INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE INGENIERÍA MECÁNICA Y ELÉCTRICA
UNIDAD ZACATENCO

“RADAR DETECTOR DE OBJETOS”

TESIS

PARA OBTENER EL TÍTULO DE:

INGENIERO EN COMUNICACIONES Y ELECTRÓNICA

PRESENTA:

SANTIAGO CRUZ JOSÉ ADRIÁN

ASESORES:

ING. MUEDANO MENESES JOSÉ JAVIER

ING. TRINIDAD ÁVILA LUCERO IVETTE

ENERO 2016 **CIUDAD DE MÉXICO**

INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE INGENIERÍA MECÁNICA Y ELÉCTRICA
UNIDAD PROFESIONAL "ADOLFO LÓPEZ MATEOS"

T E M A D E T E S I S

QUE PARA OBTENER EL TÍTULO DE	**INGENIERO EN COMUNICACIONES Y ELECTRÓNICA**
POR LA OPCIÓN DE TITULACIÓN	**TESIS Y EXAMEN ORAL INDIVIDUAL**
DEBERA (N) DESARROLLAR	**C. JOSE ADRIAN SANTIAGO CRUZ**

"RADAR DETECTOR DE OBJETOS"

DISEÑAR Y CONSTRUIR UN PROTOTIPO CAPAZ DE DETECTAR OBJETOS, CON AYUDA DEL SENSOR ULTRASÓNICO HC-SRF04

- **ANTECEDENTES**
- **INTRODUCCIÓN AL SISTEMA DEL RADAR**
- **DETERMINACIÓN DE LA DISTANCIA, VELOCIDAD Y POSICIÓN**
- **INTERACCIÓN Y RECEPCIÓN DE LA SEÑAL**
- **DESARROLLO DEL PROTOTIPO**

CIUDAD DE MÉXICO, A 29 DE NOVIEMBRE DE 2016.

A S E S O R E S

ING. LUCERO IVETTE TRINIDAD ÁVILA

ING. JOSÉ JAVIER MUEDANO MENESES

ING. PATRICIA LORENA RAMÍREZ RANGEL
JEFA DEL DEPARTAMENTO ACADÉMICO DE INGENIERÍA ELÉCTRICA

AGRADECIMIENTO

Agradezco a mis padres por todo el apoyo que me han brindado a lo largo de mi vida académica, a pesar de que ellos no tuvieron la oportunidad de estudiar ellos me dieron la oportunidad de crecer, superarme y ser alguien en la vida, es por eso que este logro es mas de ellos, tantos años de lucha y sacrificio, es algo que no cualquiera ofrece, por eso y mucho más los quiero.

A mi madre porque es la mujer más fuerte del mundo, siempre está al cuidado de todos nosotros, sus hijos. Y soportando no solo sus propios problemas, sino también los míos porque a pesar de todo, siempre ha estado ahí cuando más la necesito y nunca me ha apartado ni un solo instante de ella.

A mi padre porque él es un ejemplo de fuerza y perseverancia, nunca se deja vencer por nada y continúa con todo el trabajo que tiene por delante, demostrándome que todas las cosas se pueden lograr trabajando.

A mis hermanos porque en parte soy el reflejo de lo que ellos fueron y son, luchando contra muchas adversidades para poder salir adelante, es algo que me motivó de todos ellos para poder lograr esto.

A mi familia en general quiero agradecerles por todo el amor brindado, porque sin lugar a dudas es algo que todos necesitamos para poder ser felices, y a mí me ha tocado una familia increíble, porque a pesar de que haya momentos difíciles siempre nos apoyamos para no dejarnos vencer tan fácilmente.

De ahora en adelante quiero demostrarles que al igual que ellos también puedo ser alguien en quien puedan confiar, demostrándoles que puedo lograr muchas cosas más, sin apartarme de ustedes porque son mi más grande tesoro, los amo.

Agradezco al Instituto Politécnico Nacional, la institución cual me brindó la oportunidad de sobresalir y por impartir sus conocimientos.

Santiago Cruz José Adrián

INDICE

OBJETIVO

Diseñar y construir un prototipo capaz de detectar objetos, con ayuda del sensor ultrasónico HC-SRF04

JUSTIFICACIÓN

El sistema de radar es un sistema complejo que utiliza muchos elementos tales como el procesamiento de señales, procesamiento de datos, dispersión electromagnética, detección, estimación de parámetros, extracción de información, antenas, transmisores y receptores. Lo anterior hace que sea un sistema muy interesante de estudiar y con ello lograr el desarrollo de nuevos campos de aplicación que contribuyan en el mejoramiento de nuestra vida cotidiana.

El desarrollo de un radar hoy en día puede llegar a facilitar la obtención de muchas cosas como podrían ser imágenes de superficies planetarias o cartografía de zonas de alta nubosidad (inaccesibles mediante sensores ópticos), es por eso que el desarrollo de un radar puede ser una tecnología que pueda ayudar a resolver estos problemas.

El desarrollo de este radar se realizó a través de Arduino, un sensor HY-SRF04 y un servomotor y para visualizar los datos obtenidos a través del sensor se visualizan en el pc mediante la comunicación serial

ANTECEDENTES

El principio del radar fue expuesto por el escritor norteamericano H. Gernsback en su novela de ciencia ficción a fines de 1911. En 1922, Marconi expuso el principio de reflexión de las ondas electromagnéticas al incidir sobre un obstáculo. Appleton en el año de 1924, estudió la reflexión de impulsos electromagnéticos en la ionósfera y un tiempo después, Pierre David localizó un avión en vuelo mediante este procedimiento. La invención de los amplificadores de potencia, primero el magnetrón y más adelante el Klystron, abrieron auténticas posibilidades en el desarrollo del sistema de radar. En los años de 1934 y 1935, las investigaciones llegaron a resultados prometedores, gracias al alemán R. Kuhnhold al británico Watson Watt, quienes diseñaban dispositivos para la localización de aeronaves en vuelo. Sin embargo, no fue hasta después de la segunda guerra mundial que el uso del radar se generalizó en el campo naval, aéreo y terrestre, tanto militar como civil. El termino radar es una contracción de las palabras en inglés Radio Detection And Ranging (radio detección y medición de la distancia) [1]. El nombre refleja la importancia dada por los primeros investigadores en este campo sobre la necesidad de un dispositivo para detectar la presencia de un blanco y la medición de su distancia. No existe alguna técnica competitiva que pueda medir con exactitud grandes distancias tanto en clima, bueno o adverso, tan bien como el radar. En la actualidad el radar es un sistema que sirve para detectar, seguir y obtener la distancia, posición, velocidad, y forma de objetos tales como aviones, barcos, naves espaciales, vehículos, personas, etc.

CAPÍTULO I INTRODUCCIÓN AL SISTEMA DE RADAR

1.1 Operación del radar

El radar es un sistema electromagnético que sirve para detectar y localizar blancos tales como aviones, barcos, naves espaciales, vehículos, gente y medio ambiente. Su operación consiste en radiar ondas electromagnéticas al espacio y detectar el eco de la señal reflejada del blanco. La energía reflejada que regresa al radar no solo indica la presencia de un objeto, puesto que comparando la señal recibida con la señal que fue transmitida, se puede determinar su posición. El radar puede realizar su función a grandes o cortas distancias y puede operar en la oscuridad, neblina, niebla, lluvia y nieve. Su habilidad de medir distancia y velocidad con una alta exactitud y además de proporcionar la imagen del blanco, son unos de sus principales atributos. El principio básico de radar es ilustrado en la figura 1. Un transmisor genera una señal electromagnética (como por ejemplo un pulso corto de onda senoidal) que es radiado al espacio por medio de una antena. Una porción de la energía transmitida es interceptada por el blanco y reflejada en muchas direcciones. La energía reflejada del blanco que regresa hacia el radar es capturada por la antena del radar. Enseguida es entregada al receptor, donde es procesada para detectar la presencia de un blanco y determinar su posición [1]. Por lo general, una sola antena es utilizada en sistemas que utilizan como señal una serie de pulsos repetitivos, compartiendo el tiempo del periodo de repetición de pulsos tanto para transmisión como para recepción. La distancia del blanco se encuentra midiendo el tiempo que se toma en viajar la señal transmitida hacia el blanco hasta que regrese hacia el radar.

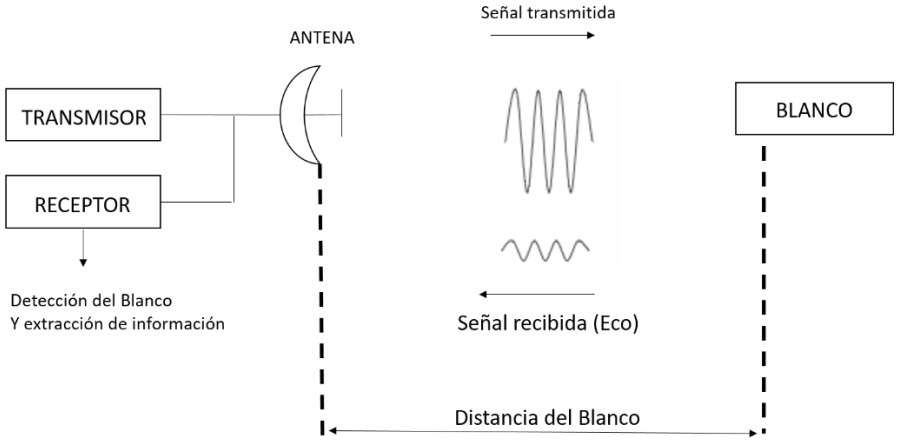


Figura 1 Principio básico del radar

El ángulo de la posición del blanco puede ser encontrado mediante el ancho de lóbulo principal de la antena de radar, cuando la señal recibida tiene máxima amplitud. Si el blanco está en movimiento, ocurre un cambio de frecuencia en la señal debido al efecto Doppler. Este cambio de frecuencia es proporcional a la velocidad del blanco en relación con el radar (también llamado velocidad radial).

El cambio de frecuencia Doppler es muy utilizado en radares para diferenciar blancos en movimiento que se desean localizar, de los objetos estáticos no

deseados denominados "clutters", los cuales son reflejados del ambiente natural tales como la tierra, el mar o la lluvia.

1.2 Clasificaciones del radar

Radar de onda continua (CW): transmite ininterrumpidamente. El radar de la policía suele ser de onda continua y detecta velocidades gracias al efecto Doppler.

Radar de onda continua con modulación (CW-FM, CW-PM): se le añade a la señal modulación de fase o frecuencia con objeto de determinar cuándo se transmitió la señal correspondiente a un eco (permite estimar distancias).

Radar de onda pulsada: es el funcionamiento habitual. Se transmite periódicamente un pulso, que puede estar modulado o no. Si aparecen ecos de pulsos anteriores al último transmitido, se interpretarán como pertenecientes a este último, de modo que aparecerán trazas de blancos inexistentes.
Según su finalidad

Radar de seguimiento: es capaz de seguir el movimiento de un blanco. Por ejemplo, el radar de guía de misiles.

Radar de búsqueda: explora todo el espacio, o un sector de él, mostrando todos los blancos que aparecen. Existen radares con capacidad de funcionar en ambos modos.

1. 3 Aplicaciones de radar

Existe una variedad extensa de aplicaciones del radar, las cuales día con día van incrementándose. En general, los radares son utilizados en aplicaciones militares, de navegación, seguridad marítima, meteorológicas y detección de la velocidad de vehículos. Los campos de aplicación más importantes son los que se indican a continuación.

Vigilancia aérea

Alarma temprana a larga distancia; intercepción controlada desde tierra; adquisición para sistemas de alarmas; determinación de altitud en radares tridimensionales; vigilancia de aeropuertos y rutas aéreas.

Vigilancia espacial y proyectiles dirigidos

Alarma de proyectiles dirigidos; adquisición de proyectiles dirigidos; vigilancia de satélites artificiales.

Búsqueda de superficie y vigilancia de operaciones militares

Búsqueda marina y navegación; topografía; localización de artillería y de morteros; control de movimiento en las pistas de los aeropuertos

Radar meteorológico

Observación y predicción de fenómenos naturales.

Seguimiento y guía

Control de tiro antiaéreo; control de tiro de superficie; guía de proyectiles; instrumentación a distancia; instrumentación de satélites; aproximación y aterrizaje de precisión.

Pueden mencionarse un gran número de aplicaciones no pertenecientes a categorías definidas: alarmas contra intrusos, monitorización de migraciones de aves, control de vehículos terrestres, etc. En todos estos sistemas se aplican los principios básicos de radar, con la apropiada definición de los parámetros de los blancos y de los requerimientos de resolución medida.

