor attached
21
22     Serial.begin(9600); // Initialize Serial communication
23 }
24
25 void loop() {
26     // Rotates the servo motor from 15 to 165 degrees
27     for(int degree = minDegree; degree <= maxDegree; degree++) {
28         moveServo(degree);
29     }
30
31     // Repeats the previous lines from 165 to 15 degrees
32     for(int degree = maxDegree; degree >= minDegree; degree--) {
33         moveServo(degree);
34     }
35 }

36 }
```

```

39 void moveServo(int degree) {
40
41     myServo.write(degree);
42     delay(30);
43
44     calculateDistance(); // Calls a function for calculating the
45     // distance measured by the Ultrasonic sensor for each degree
46
47 }
48
49 void calculateDistance() {
50
51     // The sensor is triggered by a HIGH pulse of 10 or more
52     // microseconds.
53     // Give a short LOW pulse beforehand to ensure a clean HIGH pulse:
54     digitalWrite(trigPin, LOW);
55     delayMicroseconds(2);
56
57     // Sets the trigPin on HIGH state for 10 micro seconds
58     digitalWrite(trigPin, HIGH);
59     delayMicroseconds(10);
60     digitalWrite(trigPin, LOW);
61
62     // Read the signal from the sensor: a HIGH pulse whose
63     // duration is the time (in microseconds) from the sending
64     // of the ping to the reception of its echo off of an object.
65     // Reads the echoPin, returns the sound wave travel time in
66     // microseconds
67     duration = pulseIn(echoPin, HIGH);
68
69     distance = microsecondsToCentimeters(duration);
70
71 }
72
73 double microsecondsToCentimeters(long microseconds) {
74
75     // The speed of sound is 340 m/s or 29 microseconds per centimeter.
76     // The ping travels out and back, so to find the distance of the
77     // object we take half of the distance travelled.

```

```

77 // Based on our testing measurements, a correction should be made
79 // by multiplying by the constant (3/4)

81     return (microseconds / 29.0 / 2.0) * (3.0 / 4.0);

83 }

85 void printData(int degree) {

87     Serial.print(degree);
88     Serial.print(",");
89     Serial.print(distance);
90     Serial.print("—");
91 }

```

Codigo/radar.ino.c

12.3. Anexo III: Código de la interfaz gráfica

```

public class Radar {

2     public void drawRadar( float distance , float pxDistance , int degree)
3     {
4         radar.drawCanvas();
5         radar.drawLine(degree);
6         radar.drawObject(distance , pxDistance , degree);
7         radar.drawText(distance , degree);
8     }

10    private void drawCanvas() {
11        pushMatrix();
12
13        translate(400,400); // line's center of rotation
14        noFill();
15        strokeWeight(1);
16
17        // draws the arc lines
18        for (int bigStroke = 0; bigStroke < 7; bigStroke++) {

```

```

    stroke(98,245,31);
20   strokeWeight(2);
    arc(0,0,100 * (bigStroke + 1),100 * (bigStroke + 1),PI,TWO_PI);
22   for (int littleStroke = 0; littleStroke < 5; littleStroke++) {
        stroke(105,105,105);
24     strokeWeight(0.01);
        arc(0,0,20 * littleStroke + 100 * bigStroke,20 * littleStroke
+ 100 * bigStroke,PI,TWO_PI);
26     }
}
28
// draws the degree lines
30   stroke(98,245,31);
31   strokeWeight(2);
32   line(-400,0,400,0);
33   line(0,0,-380*cos(radians(30)), -380*sin(radians(30)));
34   line(0,0,-380*cos(radians(60)), -380*sin(radians(60)));
35   line(0,0,-360*cos(radians(90)), -360*sin(radians(90)));
36   line(0,0,-380*cos(radians(120)), -380*sin(radians(120)));
37   line(0,0,-380*cos(radians(150)), -380*sin(radians(150)));
38   line(-400*cos(radians(30)),0,400,0);