1.4 Frecuencias de radar

Los sistemas de radar trabajan en diferentes porciones de las bandas de frecuencia UHF, L, S, C, X, KU, K, Ka y hasta milimétricas como se muestra en la tabla 1. En principio estas bandas fueron designadas de esta forma en la segunda guerra mundial como secreto militar, donde el código original de letras era P, L, S, X, y K posteriormente se le añadieron mas bandas. Una vez que ya no fue secreto militar se divulgaron y en la actualidad permanecen estas designaciones. En general los radares trabajan en longitudes de onda entre 100 cm o mayores, hasta 10 –7mts o menores, lo cual en frecuencia sería aproximadamente de menos de 1 GHz, hasta mayores de los 100 GHz [1]. Sin embargo, no existen límites fundamentales para las frecuencias de radar. Cualquier dispositivo que detecte y localice blancos por medio de la radiación electromagnética y utilice el eco devuelto por el blanco puede ser clasificado como un radar, cualquiera que sea la frecuencia utilizada.

Tabla 1 Bandas de frecuencia del radar

DESIGNACION DE BANDA	RANGO DE FRECUENCIA (GHz)	LONGITUD DE ONDA (cm)
UHF	0.3 - 1.0	30 - 100
L	1.0 2.0	15 – 30
S	2.0 - 4.0	7.5 – 15
C	4.0 - 8.0	3.75 – 7.5
X	8.0 - 12.0	2.5 – 3.75
Ku	12.0 - 18.0	1.67 – 2.5
K	18.0 - 27.0	1.11 – 1.67
KR	27.0 - 40.0	0.75 – 1.11
Milimétrica	40.0 - 300.0	0.1 – 0.75

1.5 Transductores Piezoeléctricos

Los transductores son elementos que transforman una magnitud física en una señal eléctrica.

La palabra "piezo" viene del griego y significa "yo presiono". Un sólido piezoeléctrico sobre el que se ejerce una presión mecánica reacciona con el desplazamiento de cargas eléctricas. La causa del fenómeno es que las tensiones de presión y tracción ejercidas sobre un material piezoeléctrico (cristal o cerámica piezoeléctricos) provocan deformaciones en la retícula cristalina. Los desplazamientos producen la aparición o desaparición de cargas moleculares que pueden manifestarse como tensiones eléctricas.

A esto se le llama efecto piezoeléctrico directo. A la inversa, las cargas provocan en los electrodos una distorsión de la retícula y en consecuencia una deformación del material piezoeléctrico. Esto se le denomina efecto piezoeléctrico inverso o reciproco. Ambos efectos tienen un aprovechamiento técnico.

1.5.1 Efecto Piezoeléctrico

Para comprender el efecto piezoeléctrico consideramos un cristal de cuarzo situado entre dos electrodos metálicos, los cuales quedan aplicados a la entrada de un osciloscopio. Si se aplica una fuerza de comprensión al cristal, aparece un impulso de tensión, justo al ejercer la fuerza sobre el cristal, lo cual demuestra que la presión mecánica ejercida sobre el cristal queda transformada en una tensión eléctrica.

Para la explicación del porqué de este fenómeno físico, se observa en la siguiente figura un corte en el modele cristalino de un cristal de cuarzo.

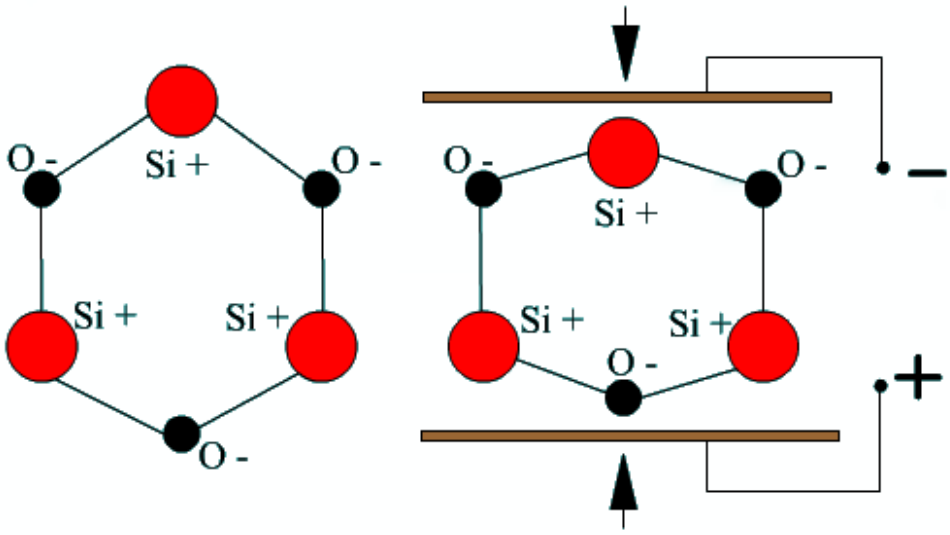


Figura 1.1 Efecto piezoeléctrico

El cuarzo está formado por iones positivos de silicio y por iones negativos de oxígeno. Estos iones están repartidos uniformemente de forma que neutralizan sus efectos hacia el exterior. Si este cristal es comprimido, por ejemplo, entre dos placas metálicas, tal como muestra la figura, en la placa superior surge un exceso de cargas negativas y en la inferior un exceso de cargas positivas, dado que ambas placas están separadas por el grueso del cristal, entre ellas aparecerá, pues una tensión eléctrica.

Si en lugar de comprimirse se tensa el cristal, las cargas que aparecerán en las placas serán naturalmente de polaridad opuesta. Siguiendo el ritmo de las oscilaciones de presión aparece entre los terminales de las placas una tensión alterna.

El fenómeno descrito es reversible, es decir si entre las caras de un cristal de cuarzo aplicamos una tensión eléctrica, el cristal se deforma.
Aquí puede observarse una simulación al comprimir y tensar un cristal como se genera la tensión.

CAPITULO II: LA DISTANCIA, VELOCIDAD Y POSICIÓN DEL OBJETO.

Los sistemas de radar emiten ondas electromagnéticas dirigidas al espacio y capturan el eco de la señal reflejada del blanco. Mediante el procesamiento de esta señal, es posible determinar la distancia, posición y velocidad de un blanco. El escoger un tipo de señal y una técnica de procesamiento de señal en un sistema de radar, depende fuertemente de la misión y operación específica del radar. La distancia, por ejemplo, es calculada mediante el tiempo de retraso de la señal recibida, el cual es obtenido mediante la correlación de la señal transmitida y el eco reflejado. Por otra parte, la velocidad del blanco es determinada mediante la frecuencia Doppler, la cual es un cambio de frecuencia de la señal reflejada debido al movimiento del blanco. La frecuencia Doppler es determinada analizando el efecto Doppler. En el caso de los ángulos de azimut y de elevación, es decir, la posición angular del blanco, es obtenida a partir del ancho de media potencia del lóbulo principal de una antena parabólica.

2.1 La distancia entre el radar y el objeto.

La distancia de un blanco es determinada por el tiempo que le toma a la señal electromagnética transmitida por el radar viaja de ida y de vuelta en la atmosfera. A este tiempo se le denomina como tiempo de retardo t_r [2]. La energía electromagnética en espacio libre viaja con la velocidad de la luz, la cual es $c = 3x10^8 m/s$. Por lo tanto, el tiempo para que la señal viaje hacia la posición del blanco a una distancia R y regrese al radar es 2R/c. La distancia del blanco es entonces

$$R = \frac{ct_r}{2} \quad (\mathbf{2.1})$$

Para obtener el tiempo de retardo de la señal se recurre a la función correlación, la cual es máxima cuando la señal transmitida y la señal recibida son similares. Al correlacionar la señal transmitida y la señal reflejada se obtiene en qué tiempo se presenta la máxima similitud entre ellas.
Este tiempo es el retardo de la señal recibida con respecto a la señal transmitida.

2.1.1 Ambigüedad en la distancia

Un radar de pulsos recibe y transmite un tren de pulsos como se puede ver en la figura 2.1.1. El período de repetición del pulso es T y el ancho de pulso es τ. Las ondas electromagnéticas empleadas por los radares de pulsos son radiadas solo durante τ segundos durante cada período T. El tiempo restante es utilizado para recibir los ecos reflejados. Cuando una señal reflejada tiene un tiempo de retraso T, significa que un blanco se encuentra a la distancia máxima que no presenta ambigüedades [2]. La ambigüedad en la distancia ocurre desde el segundo pulso enviado por el radar, como se muestra en la figura 2.1

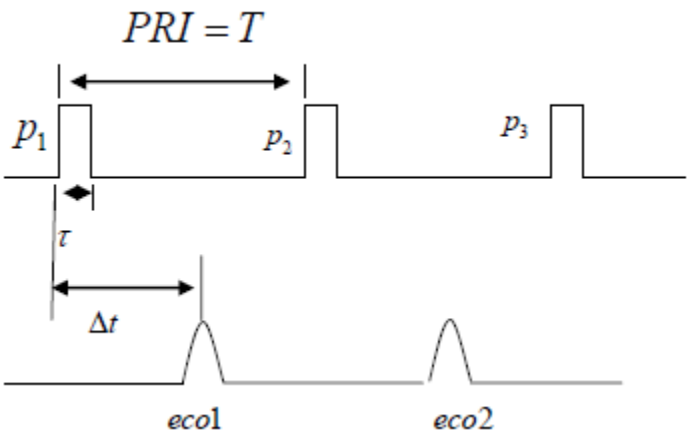


Figura 2.1 Tren de pulsos transmitidos y recibidos

El eco 1 representa la señal reflejada de un blanco que se encuentra a una distancia $R_1 = c\Delta t/2$ debido al pulso 1. Sin embargo, el eco 2 se puede interpretar como la señal reflejada del mismo blanco debido al pulso 2, o podría ser la señal reflejada de un blanco que se encuentre a una distancia lejana R_1 debido al pulso 1. Por lo tanto, la ambigüedad en la distancia se presenta en el eco 2. Para resolver este problema, una vez que el pulso ha sido transmitido el radar debe esperar el tiempo suficiente hasta que los ecos de los blancos que se encuentran en la distancia máxima de detección regresen al radar antes de emitir el siguiente pulso. Por lo tanto, la distancia máxima sin ambigüedades será determinada por

$$R_{max} = c\frac{T}{2} \quad (\mathbf{2.2})$$

2.1.2 La resolución en distancia (∆R)

Describe la capacidad que tiene el radar de detectar objetos que se encuentran cercanos uno del otro como objetos distintos [2]. Es decir, que todos los blancos que se encuentren separados al menos por una distancia ∆R podrán identificarse sin problemas.

Para encontrar la resolución de un radar es necesario relacionar la distancia de separación entre los blancos y el ancho del pulso τ que emite el radar. Por lo tanto, primero asumimos que tenemos dos blancos separados a una distancia ∆R = cτ/4 como se muestra en la figura. En este caso, el pulso reflejado del blanco 1 $(ecoB_1)$ y el pulso reflejado del blanco 2 $(ecoB_2)$ se traslapan al transcurrir τ / 2 segundos después de que el borde inicial del pulso incidente golpea el blanco 1. Por lo tanto, los blancos no pueden ser identificados como dos blancos distintos.

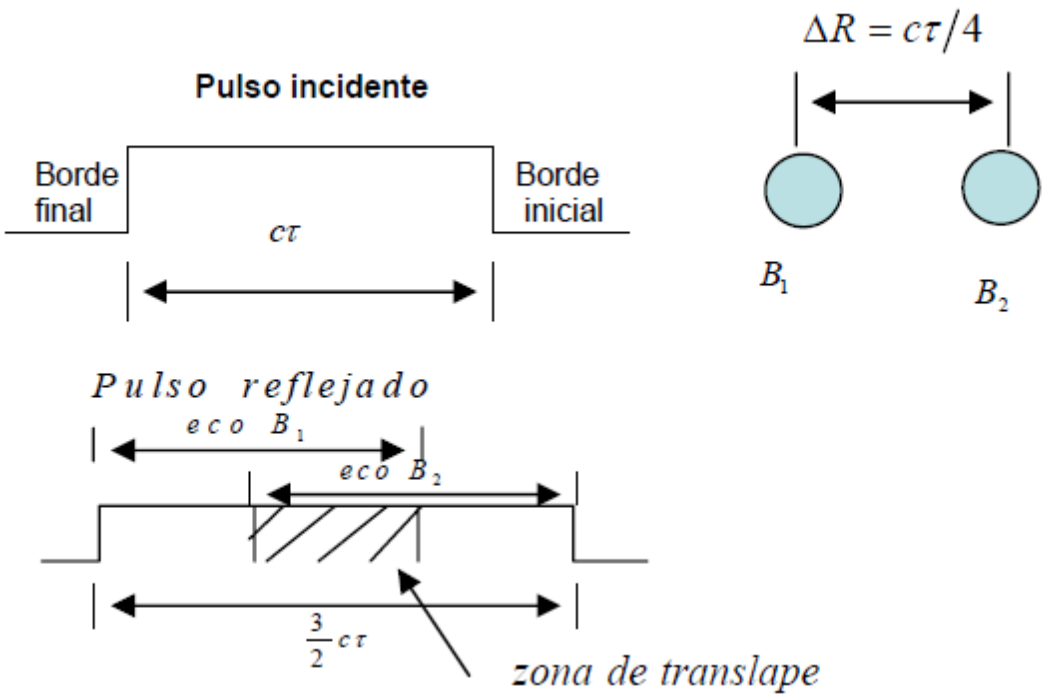


Figura (2.2 A) Blancos que no pueden ser identificados [2]