40   popMatrix();
}
42
private void drawLine(int degree) {
43   pushMatrix();

44   translate(400,400); // line's center of rotation
45   strokeWeight(9);
46   // Green
47   stroke(30,250,60);

48   // draws the line according to the degree
49   line(0,0,380*cos(radians(degree)), -380*sin(radians(degree)));

50   popMatrix();
}
51
private void drawObject(float distance, float pxDistance, int
52   degree) {

```

```

58 pushMatrix() ;

60 translate(400,400); // line's center of rotation
61 strokeWeight(9);
62 // Red
63 stroke(255,10,10);

64 // draws the object according to the degree and the distance
65 if (distance < 35) line(pxDistance*cos(radians(degree)),-
66 pxDistance*sin(radians(degree)),380*cos(radians(degree)), -380*sin(
radians(degree)));

67 popMatrix();
68 }

70 private void drawText( float distance , int degree ) {
71 pushMatrix();

72 fill(0,0,0);
73 noStroke();
74 rect(0, 720, width, 576);

75 fill(98,245,31);
76 textSize(15);
77 text("5",440,420);
78 text("10",495,420);
79 text("15",545,420);
80 text("20",595,420);
81 text("25",645,420);
82 text("30",695,420);
83 text("35",745,420);

84 textSize(20);
85 String label;
86 if(distance > 35) label = "Out of Range";
87 else label = "In Range";
88 text("Object: " + label, 50, 450);
89 text("Degree: " + degree + " ", 450, 450);
90 text("Distance: " , 600, 450);
91 if(distance < 35) text(" " + distance +" cm" , 650, 450);
92 textSize(15);

```

```

    text("Acoustic Radar", 50, 480);

98
    textSize(15);
100
    fill(98,245,60);
102
    translate(400+400*cos(radians(30)),400-400*sin(radians(30)));
104
    rotate(-radians(-60));
106
    text("30",0,0);
108
    resetMatrix();
110
    translate(400+400*cos(radians(60)),400-400*sin(radians(60)));
112
    rotate(-radians(-30));
114
    text("60",0,0);
116
    resetMatrix();
118
    translate(400+350*cos(radians(90)),500-480*sin(radians(90)));
120
    rotate(radians(0));
122
    text("90",0,0);
124
    resetMatrix();
126
    translate(400+400*cos(radians(120)),400-400*sin(radians(120)));
128
    rotate(radians(-30));
130
    text("120",0,0);
132
    resetMatrix();
134
    translate(400+400*cos(radians(150)),400-400*sin(radians(150)));
136
    rotate(radians(-60));
138
    text("150",0,0);

140
    popMatrix();
142
}
144
}

```

Codigo/Radar.pde

```

import processing.serial.*;
2 import java.awt.event.KeyEvent;
3 import java.io.IOException;
4
5 Serial port;
6 Radar radar;
7
8 int degree;
9 float distance, pxDistance;
10

```

```

12  void setup() {
13    size(800, 500);
14    smooth();
15
16    port = new Serial(this, Serial.list()[0], 9600);
17    port.bufferUntil('-');
18
19    radar = new Radar();
20  }
21
22  void draw() {
23    fill(98,245,31);
24    noStroke();
25    fill(0,4);
26    rect(0, 0, width, 1010);
27    fill(98,245,31);
28
29    radar.drawRadar(distance, pxDistance, degree);
30  }
31
32  void serialEvent(Serial port) {
33    String data = port.readStringUntil('-');
34    try {
35
36      data = data.substring(0, data.length() - 1);
37      degree = int(data.substring(0, data.indexOf(",")));
38      distance = float(data.substring(data.indexOf(",") + 1, data.
39      length()));
40      pxDistance = convertDistanceToPixels(distance);
41
42      printData();
43
44    } catch (Exception e) {
45      System.out.print("Something failed");
46    }
47
48  void printData() {
49    System.out.print("Degree: ");
50    System.out.print(degree);

```