Ahora bien, se supone que los dos blancos están separados por una distancia de al menos cτ/2. En este caso, los pulsos reflejados de los blancos no se traslapan. Esto debido a que la distancia de separación entre los blancos de ida y de vuelta que recorre el pulso incidente es igual al ancho de pulso en distancia. Por lo tanto, en este caso los blancos son identificables como blancos distintos

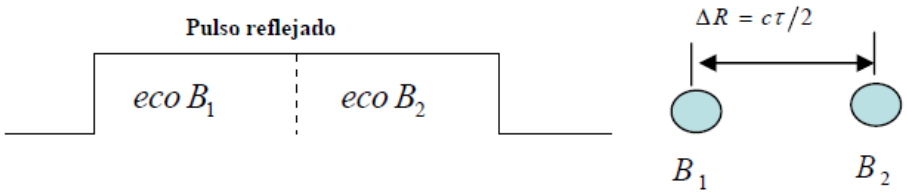


Figura (2.2 B) Blancos perfectamente identificables [2]

Para alcanzar una resolución de distancia precisa hay que reducir al mínimo el ancho del pulso. Sin embargo, esto reducirá el promedio de la potencia transmitida y aumentará el ancho de banda de operación. Por lo tanto, si se desea tener una determinada resolución en un radar de pulsos, el ancho del pulso estará dada por

$$\tau = \frac{2\Delta R}{c} \quad (\mathbf{2.3})$$

2.2 La velocidad del objeto

Para determinar la velocidad radial de un blanco, el radar utiliza el efecto Doppler. Este efecto describe el cambio en la frecuencia central de una señal incidente, debido al movimiento del blanco con respecto a la fuente de radiación. Dependiendo de la dirección del blanco en movimiento, este cambio de frecuencia es positivo (cuando el blanco se acerca hacia el radar) o negativo (cuando el blanco se aleja). Una señal incidente sobre un blanco tiene frentes de onda de igual fase separados por λ, como se aprecia en la figura (2.3 A). Si el blanco se mueve hacia el radar con una velocidad v, causará que los frentes de onda reflejados tengan menos separación entre ellos, originando que la longitud de onda sea menor. Por otra parte, si el blanco se está alejando de la fuente de radiación, provocará que la separación de los frentes de onda reflejados se expanda, es decir, la longitud de onda estará siendo mayor. Cabe señalar que el aumento o disminución en la longitud de onda de los frentes de onda representa un cambio en la frecuencia de la señal reflejada.

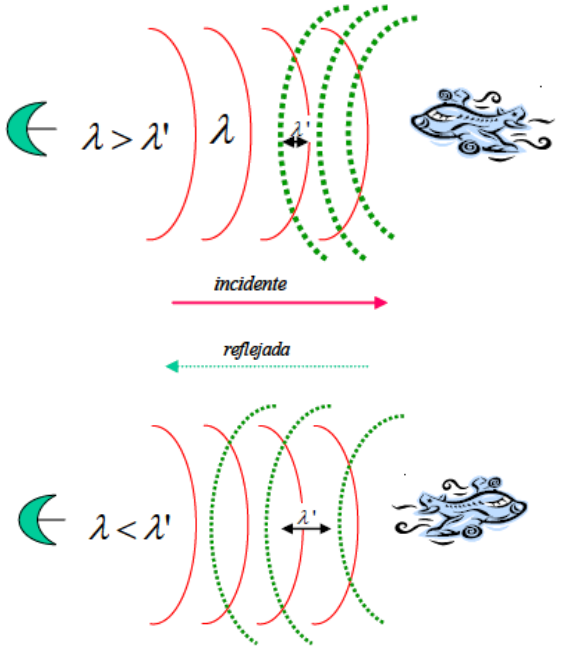


Figura (2.3 A) Efectos del movimiento del blanco sobre los frentes de onda [2]

Por lo tanto, cuando los pulsos transmitidos por el radar inciden sobre el blanco, éste ocasionará que el ancho del pulso cambie, ya sea incrementando o disminuyendo. Por lo tanto, enseguida se obtendrá una expresión que representa el cambio del ancho de pulso reflejado con respecto al original.

Si se tiene un pulso de ancho τ, que choca contra un blanco que viaja a una velocidad (*v)* hacia el radar, como se puede ver en la figura **(**2.3 B). Y se dice que *d* es la distancia en metros que se mueve el blanco durante un intervalo Δt, donde Δt es igual al lapso de tiempo entre el frente principal del pulso que choca contra el blanco y el frente final del pulso que choca al blanco.

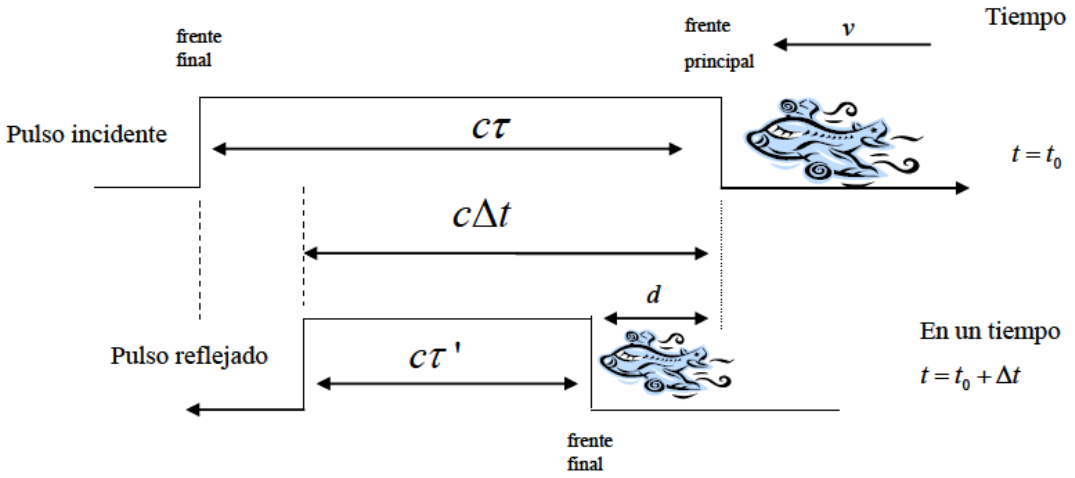


Figura (2.3 B) Impacto de la velocidad del blanco en un solo pulso [2]

2.3 Determinación de la posición

El radar mide la posición angular relativa de un blanco por medio del ángulo de azimut y el ángulo de elevación. La precisión para determinar estas mediciones se basa en la utilización de patrones de radiación de antenas muy directivas. Es decir, que el ángulo de apertura del lóbulo principal a los 3dB de la antena debe ser muy estrecho para obtener un error menor en las mediciones. Esto debido a que cuando un radar detecta un blanco, se sabe que se encuentra dentro del patrón de radiación de la antena [2]. Sin embargo, la posición exacta dentro del lóbulo es desconocida. Es por ello que se puede determinar la posición relativa del blanco mediante la posición de la línea de vista de la antena, asumiendo un error debido al ancho del lóbulo. En la figura 2.4A se muestra la posición del lóbulo y del blanco.

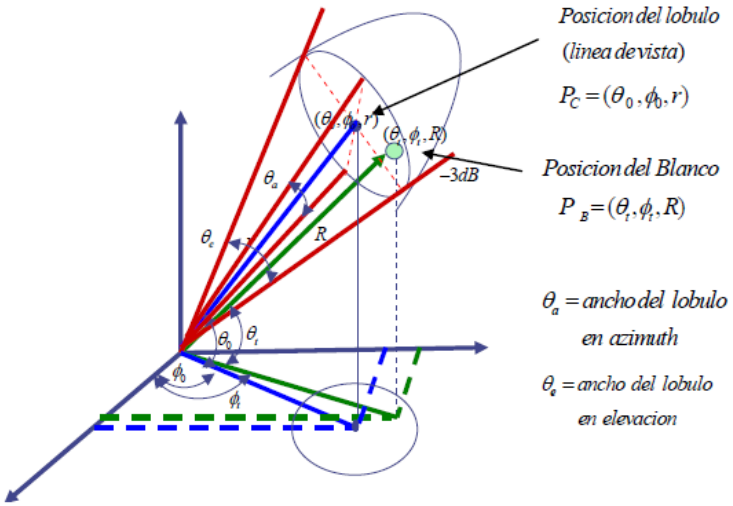


Figura (2.4A) Esquema del diagrama de radiación

Para encontrar una expresión del error de los ángulos de azimut ϕ_T y elevación θ_T asumimos dos barridos del lóbulo, uno barrido horizontal (azimut) y un barrido vertical (elevación).

Barrido Horizontal (azimut) [2]***:*** La técnica se basa en mover el lóbulo de radiación horizontalmente hasta que el eco reflejado del blanco sea cero. Por lo tanto, suponer que el lóbulo de la antena se mueve horizontalmente en incrementos de $\Delta\theta_a$ como se expresa en la figura (2.4B)

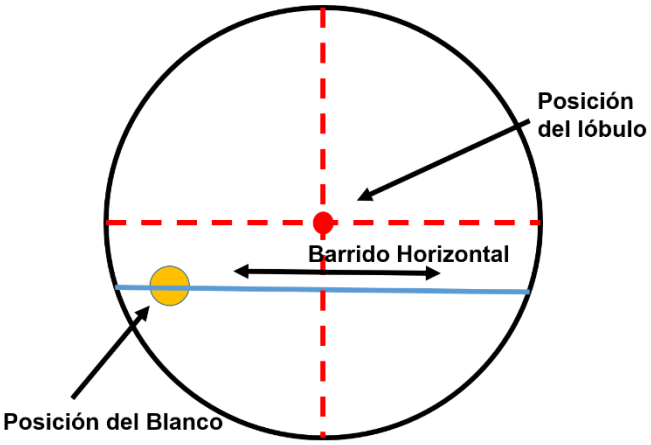


Figura (2.4 B) Área del lóbulo de radiación donde se encuentra el blanco

Cada movimiento del lóbulo se puede expresar como $\phi_0 + \Delta\theta_a$. Éste representa un movimiento al lado contrario de las manecillas del reloj. La ecuación de movimiento en el ángulo de azimut de la antena queda

$$\phi_{mov} = \phi_0 + n\Delta\theta_a \qquad (\mathbf{2.41})$$

donde *n* es el número de movimientos realizados. Ahora, si decimos que la señal reflejada es cero (eco=0) cuando n =N, entonces el ángulo de azimut del blanco será

$$\phi_t = \phi_0 + N.\Delta\theta_a \qquad (\mathbf{2.42})$$

Por lo tanto, el error en el ángulo de azimut ϕ_{error} será definido como

$$\phi_{error} = \phi_T - \phi_0$$
$$\therefore \phi_{error} = N.\Delta\theta_a \qquad (\mathbf{2.43})$$

Cuando el blanco se encuentre en el límite del área de radiación contrario al punto de partida del movimiento del lóbulo, entonces se obtendrá el máximo error en el ángulo de azimut $\phi_{error\ max}$ definido como

$$\boldsymbol{\phi_{error\ max} = \phi_0 + N_{max}.\Delta\theta_a} \qquad (\mathbf{2.44})$$

Donde $N_{max} = \theta_a/\Delta\theta_a$ es el número máximo de movimientos del lóbulo necesario para cubrir el área del ancho del lóbulo en azimut a los 3dB

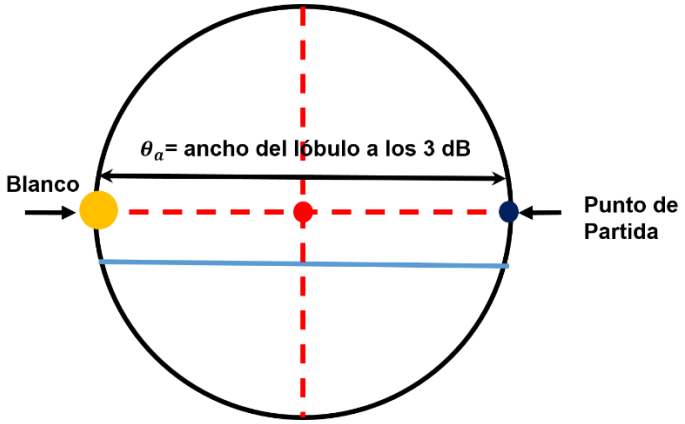


Figura (2.3 C) Muestra la localización del blanco cuando ocurre el error máximo

Por lo tanto el error máximo en el ángulo de azimut $\phi_{error\ max}$ será

$$\phi_{error\ max} = \phi_0 + \theta_a/2 \qquad (\mathbf{2.45})$$

Barrido vertical (elevación) [2]**:** En este caso el lóbulo de radiación se moverá verticalmente hacia arriba hasta que el eco reflejado del blanco sea cero. Por lo tanto, suponiendo movimientos verticales en incrementos de $\Delta\theta_e$ como se expresa en la figura 2.3 D