```

52     System.out.print(" - ");
53     System.out.print("Distance: ");
54     System.out.println(distance);
55 }
56 float convertDistanceToPixels(float distance) {
57     float conversionRate = 10; // 1cm = 10px
58     return distance * conversionRate;
59 }
```

Codigo/radar_view.pde

12.4. Anexo IV: Correcciones

1

- Se reemplazó el término 'coordenadas polares' por distancia y ángulo.

2

- Se corrige 'diseñado' por 'ideado' considerado que el diseño es pre-existente.
Se intenta aclarar también que el proyecto se construye con los conocimientos básicos de la materia para luego medir sobre él con los instrumentos más complejos que se aprenderán luego.
- Se aclara que la precisión del sensor de 3 mm mencionada proviene de la ficha de datos del componente. La misma no aclara cuál es su punto de referencia, se explica más adelante cómo definimos el cero nosotros mismos.
- Se omite una breve descripción de los posibles usos para los servomotores.

4

- Se explica que el sensor puede medir en rangos mucho más amplios que el trabajado de 10 a 35 cm, pero se verá más adelante (En la sección de calibración) cómo se realizaron mediciones de 2 a 60 cm y por qué se eligió este intervalo trabajado.

- Se expande la explicación sobre el mal funcionamiento del motor en los límites de 0 y 180 grados. El mismo fue puesto a prueba de diferentes formas, por separado. Al persistir el problema, se acotó el intervalo de grados para no tener que comprar otro motor.
- Se explica que la magnitud de 20 grados tomados como ángulo de detección del radar fue obtenida experimentalmente por una de nuestras fuentes (A pesar de que la ficha de datos del sensor diga que es 15 grados).
- Se explica el problema que acarrea este ángulo de detección del radar, provocando que el sensor detecte objetos frente a él 10 grados antes de donde están y termine de verlos 10 grados después.
- El ángulo no es medido por el sensor, sino que es una variable que se va incrementando y decrementando en uno en cada pasada para luego darle la orden al servomotor de apuntar al sensor allí (Véase el Anexo II: Código del microcontrolador).

5

- Se mejora la explicación.
- Para el desarrollo del código del microcontrolador se utiliza una única función de una librería; move(int) de Servo.

5.3.1

- Se omite una breve explicación sobre el uso del pin GND del sensor.
- Se corrige la mención de 'Trigger' como una salida, pues es una entrada.
- En este apartado, no se hacen comentarios sobre la variación de la velocidad del sonido con la humedad y temperatura. Estas cuestiones se tendrán en cuenta para el cálculo de la incertezza.

6.1.1

- Se mejora la explicación del código del microcontrolador. Es importante remarcar que La función pulseIn() inicia una cuenta en microsegundos que termina cuando se recibe un HIGH de 'Echo', es decir, cuando la onda emitida por el transmisor y rebotada por el obstáculo llega al receptor del sensor. (Véase el Anexo II: Código del microcontrolador con el código documentado)

6.1.2

- Se explica que el enlace al repositorio presentado fue creado por nosotros, sobre él se trabajó durante toda la cursada. Algunos comentarios fechados y publicados por nosotros se pueden ver en:

<https://github.com/abrdn/Acoustic-Radar/commits/master>

7.3