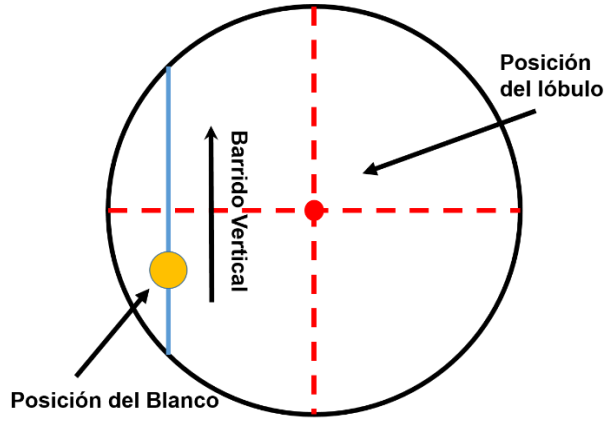


Figura (2.3 D) Área del lóbulo de radiación donde se encuentra el blanco

La ecuación de movimiento en el ángulo de elevación de la antena queda

$$\theta_{mov} = \theta_0 + m\Delta\theta_e \qquad (\mathbf{2.46})$$

Donde *m* es el número de movimientos realizados. Ahora, si decimos que la señal reflejada es cero (eco=0) cuando m=M, entonces el ángulo de elevación del blanco será

$$\theta_t = \theta_0 + M.\Delta\theta_e \qquad (\mathbf{2.47})$$

Por lo tanto, el error en el ángulo de elevación $\boldsymbol{\theta_{error}}$ será definido como

$$\theta_{error} = N.\Delta\theta_e \qquad (\mathbf{2.48})$$

Y el error máximo en la posición será

$$\phi_{errormax} = \phi_0 + \theta_e/2 \qquad (\mathbf{2.49})$$

2.4 Efecto Piezoeléctrico en el Sensor

Los sensores ultrasónicos emplean el fenómeno de la piezoelectricidad, esto es, cuando se deforman algunos materiales sólidos generan dentro de ellos una carga eléctrica. Este efecto es reversible en el sentido que, al aplicar una carga al sensor, éste se deformará mecánicamente como respuesta.

La forma del movimiento efectuado depende de la forma y orientación del cuerpo con relación a los ejes de los cristales y la posición de los electrodos. Los electrodos metálicos se recubren con otros metales para unirlos al material piezoeléctrico y aplicarles o extraerles la carga eléctrica. Como los materiales piezoeléctricos son aisladores, los electrodos se convierten en placas de un capacitor. Por tanto, un elemento piezoeléctrico que se emplea para convertir movimiento en señales eléctricas, puede considerarse como generador de carga y de forma general modelarse como un capacitor. La deformación mecánica genera una carga, y ésta se convierte en un voltaje definido que aparece entre los electrodos de acuerdo con la ley general de los capacitores.

El efecto piezoeléctrico es sensible a la dirección, porque la tensión produce una polaridad definida en el voltaje, mientras que la compresión produce una opuesta. Así pues, si al transductor piezoeléctrico de un sensor ultrasónico, con los cortes requeridos, se le aplica en sus extremos (electrodos) un voltaje, el cristal sufre cambios en sus dimensiones, lo que ocasiona un cambio de presión en el medio que lo rodea (en este caso el aire), y viceversa, al ser sometido el cristal a un cambio de presión aparecen cargas eléctricas en sus extremos, donde se crea una diferencia de potencial. Por lo que este tipo de transductor puede funcionar como emisor ó receptor ultrasónico.

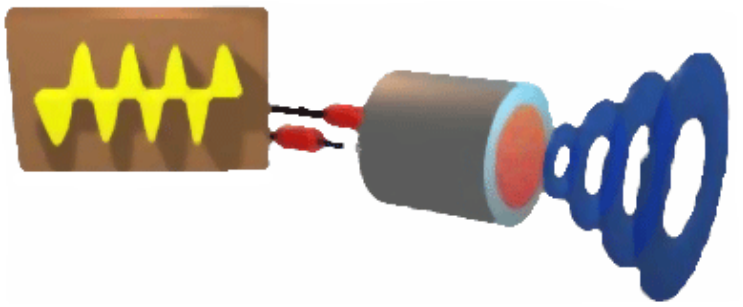


Figura 2.4 Generación de una onda ultrasónica debido al efecto piezoeléctrico.

CAPITULO III INTERACCIÓN Y RECEPCIÓN DE LA SEÑAL

La mayor parte de sistemas de radar usan la Sección Transversal del objeto (RCS - Radar Cross Section), como medio de discriminación para detectar objetos. Por lo tanto, la predicción exacta de la RCS de objetos es crítica para diseñar y desarrollar algoritmos de discriminación robustos. Adicionalmente la medición e identificación del centro de dispersión para un blanco dado, ayuda en el desarrollo de técnicas de reducción de la RCS. Así mismo, debido a que el objeto es detectado mediante el nivel de la señal reflejada del blanco, se determina la relación señal a ruido (SNR). Puesto que, mediante el Cálculo de este parámetro, es posible obtener la potencia mínima o el voltaje umbral necesario para detectar cierto tipo de blanco.

3.1 Sección Transversal de Radar (RCS)

Cuando las ondas electromagnéticas interactúan sobre un blanco (objeto) son dispersadas en todas direcciones. La medida de este campo electromagnético dispersado a una distancia lejana se conoce como Sección Transversal de Radar. La sección transversal de radar puede verse como una comparación entre la potencia del campo reflejado por un blanco que se dirige al radar, con la potencia del campo reflejado que tendría una esfera perfectamente lisa de área de $1m^2$. Esta puede ser considerada como una medida de área efectiva que da una referencia de la capacidad que tiene un blanco de reflejar señales en la dirección del radar. En la tabla se presentan datos experimentales típicos de RCS para distintos Blancos.

Tabla 3.1 Valores típicos de RCS para distintos blancos.

BLANCO	σ [m^2]
Insecto	10^{-5}
Pájaro	0.01
Misil pequeño	1
Hombre	1
Avión Pequeño	2
Avión Caza	10
Avión Comercial	40

3.1.1 Definición de la RCS

Suponiendo que la densidad de potencia de una onda sobre un blanco localizada en una distancia *R* lejos del radar es $P_{Di.}$ La cantidad de potencia reflejada del blanco es

$$P_r = \sigma P_{Di} \quad (\mathbf{3.1})$$

σ denota la sección transversal del blanco. Si decimos que P_{Dr} es la densidad de potencia de las ondas dispersadas a la antena receptora. Se deduce que

$$P_{Dr} = \frac{P_r}{4\pi R^2} \quad (\mathbf{3.2})$$

Sustituyendo la ec (3.1) en (3.2) Obtenemos

$$\sigma = 4\pi R^2 \left(\frac{P_{Dr}}{P_{Di}}\right) \quad (\mathbf{3.3})$$

y para asegurar que la antena receptora del radar está en el campo lejano, es decir, que las ondas reflejadas recibidas por la antena son planas, la ecuación anterior queda como

$$\sigma = 4\pi R^2 \lim_{R\to\infty} \left(\frac{P_{Dr}}{P_{Di}}\right) \quad (\mathbf{3.4})$$

La ecuación anterior define la sección transversal de radar dispersada de un blanco. Dicha RCS es la medida de todas las ondas dispersadas por el blanco que tienen la misma polarización que la antena receptora. Por lo tanto, σ representa una parte del total de la sección transversal dispersada del blanco σ_t, donde $\sigma_t > \sigma$. Asumiendo un sistema de coordenadas esférico definido por (ρ, θ, φ) entonces la sección transversal dispersada del blanco en una distancia ρ que está en función de (θ, φ). Si decimos que los ángulos (θ_i, φ_i) definen la dirección de propagación de las ondas incidentes. Además que los ángulos (θ_r, φ_r)definen la dirección de propagación de las ondas dispersas. Entonces, el total de RCS dispersado o reflejado del blanco está dado por

$$\sigma_t = \frac{1}{4\pi} \int_{\varphi_s=0}^{2\pi} \int_{\theta_s=0}^{\pi} \sigma(\theta_s, \varphi_s) sin\theta_s d\theta d\varphi_s \quad (\mathbf{3.5})$$

La cantidad de ondas reflejadas de un blanco es proporcional al tamaño del blanco con respecto a la longitud de onda λ de las ondas incidentes. De hecho, un radar no será capaz de detectar blancos más pequeños que la longitud de onda que este tenga. Por ejemplo, si los radares meteorológicos usan la frecuencia en banda-L las gotas de lluvia se vuelven casi invisibles al radar ya que éstas son mucho más pequeñas que la longitud de onda

CAPÍTULO IV DESARROLLO DEL PROTOTIPO

4.1 DISPOSITIVOS Y DESARROLLO

Como se observa en la figura 4.1 se usa una fuente de alimentación externa de 5v. Esto ayuda a darle energía al servo debido a que si está conectado en el pin que entrega 5V en el Arduino se obtienen medidas erróneas, se necesita dar un pulso al servo para su funcionamiento por eso es necesario conectarlo en algún pin de Arduino que genere un PWM conectándolo aquí en el pin 9, se conecta también el sensor ultrasónico el cual cuenta con dos pines (Echo y Trig) las cuales se conectan a los pines digitales en este caso 7 y 8 del Arduino

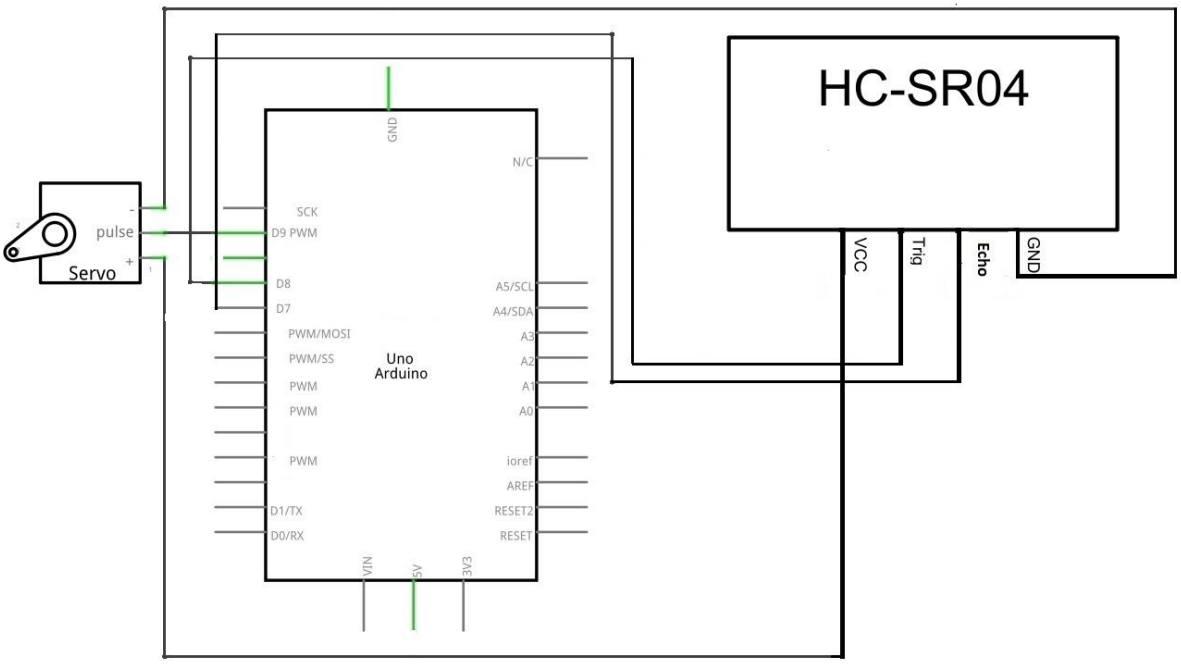


Figura 4.1 Esquema del prototipo

4.1.1 PROGRAMACIÓN DE ARDUINO

Arduino es una plataforma de hardware de código abierto, basada en una sencilla placa con entradas y salidas, analógicas y digitales, cuenta con 14 pines digitales de entrada y salida (de los cuales 6 se pueden utilizar como salidas PWM), 6 entradas analógicas, un cristal de cuarzo de 16 MHz. Arduino Uno es una placa electrónica basada en ATmega328P con un entorno de desarrollo que está basado en el lenguaje de programación Processing.

El uso del void setup y el void loop en Arduino es de carácter obligatorio, por lo que no será posible escribir un algoritmo en esta plataforma sin contar con dichas funciones. Cada vez que abrimos una nueva ventana contando con las dos funciones escritas por defecto.

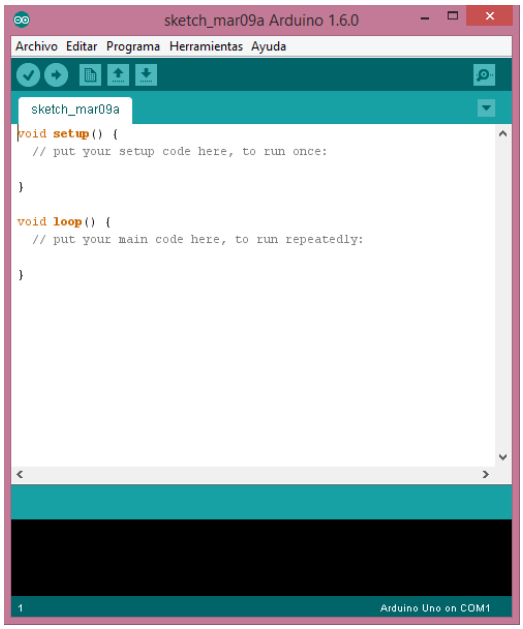


Figura 4.1 Ventana de entorno de Arduino

void setup()

El setup es la primera función en ejecutarse dentro de un programa en Arduino. Es, básicamente, donde se "setean" las funciones que llevará a cabo el microcontrolador.

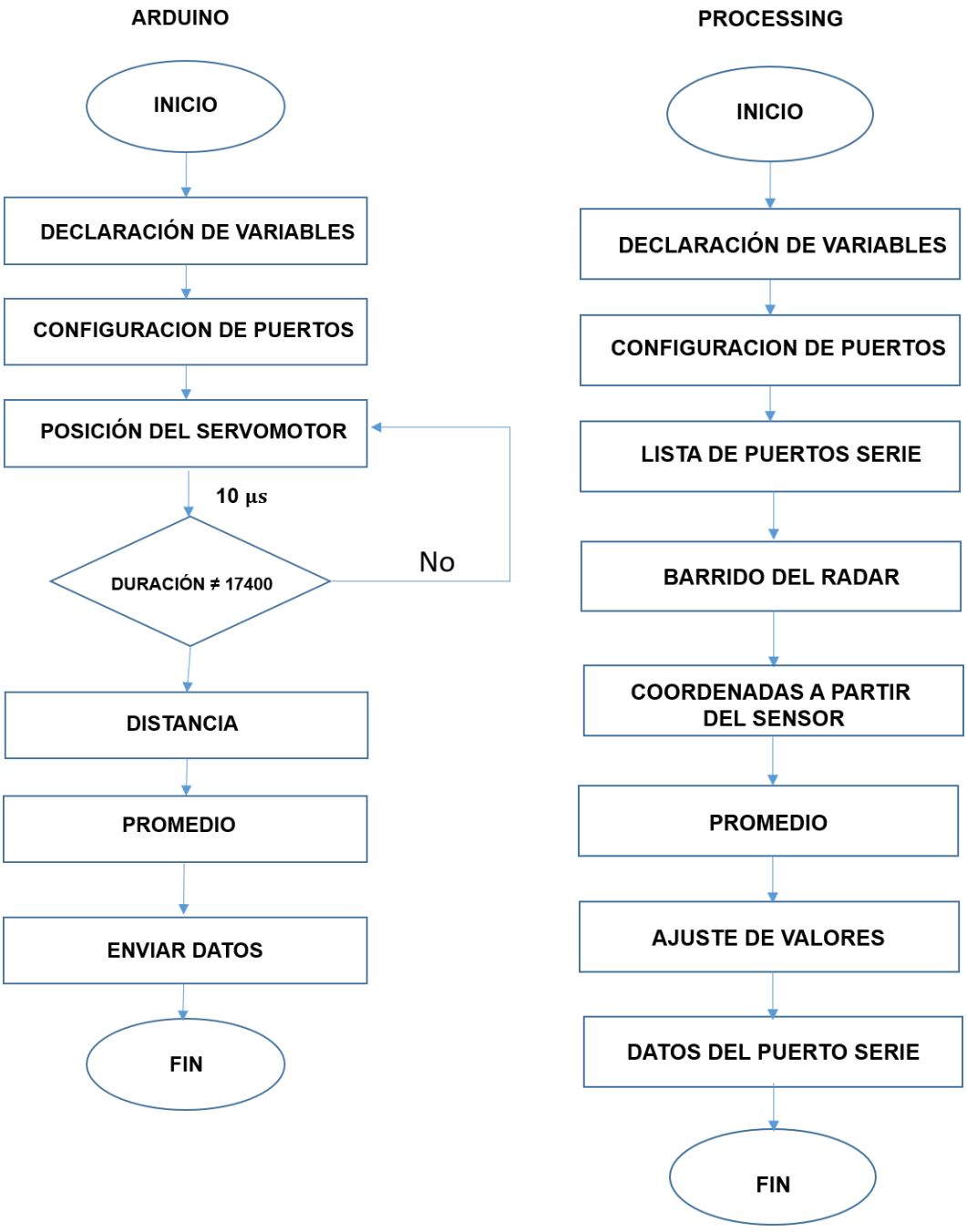
Aquí es donde se establece criterios que requieren una ejecución única. Por ejemplo, si nuestro programa va a usar comunicación serial, en el setup establecemos el comando Serial.begin para indicarle al programa que vamos a iniciar la comunicación.

Si vamos a utilizar un pin determinado como salida de voltaje, usamos el pinMode para indicarle a Arduino que determinado pin funcionará como salida, usando el parámetro OUTPUT.

void loop()

Loop en inglés significa lazo o bucle. La función loop en Arduino es la que se ejecuta un número infinito de veces. Al encenderse el Arduino se ejecuta el código del setup y luego se entra al loop, el cual se repite de forma indefinida hasta que se apague o se reinicie el microcontrolador.

4.1.2 DIAGRAMAS DE FLUJO



4.1.3 SERVO MOTOR SG9

Figura 4.2 Servomotor SG9

Un servomotor se diferencia de un motor de CD normal en que los servomotores solo pueden dar un giro 180° máximo, pero con la peculiaridad de poder guardar una posición en específico.

Por lo general, los servos tienen 3 cables. Dos de ellos son para la alimentación, y el tercero es el cable por el que mandaremos la señal desde Arduino para que el servo se mueva o se coloque en una determinada posición.

Para hacer esto, se necesita un pulso PWM, en este caso se conecta al pin 9 del arduino, se configura para mandar hacer esta función, añadimos la librería Servo.h y se crea una variable para el servomotor e iniciar su recorrido de izquierda a derecha, y otra para determinar la posición en que se encuentre este.

Se crea un ciclo for para contar paso a paso la posición del servo de izquierda a derecha y repetir varias veces el incremento en cada posición

```
#include <Servo.h>
Servo Servo180;
int Posicion = 0;
int contador = 0;

int PinServo = 9;
 int servoMin = 700;
int servoMax = 2400;
void setup() {
Servo180.attach(PinServo,700,2400);
```

```
void loop() {
   for(Posicion = 0; Posicion < 180; Posicion++) {
   Servo180.write(Posicion);
      for (contador = 0; contador<numLecturas; contador++) {
```

servoMin y **servoMax** permiten establecer la duración mínima y máxima del pulso de salida para el servo correspondiente a 0° y 180°. Por defecto, Arduino fija en 544 el ancho del pulso para el ángulo mínimo y en 2400 el ancho de pulso para el ángulo mayor.

4.1.4 SENSOR HY-SRF04

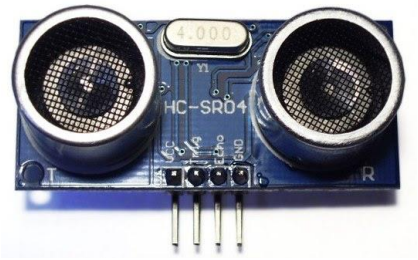


Figura 4.3 Sensor ultrasónico HY-SRF04

El funcionamiento de este módulo es muy sencillo. Esta alimentado con 5V y se debe suministrar un pulso de 10µs para activar el modulo a través del pin Trig. En ese momento, el modulo lanzará una ráfaga de 8 pulsos ultrasónicos a 40Khz y la salida Echo pasa a nivel alto hasta que el modulo recibe un eco, momento en el que volverá de nuevo a pasar a un nivel bajo. Por tanto, la salida Echo es un pulso cuyo ancho será proporcional a la distancia respecto a un objeto. Si no se detecta un objeto, la salida Echo pasara a nivel bajo después de 30ms.

Si el ancho del pulso se mide en µs, el resultado se debe dividir entre 58 para saber la distancia en centímetros, y entre 148 para saber la distancia en pulgadas [5].

Estos valores son obtenidos de:

$$distancia = \frac{(tiempo\ x\ velocidad\ del\ sonido)}{2} (\mathbf{4.1})$$

Si la velocidad del sonido es 340 metros por segundo se obtiene que es equivalente a

$$\frac{1}{340\ m} = 0.0029\mu s = 29\mu s (\mathbf{4.2})$$

29µs por centímetro, y como el sonido tiene que viajar dos veces la distancia hacia el objeto, una de ida y otra de vuelta, entonces cada 2x29=58µs recorrerá un centímetro.

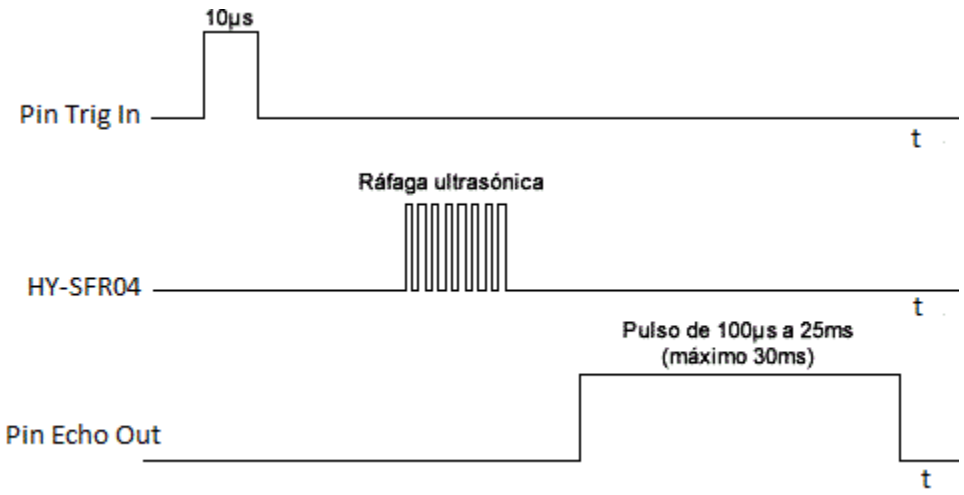


Figura 4.4 Funcionamiento del sensor ultrasónico

Aquí se configura el pulso de 10 µs para inicial el modulo, se establece una distancia máxima con valor de 17400 µs (3 metros) y se calcula la distancia en centímetros de acuerdo a la ecuación (4.1).

```
    digitalWrite(PinTrig, HIGH);
    delayMicroseconds(10);
    digitalWrite(PinTrig, LOW);

    duracion = pulseIn(PinEcho, HIGH, 17400);
       if (!duracion){
       duracion = 17400;          }
    distancia = duracion/58;
    total = total + distancia;
    delay(50);       }

  promedio = total/numReadings;
  total = 0;
```

delay() es una espera en milisegundos
delayMicroseconds() es una espera en microsegundos
digitalWrite que se utiliza para activar las salidas digitales
numReadings es una variable de Arduino que permite configurar cuantas medidas se tomarán en cada posición del servo. El valor en esta posición será un promedio de los valores obtenidos. Esto se usa para evitar posibles fallos en la medida. Cuanto más aumente este valor se obtendrá una medida más exacta en puntos en los que puedan existir reflexiones o multitrayectos del sonido. El aumentar este valor implica un mayor tiempo empleado para rastrear una zona.

4.1.5 COMUNICACIÓN SERIAL

El HY-SFR04 que use, con ayuda de un servo, puede hacer un barrido de 180° para explorar un área. Arduino se encarga de tomar las medidas y con un programa desarrollado en Processing se muestran los datos en la PC. La comunicación Arduino - Processing se realiza mediante el puerto serie.
A través de processing se visualiza la pantalla de un radar con un barrido de 180° midiendo la distancia de un área y visualizando si se producen cambios.
Para el puerto serie es necesario que Arduino envía una serie de parámetros por el puerto serie que Processing debe descomponer e interpretar. Estos parámetros son Grados y Distancia.

Figura 4.5 Mediciones mostradas en COM de Arduino

Se declaran las variables promedio y posición, se establece un identificador de impresión a las lecturas y operaciones que haga el programa (“Promedio” y “Distancia”) esto servirá para la comunicación serial con processing y pueda detectar los valores que necesitamos convertir

```
int promedio = 0;
int Posicion = 0;

   Serial.print("Posicion");
   Serial.print(Posicion);
   Serial.print("Distancia");
   Serial.println(promedio);
```

4.1.6 COMUNICACIÓN SERIAL A TRAVÉS DE PROCESSING

Processing es una herramienta / IDE que permite programar gráficos y animación, además de ser software libre es fácil de usar y muy poderoso ya que la mejor parte es que trabaja de la mano con Arduino en el mismo estilo de C / C ++ usando el mismo sistema de programación, pudiendo así tomar los datos de Arduino y todo lo que está enchufado en él mediante comunicación serial para visualizarlo en la pantalla, en este caso como una pantalla de radar.
Primero se configuraron las variables y se cargaron en las bibliotecas de puerto serie para asegurar de que se puedan leer los datos enviados por el Arduino. También se incluye una función de la biblioteca en serie llamada serialEvent(), que llama los datos que se envían y permite leer los datos fácilmente. En pocas palabras se divide para la obtención de la posición del servo motor que mide los grados y el valor del sensor que mide la distancia, que son los parámetros que se obtienen mediante el COM de arduino.