- La señal del pin de control del servomotor es enviada por el microcontrolador, se percibe periódica al repetirse siempre el mismo ciclo de ejecución.
- Con respecto a la medición del valor medio y eficaz de la tensión del pin de control del servomotor, calculamos el factor de cresta de esta señal para verificar si la medición con el voltímetro digital elegido es buena. De la visualización de la señal en el osciloscopio obtuvimos que $V_p = 2.3 \text{ V}$. Sabiendo que el V_{RMS} de una señal cuadrada es igual a su V_p podemos decir que:

$$F_{CRESTA} = \frac{V_{pico}}{V_{RMS}} = \frac{2,3V}{2,3V} = 1$$

Si comparamos con el factor de cresta del instrumento: <3:1 en la "parte baja" de la escala y <6:1 en la "parte alta" de la escala. Dado que las mediciones se mantuvieron en la parte alta de la escala, el fabricante garantiza un factor de cresta máximo de 6:1. Dado que el factor de cresta del instrumento resulta mayor que el de la señal podemos decir que la medición resulta confiable.

Cabe aclarar que es incorrecto decir que el servomotor se mantiene quieto por $30\mu\text{s}$, este es el tiempo que se le da para que llegue al ángulo de destino.

8

- Se mejora la descripción de los bancos de mediciones como es pedido.

8.1

- Se corrige 'salida USB' por 'entrada de alimentación USB'.
- Se explica cómo se encontró el pin de alimentación de la entrada USB, ya que no se encuentra señalizado ninguno de los cuatro en la placa.

8.2

- Se le proporcionó a la placa una tensión de 5.4 V aproximadamente al ser este el valor medido previamente sobre la entrada USB. Aquí se ha cometido un error, puesto que cuando se conecta la placa a una fuente de alimentación externa vía jack se le debe proporcionar una tensión de entre 7-12 V a diferencia de cuando de conecta a USB⁶. Tensiones menores a los 7 V pueden causar que los pines de 5 V de la placa fluctúen, causando inestabilidad y lecturas menos precisas.
- Se aclaró que los 20-40 mA por pin a los que se hizo referencia son la corriente de salida que puede entregar cada pin. Sin embargo, la corriente de entrada no puede ser explicada por la corriente de salida de los pines, ya que los pines brindan la corriente mencionada a partir de la carga que le proveen los capacitores de la placa. Sin embargo, y según el manual de la placa, el consumo máximo es de 1 A cuando la misma es alimentada por una fuente externa, con lo cual consideramos que los resultados fueron satisfactorios. Cabe aclarar que el valor teórico de corriente máxima 1.6 mA mencionado en las correcciones es el del sensor ultrasónico, y la medida hecha fue sobre el aparato completo.

⁶Se pueden ver estas especificaciones en: <https://www.arduino.cc/en/Main/ArduinoBoardUno>

8.2.1

- Se corrigió un error en la explicación de la medición. Anteriormente se mencionaba el regulador de tensión presente en la entrada USB, pero en este caso se utiliza la entrada (jack) provisto para la fuente extera.
- Se omitió especificar que el voltímetro utilizado es un TRUE RMS, debido a que se mide en CC y el funcionamiento en este modo es igual para todos los voltímetros del laboratorio.

8.3

- Se volvió a consultar la hoja de datos original para verificar los valores del informe. Se corrigió la unidad 'V' pasada al informe por la correcta; 'mV'.

8.5