Estas funciones nos permiten una comunicación serie entre arduino y processing el cual nos establece un puerto serie.

```
import processing.serial.*;
Serial myPort;
```

Se configuraron las variables para, almacenar valores de coordenadas, y para almacenar los valores nuevos que vaya registrando el servomotor, además de que algunas variables nos sirven para establecer el ancho y el radio de los objetos.

```
float x, y;
int radio = 350;
int ancho = 300;
int grados = 0;
int valorsensor = 0;
int movimiento = 0;
int[] nuevo = new int[181];
int[] viejo = new int[181];
PFont fuente;
int radarDistancia = 0;
int primerbarrido = 0;
int rc = 10;
```

Se configura la pantalla para que se visualice como un radar, dibujando el texto, las medidas y el cuadriculado. Con processing se utiliza la función draw() apareciendo primero todo secuencialmente desde la parte superior.
Así que las líneas y el texto estarán situadas en la parte inferior para que siempre estén visibles. Asignando también un fondo, tamaño y fuente de letra para visualizar en la ventana

```
void setup(){
size(750, 450);
background (0);
fuente = createFont("Arial", 12);
```

```
textFont(fuente);
  println(Serial.list());
myPort = new Serial(this, Serial.list()[1], 9600);
myPort.bufferUntil(retornocarro);
```

Dibujando la pantalla

```
  void draw(){
  fill(0);
  noStroke();
```

Se dibuja una elipse con una anchura y altura de 750 con posición central establecidos en el radio

```
  ellipse(radio, radio, 750, 750);
  rectMode(CENTER);
```

Se dibuja una recta con las posiciones (X,Y,Ancho, Alto)

```
  rect(350,402,800,100);
```

Si en el extremo derecho, “movimiento= 1” empezara a girar de derecha a izquierda y si el servo está en 0 empezara de izquierda a derecha

```
 if (grados >= 179)
movimiento = 1;
  if (grados <= 1)
  movimiento = 0;
```

Para dibujar las líneas y medidas se utilizaron ciclos FOR. La función draw () dibuja un marco, Para comenzar a mostrar los valores en pantalla. Se usó un ciclo FOR para recorrer cada elemento de la matriz, newValue y oldValue, están configurados para guardar 181 valores (1 punto por cada posición del servo con 1 adicional por si acaso, haciendo un ciclo a través de estos para mostrar constantemente las lecturas anteriores).

Si tuviera que usar la propia posición del servo para recorrer la matriz entonces no habría datos previos entonces esto haría que la posición del servo está siempre cambiando.

Cuando se trabaja con un círculo unitario, no se utiliza el sistema de coordenadas cartesiano, En su lugar se utiliza un sistema de coordenadas polares. En un sistema de coordenadas polares, la ubicación se especifica por (r, θ), donde r es el radio y θ es el ángulo de rotación. El círculo unitario tiene su origen en su centro, y se miden los ángulos de rotación comenzando en el borde medio derecho del círculo unitario y moviéndose en dirección contraria a las agujas del reloj a su alrededor.

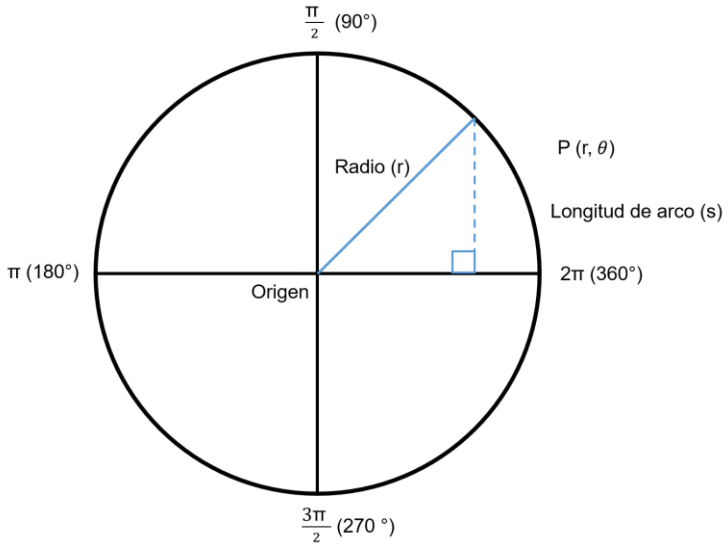


Figura 4.6 Circulo unitario

Usando la trigonometría se crean los puntos alrededor de un circulo. Así el radio más el coseno de la posición del servo se convierte a radianes, Desde que inicia en 0 hasta 90 grados se añaden 180 para volver a empezar desde la derecha. Se añade +1(i) cada vez que rebobina para hacerlo coincidir de nuevo con el servomotor. Ya que es un semicírculo se plotean líneas y vértices desde el origen

Figura 4.7 Relación entre el circulo unitario y función seno

Para calcular las coordenadas X e Y de cada posición del servo y la lectura a distancia del sensor, se usa la trigonometría. Utilizando seno, coseno y la conversión de la posición del servo a un radián usando el sensor de lectura como la distancia desde el centro para dibujar el punto. Coseno es para los valores en X de izquierda a derecha. Seno es para los valores de Y.

Configuración del barrido del radar

En la configuración del barrido del radar se establece un espesor de las líneas que estarán en movimiento y con el ciclo FOR se dibujan 20 líneas para formar un barrido uniforme un grado menos que la anterior, además de “stroke” nos permite asignar el color que queramos a las líneas utilizando los valores RGB (en este caso solo se usó el verde), con “else” se establece el barrido de derecha a izquierda a derecha y volviendo a colorear las líneas en el nuevo barrido

```
strokeWeight(7);
  if (movimiento == 0) {
    for (int i = 0; i <= 20; i++) {
     stroke(0, (10*i), 0);
      line(radio, radio, radio + cos(radians(grados+(180+i)))*ancho, radio +
sin(radians(grados+(180+i)))*ancho);
    }
  } else {
    for (int i = 20; i >= 0; i--) {
      stroke(0,200-(10*i), 0);
      line(radio, radio, radio + cos(radians(grados+(180+i)))*ancho, radio +
sin(radians(grados+(180+i)))*ancho);
    }
```

Para obtener los valores de puerto serie Arduino envía una serie de parámetros por el puerto serie que Processing debe descomponer e interpretar. Estos parámetros son la posición y la distancia, donde la posición y distancia son datos numéricos y son los valores que detecta el sensor y que nos muestra en el COM de Arduino

Se obtiene el valor de la posición en el servo y el valor de la lectura del sensor, estableciendo estos valores a las nuevas variables se va a almacenar en los acarreos

```
String ObtenerPosicion = xString.substring(1, xString.indexOf("Distancia"));
String ObtenerDistancia = xString.substring(xString.indexOf("Distacia")+1, xString.length());
grados = Integer.parseInt(ObtenerPosicion);
valorsensor = Integer.parseInt(ObtenerDistancia);
viejo[grados] = nuevo[grados];
nuevo[grados] = valorsensor;
```

La pantalla del radar quedaría de la siguiente manera

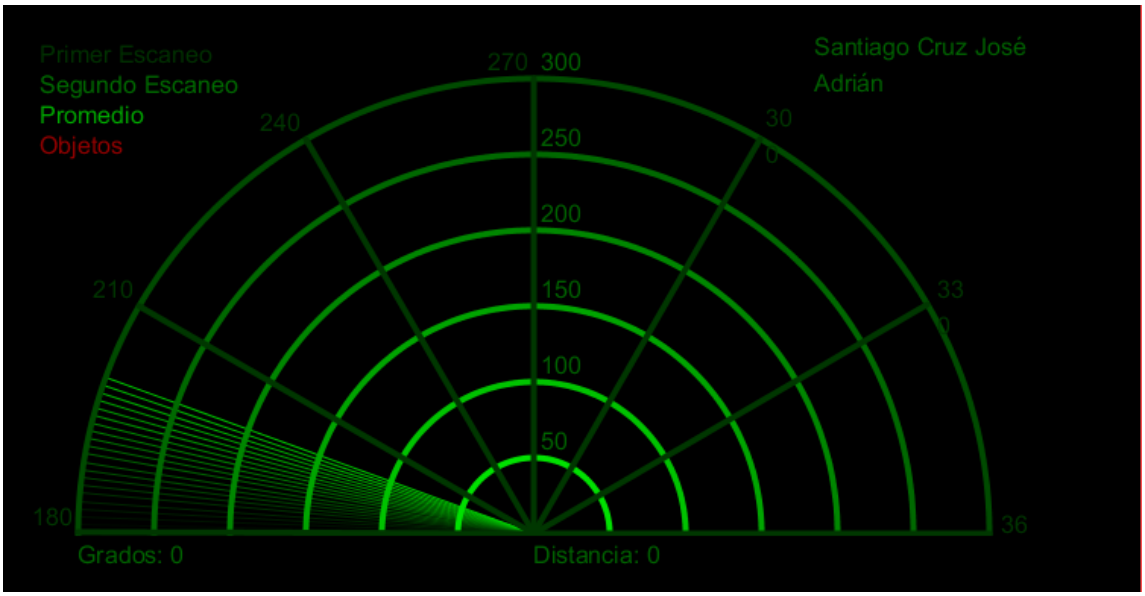


Figura 4.8 Visualización de radar

4.2 PRUEBAS

Como se puede apreciar en la figura 4.9, se aprecia que hay una abertura de las cajas aproximado de 0 a 20° y de 165° a 180° donde se aprecia en la figura 4.9 la distancia a la que se encuentran otros objetos detectables.

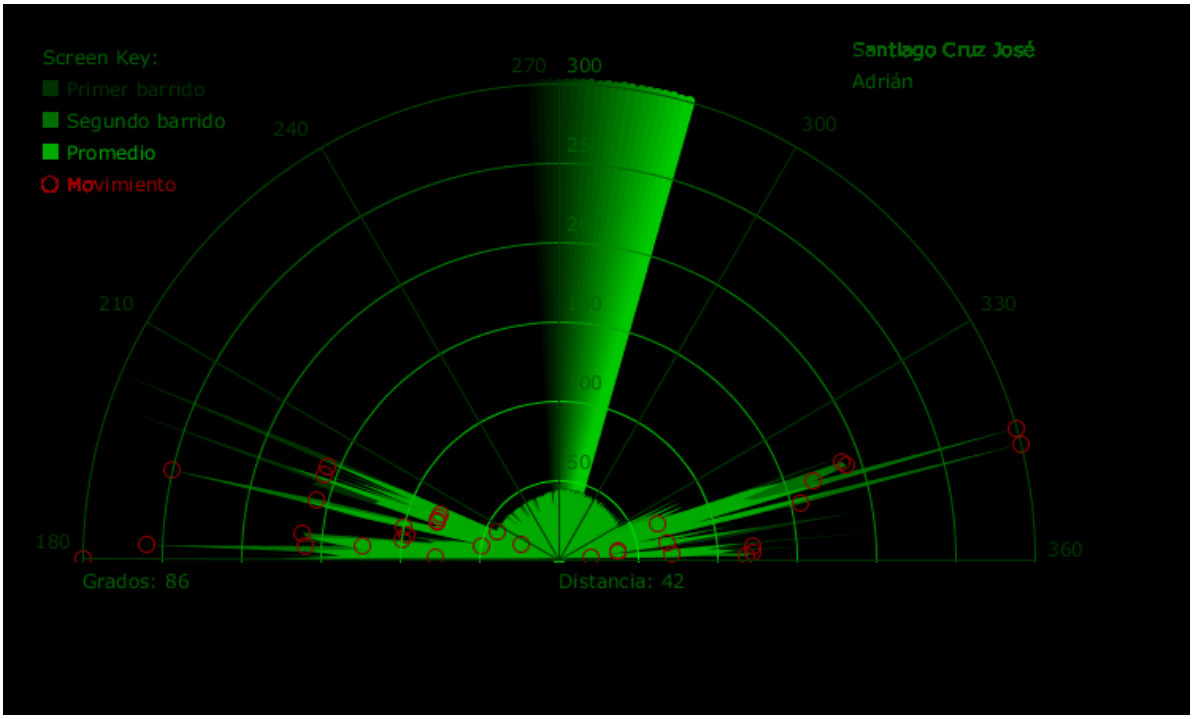


Figura 4.9 Prueba de la visualización del radar

En la figura 4.10 se aprecia la distancia de las cajas a una distancia de 50 cm y en la figura 4.9 se observa el área cubierta por las cajas a una distancia aproximada de 50 cm

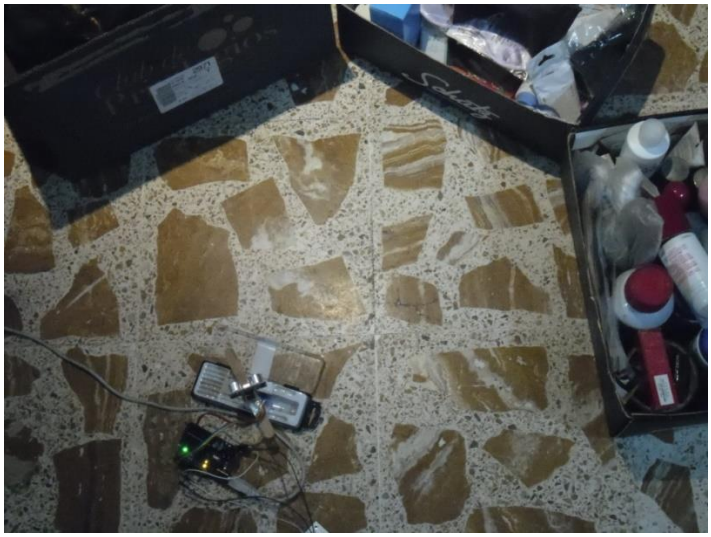


Figura 4.10 Radar en operación con objetos

Prueba 2

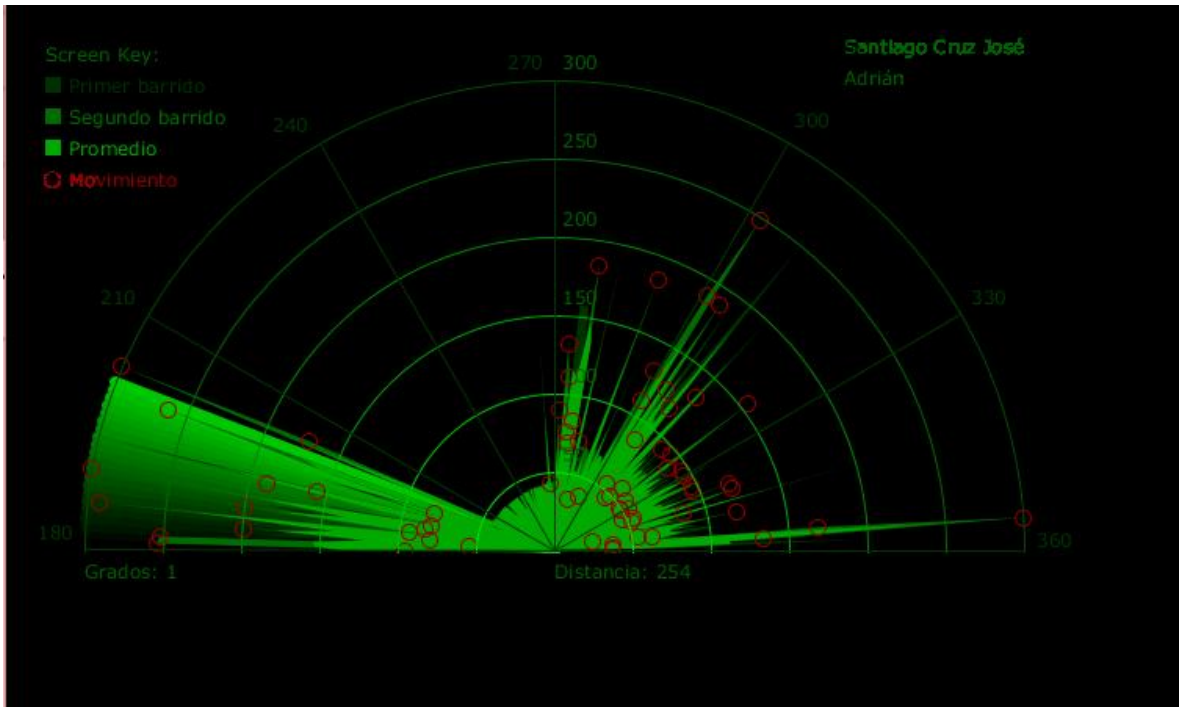


Figura 4.9 Segunda prueba del radar

Comparando la figura 4.11 con la figura 4.11 se puede apreciar la distancia que es aproximada a los 50 cm que es lo que muestra la figura 4.10 aproximadamente a los 30°

Figura 4.11 Radar en operación con objeto

4.3 COSTOS

Arduino 1	$250
Sensor Ultrasónico HC-SRF04	$70
Servomotor SG 90	$70
Otros materiales:	$100
Costo total del material	**$490**

Salario Promedio de un Ingeniero recién egresado mensualmente:	$10,000
Salario por hora de un Ingeniero recién egresado:	$62.5
Horas completas para el desarrollo del trabajo:	200 Hrs
Costo por diseño y construcción (horas x salario):	**$12,500**

Costo total:	**$12,990**

CONCLUSIONES

En el desarrollo del proyecto se logra al objetivo de diseñar y construir un prototipo capaz de detectar objetos que se encuentren a su paso. Esto se llevó a cabo gracias al estudio de las ondas ultrasónicas, y funcionamiento de un radar, además se logró con la ayuda de dispositivos que hoy en día se pueden conseguir en cualquier tienda de electrónica. A pesar de haber logrado el objetivo, en el proyecto se muestra aun un margen de error, comparando la medida real con la que se muestra en pantalla, además no existe una restricción en cuanto a los objetos que pueda detectar, prácticamente todos los materiales que reflejan el sonido son detectados, independientemente de su forma o color, aun materiales transparentes o delgados no presentan algún problema para el radar, como la distancia hacia el objeto es medida por medio del tiempo de recorrido del sonido, y no por una medición de la intensidad, el sensor ultrasónico es insensible al ruido de fondo, pudiendo funcionar así hasta en medios polvorientos, viscosos etc.

Con un mayor tiempo de análisis se puede lograr realizar una guía y trazado de superficies y áreas nubosas, se puede obtener una visualización de distancia o volumen, lo cual es ideal para superficies estrechas y atmosferas corrosivas como por ejemplo áreas donde haya ocurrido un accidente nuclear y de este método poder escanear la zona a una distancia remota lejos del peligro para la vida humana, cabe mencionar que al inicio del proyecto uno de los objetivos era detectar personas, ya que como sabemos el ruido ambiente no afecta la operación del ultrasónico y con ayuda de ondas electromagnéticas que se propagan a través de los muros y otros obstáculos se podría medir los cambios en el medio de propagación y analizar las modificaciones en los distintos tipos de ondas, y así tal vez encontrar la presencia de una persona, hoy en día, por ejemplo, se observan las radiografías con ultrasonido, es por eso que el campo de exploración es grande y está en desarrollo.

El desarrollo de este proyecto me ayudo a poner en practica algunas de las cualidades que aprendí a lo largo de mis estudios, física, matemáticas, programación etc., con una serie de fracasos, análisis, expectativas esto me ayudo a formar un criterio más amplio sobre la ingeniería y mi propio razonamiento en cuanto a mi educación profesional.

BIBLIOGRAFIA

[1] “Introducción al Sistema del Radar”, Skolnik M. L. Editorial McGRAW-HILL, Tercera Edicion, 2001.

[2] “Radar Systems Analysis and Design using Matlab”, Mahafza, B. R. Editorial CHAPMAN & HALL/CRC, Boca Raton,2005.

[3] Tutorial para socializarse con processing
https://processing.org/tutorials/gettingstarted/

[4] Tutorial para formas trigonométricas en processing
https://processing.org/tutorials/trig/

ANEXO 1
PROGRAMA ARDUINO

```
#include <Servo.h>
Servo Servo180;
int PosicionIzquierdaADerecha = 0;
int contador = 0;
long total = 0;
int promedio = 0
long duracion = 0;
int distancia = 0;
int PinEcho = 7;
int PinTrig = 8;
int PinServo = 9;
int servoMin = 455;    // Anchura del pulso, en microsegundos, correspondiente al mínimo (0 grados)
ángulo en del servo (por defecto 544)
int servoMax = 2400;   // Anchura del pulso, en microsegundos, correspondiente al máximo (180 grados)
ángulo en del servo (por defecto 2400)
const int numLecturas = 2;
void setup() {
Servo180.attach(PinServo,455,2400); // Pin de salida para el servo, recorrido mínimo, recorrido máximo
Serial.begin(9600);   // Establece la velocidad de datos del puerto serie
pinMode(PinTrig, OUTPUT);
pinMode(PinEcho, INPUT);
digitalWrite(PinTrig, LOW);   // Pone el pin a un estado lógico bajo
}

void loop() {
  for(PosicionIzquierdaADerecha = 0; PosicionIzquierdaADerecha < 180;
PosicionIzquierdaADerecha++) {
  Servo180.write(PosicionIzquierdaADerecha);
     for (contador = 0; contador<numLecturas; contador++) {   // Repite tantas veces como número de
lecturas en cada posición
     digitalWrite(PinTrig, HIGH);
     delayMicroseconds(10);
     digitalWrite(PinTrig, LOW);

     duracion = pulseIn(PinEcho, HIGH, 17400);   // Devuelve la longitud del pulso del pin Echo en us
(3metros maximo)
        if (!duracion){
        duracion = 17400;
        }
     distancia = duracion/58;
     total = total + distancia;
     delay(50);    // Tiempo de espera hasta la siguiente medida
     }
   promedio = total/numLecturas;
  total = 0;    // Resetea la variable
```

```
// Envia datos por el puerto serie //

   Serial.print("Posicion");
   Serial.print(PosicionIzquierdaADerecha);
   Serial.print("Promedio");
   Serial.println(promedio);
  }
   for(PosicionIzquierdaADerecha = 180; PosicionIzquierdaADerecha > 0;
PosicionIzquierdaADerecha--) {
   Servo180.write(PosicionIzquierdaADerecha);
      for (contador = 0; contador<numLecturas; contador++) {

// Pulso de 10us para inicial el modulo//

      digitalWrite(PinTrig, HIGH);
      delayMicroseconds(10);
      digitalWrite(PinTrig, LOW);

      duracion = pulseIn(PinEcho, HIGH, 17400);
         if (!duracion){
         duracion = 17400;
         }
      distancia = duracion/58;
      total = total + distancia;
      delay(50);
      }

   promedio = total/numLecturas;
   total = 0;

   Serial.print("Posicion");
   Serial.print(PosicionIzquierdaADerecha);
   Serial.print("Promedio");
   Serial.println(promedio);
   }
}
```

ANEXO 2
PROGRAMA PROCESING

```
import processing.serial.*;
Serial myPort;  // Declara el puerto serie
float x, y;
int radio = 350;
int ancho = 300;
int grados = 0;
int valorsensor = 0;
int movimiento = 0;
int[] nuevo = new int[181]; // Matriz para almacenar cada valor nuevo de cada posición del servo
 int[] viejo = new int[181];    // Matriz para almacenar cada valor previo de cada posición del servo
PFont fuente;
int radarDistancia = 0;
int primerbarrido = 0;
int retornocarro = 10;
void setup(){
size(750, 450); // Establece el tamaño de la ventana
 background (0);    // Establece a negro del fondo de la ventana
fuente = createFont("Arial", 12);
textFont(fuente);
println(Serial.list());
myPort = new Serial(this, Serial.list()[1], 9600);
myPort.bufferUntil(retornocarro);  // Almacena en el bufer hasta llegar un retorno de carro
}

// Dibujando la pantalla //

void draw(){
  fill(0);
  noStroke();
  ellipse(radio, radio, 750, 750);     // Dibuja un círculo con una anchura y altura = 750 con posición
central (x y) establecidos en el radio
  rectMode(CENTER);
  rect(350,402,800,100); // Dibuja un rectángulo (x, y, ancho, alto)
  if (grados >= 179) {
  movimiento = 1;
  }
  if (grados <= 1) {
  movimiento = 0;
  }

// Configuración del barrido de radar //

  strokeWeight(7);
  if (movimiento == 0) {
    for (int i = 0; i <= 20; i++) {
      stroke(0, (10*i), 0); // Establece el trazo de color (rojo, verde, azul) dependiendo el valor de i
```