- Se eliminó una comparación entre el valor de tensión del pin de control y el de alimentación del servomotor. Cabe aclarar que estos son diferentes conceptos, nosotros medimos sobre una entrada de control a un potenciómetro digital, no la entrada de alimentación Vcc.
- El ancho de pulso de la señal visualizada y su relación con la posición del servomotor serán mencionadas más adelante en la correspondiente sección de resultados.

8.6

- Se aclara que los resultados de esta medición confirman lo visualizado en el osciloscopio anteriormente.

9.1

- En esta sección se buscó comparar distancias conocidas con los valores medidos del aparato tal y como los devuelve el sensor. Los valores presentados en las tablas son los obtenidos en el puerto de Arduino, sin modificaciones. Las distancias teóricas fueron medidas con regla, la incerteza de los bancos de

mediciones utilizados se tendrá en consideración más adelante en la sección del cálculo de la incertezas.

- La razón por la que elegimos un rango acotado de trabajo fue que sólo en este funcionaba bien la constante de corrección $3/4$. Esto se debe a que la incidencia porcentual del error disminuía al medir en distancias más grandes, por lo tanto llegamos a la conclusión de que el error encontrado es de offset, provocado por la referencia al cero impuesta por nosotros mismos.

9.1.1

- Se modificó por completo la sección partiendo del cálculo utilizado en el código del microcontrolador para propagar incertezas. A su vez se consideraron los errores del banco de mediciones y de apreciación de la interfaz gráfica. Por último se muestra una cota para el error.

9.2

- La posición central es 90 grados, siendo la inicial 15 y la final 165 grados.
- Se aclara que la columna de 'Ángulo medido' se completa con el ángulo medio del intervalo pintado en rojo por la interfaz gráfica, donde se encuentra el obstáculo.
- Se cortan los decimales de las distancias medidas de acuerdo a la nueva cota del error calculada en la sección anterior.

9.3

- Se corrige 'consumo' en las mediciones de tensión por 'alimentación', debido a que el término anterior es erróneo.
- Se omite especificar el hecho de que los voltímetros sean TRUE RMS en los casos en los que no se mide en modo AC.

9.3.4

- Como se mencionó anteriormente, se cambió la unidad 'V' por 'mV' al ser esta la presente en la hoja de datos original.
- Se expande la explicación del funcionamiento de 'Echo' para aclarar por qué las mediciones son dispersas y los valores medidos dependen de los objetos frente al sensor. Este pin recibe un HIGH cuando la onda rebotada en el objeto llega al receptor del sensor ultrasónico. El tiempo que tarde en llegar este HIGH dependerá de qué tan próximo se encuentra el primer obstáculo frente al sensor en ese ángulo.

9.3.5

- Se omite una breve explicación que dice que si el ciclo de trabajo la señal visualizada fuera una función, su derivada sería una triangular.
- En la sección que enumera las mediciones que se harán sobre el radar con los instrumentos aprendidos en la materia se aclara que una de nuestras metas era visualizar las ondas de los pines 'Trigger' y 'Echo' pero nos encontramos con que las mismas son codificadas por la placa Arduino. Nos fue imposible verlas en el osciloscopio.

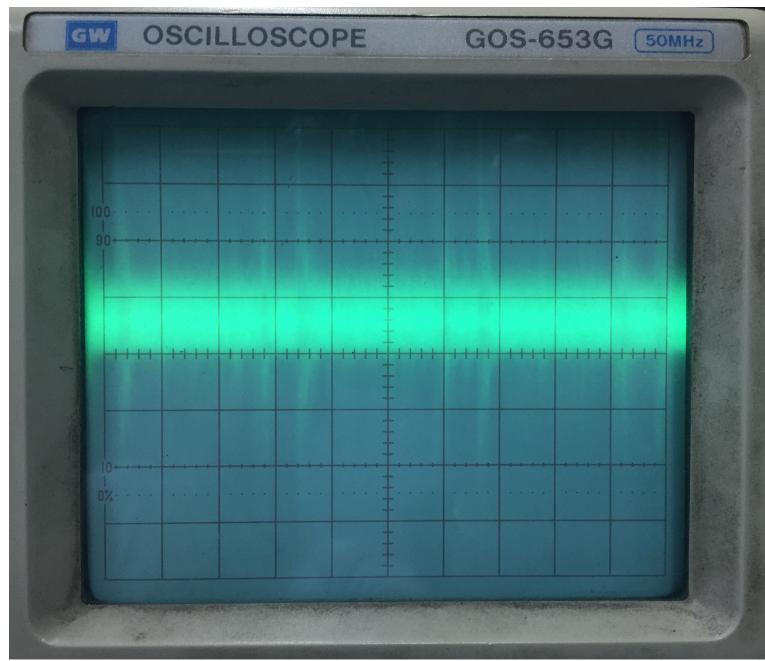


Figura 43: Intento de visualizar señal del pin 'Trigger' en el osciloscopio