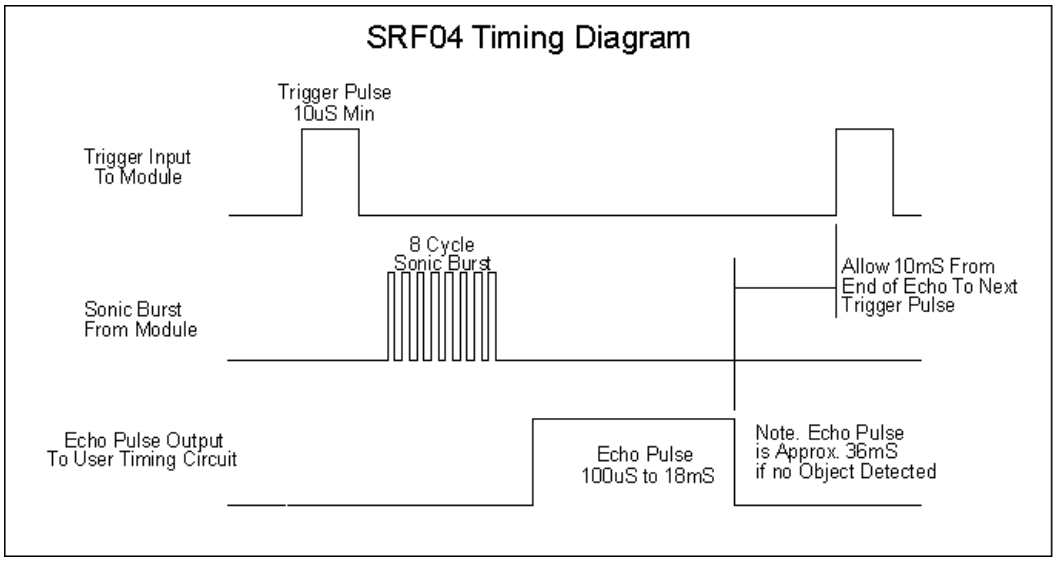
```
    line(radio, radio, radio + cos(radians(grados+(180+i)))*ancho, radio +
sin(radians(grados+(180+i)))*ancho); // Se establecen coordenadas (inicio x, inicio y, fin x, fin y)
   }
  } else {
   for (int i = 20; i >= 0; i--) {
    stroke(0,200-(10*i), 0); // utilizamos valores estándar RGB, entre 0 y 255
    line(radio, radio, radio + cos(radians(grados+(180+i)))*ancho, radio +
sin(radians(grados+(180+i)))*ancho);
   }
  }

// Configuración de las formas hechas a partir de los valores de los sensores //

  noStroke();
  fill(0,50,0); // establece el color de relleno (Rojo, Verde, Azul)
  beginShape();   //empieza a dibujar la forma
   for (int i = 0; i < 180; i++) {   // por cada grado en la matriz
    x = radio + cos(radians((180+i)))*((viejo[i])); // Crea cordenadas x
    y = radio + sin(radians((180+i)))*((viejo[i])); // Crea cordenadas y
    vertex(x, y);
   }
  endShape();

// Segundo Barrido //

  fill(0,110,0);
  beginShape();
   for (int i = 0; i < 180; i++) {
    x = radio + cos(radians((180+i)))*(nuevo[i]);
    y = radio + sin(radians((180+i)))*(nuevo[i]);
    vertex(x, y);
   }
  endShape();

// Promedio //

  fill(0,170,0);
  beginShape();
   for (int i = 0; i < 180; i++) {
    x = radio + cos(radians((180+i)))*((nuevo[i]+viejo[i])/2);
    y = radio + sin(radians((180+i)))*((nuevo[i]+viejo[i])/2);
    vertex(x, y);
   }
  endShape();

// Despues de los dos barridos, resalta con un círculo rojo //

  if (primerbarrido >= 360) {
```

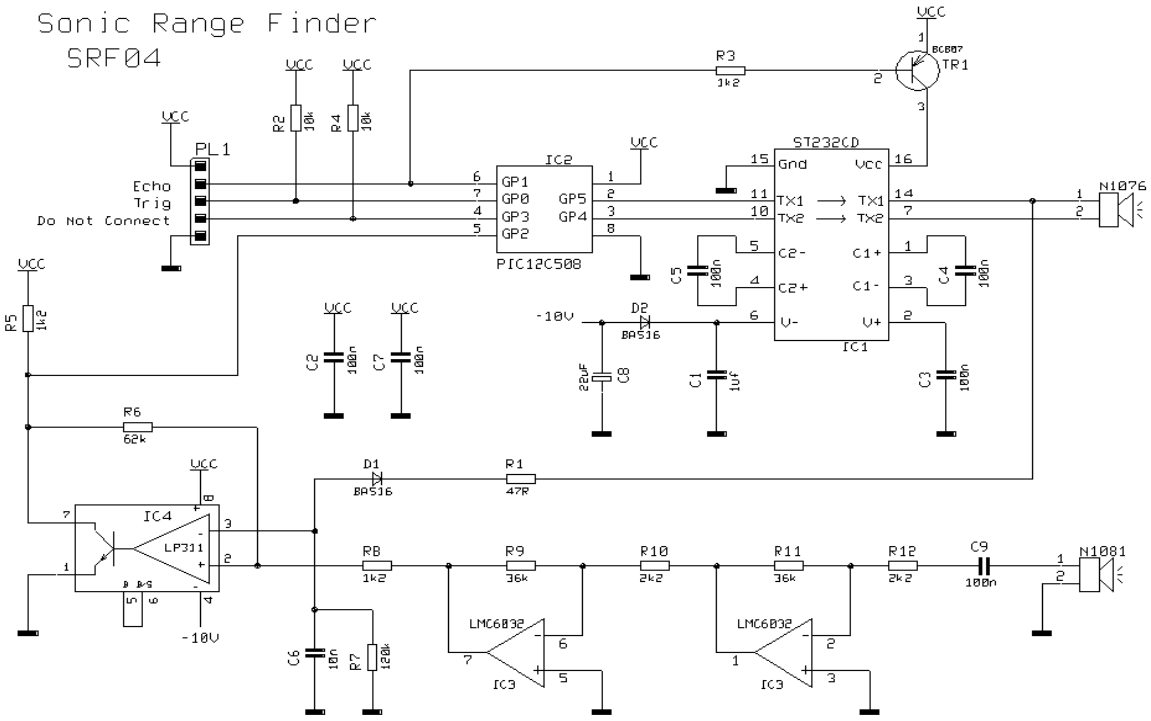
```
   stroke(150,0,0);
   strokeWeight(1);
   noFill();
    for (int i = 0; i < 180; i++) {
      if (viejo[i] - nuevo[i] > 35 || nuevo[i] - viejo[i] > 35) {
        x = radio + cos(radians((180+i)))*(nuevo[i]);
        y = radio + sin(radians((180+i)))*(nuevo[i]);
        ellipse(x, y, 10, 10);
      }
    }
  }

// Ajusta los valores de la distancia y afuera pone los valores, 50, 100, 150 etc.. //

  for (int i = 0; i <=6; i++){
   noFill();
   strokeWeight(1);
   stroke(0, 255-(30*i), 0);
   ellipse(radio, radio, (100*i), (100*i));
   fill(0, 100, 0);
   noStroke();
   text(Integer.toString(radarDistancia+50), 380, (305-radarDistancia), 50, 50);
   radarDistancia+=50;
  }
  radarDistancia = 0;

// Dibuja las líneas de la cuadrícula en el radar cada 30 grados y escribe sus valores de 180, 210,
240, etc .. //

  for (int i = 0; i <= 6; i++) {
   strokeWeight(1);
   stroke(0, 55, 0);
   line(radio, radio, radio + cos(radians(180+(30*i)))*ancho, radio +
sin(radians(180+(30*i)))*ancho);
   fill(0, 55, 0);
   noStroke();
   if (180+(30*i) >= 300) {
    text(Integer.toString(180+(30*i)), (radio+10) + cos(radians(180+(30*i)))*(ancho+10), (radio+10)
+ sin(radians(180+(30*i)))*(ancho+10), 25,50);
   } else {
    text(Integer.toString(180+(30*i)), radio + cos(radians(180+(30*i)))*ancho, radio +
sin(radians(180+(30*i)))*ancho, 60,40);
   }
  }

// Escribe los valores de la información. //

  noStroke();
```

```
  fill(0);
  rect(350,402,800,100);
  fill(0, 100, 0);
  text("Grados: "+Integer.toString(grados), 100, 380, 100, 50 // Utilizamos Integet.toString para
convertir numéros en forma de texto ()
  text("Distancia: "+Integer.toString(valorsensor), 100, 400, 100, 50);
  text("Santiago Cruz José Adrián", 540, 380, 250, 50);
  fill(0);
  rect(70,60,150,100);
  fill(0, 100, 0);
  text("Datos:", 100, 50, 150, 50);
  fill(0,50,0);
  rect(30,53,10,10);
  text("Primer barrido", 115, 70, 150, 50);
  fill(0,110,0);
  rect(30,73,10,10);
  text("Segundo barrido", 115, 90, 150, 50);
  fill(0,170,0);
  rect(30,93,10,10);
  text("Promedio", 115, 110, 150, 50);
  noFill();
  stroke(150,0,0);
  strokeWeight(1);
  ellipse(29, 113, 10, 10);
  fill(150,0,0);
  text("Movimiento", 115, 130, 150, 50);
}

// para obtener los valores de puerto serie //

void serialEvent (Serial myPort) {
  String xString = myPort.readStringUntil(retornocarro);
    if (xString != null) {
       xString = trim(xString);
       String ObtenerPosicion = xString.substring(1, xString.indexOf("Promedio"));        String
ObtenerPromedio = xString.substring(xString.indexOf("Promedio")+1, xString.length());
       grados = Integer.parseInt(ObtenerPosicion);
       valorsensor = Integer.parseInt(ObtenerPromedio);
       viejo[grados] = nuevo[grados];
       nuevo[grados] = valorsensor;
       primerbarrido++;
       if (primerbarrido > 360) {
       primerbarrido = 360;
       }
  }
}
```

ANEXO 3
SRF04 - Ultra-Sonic Ranger
Technical Specification

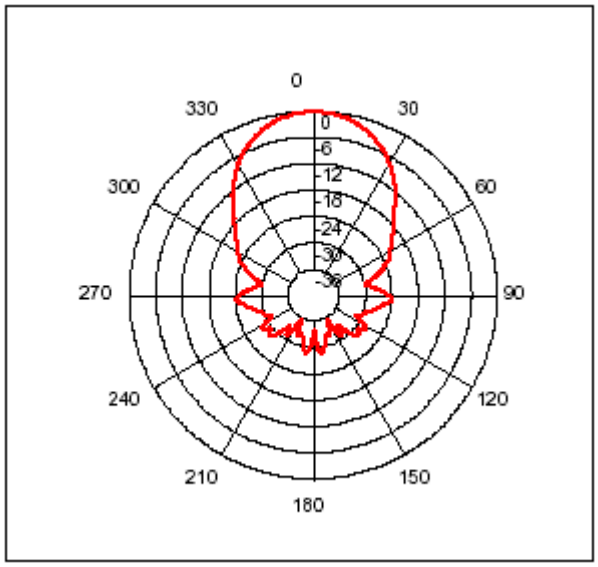
The SRF04 Timing diagram is shown below. You only need to supply a short 10uS pulse to the trigger input to start the ranging. The SRF04 will send out an 8 cycle burst of ultrasound at 40khz and raise its echo line high. It then listens for an echo, and as soon as it detects one it lowers the echo line again. The echo line is therefore a pulse whose width is proportional to the distance to the object. By timing the pulse it is possible to calculate the range in inches/centimeters or anything else. If nothing is detected then the SRF04 will lower its echo line anyway after about 36mS.





The receiver is a classic two stage op-amp circuit. The input capacitor C8 blocks some residual DC which always seems to be present. Each gain stage is set to 24 for a total gain of 576-ish. This is close the 25 maximum gain available using the LM1458. The gain bandwidth product for the LM1458 is 1Mhz. The maximum gain at 40khz is 1000000/40000 = 25. The output of the amplifier is fed into an LM311 comparator. A small amount of positive feedback provides some hysterisis to give a clean stable output.

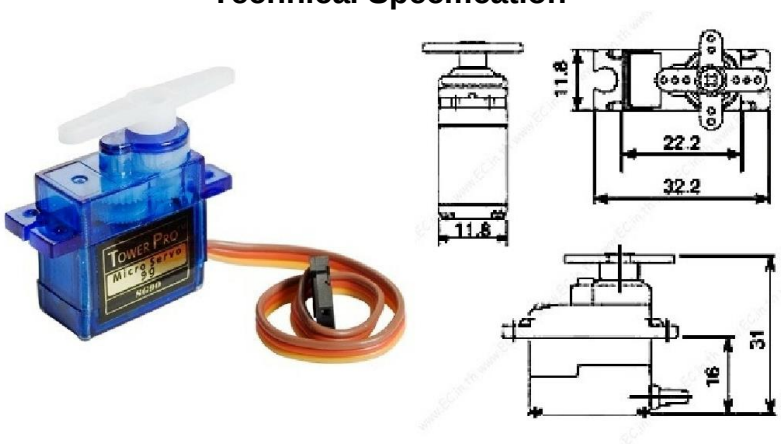
In operation, the processor waits for an active low trigger pulse to come in. It then generates just eight cycles of 40khz. The echo line is then raised to signal the host processor to start timing. The raising of the echo line also shuts of the MAX232. After a while – no more than 10-12mS normally, the returning echo will be detected and the PIC will lower the echo line. The width of this pulse represents the flight time of the sonic burst. If no echo is detected then it will automatically time out after about 30mS (Its two times the WDT period of the PIC). Because the MAX232 is shut down during echo detection, you must wait at least 10mS between measurement cycles for the +/- 10v to recharge.



Maximum range is a little over 3m. As and example of the sensitivity of this design, it will detect a 1inch thick plastic broom handle at 2.4m.
Average current consumption is reasonable at less than 50mA and typically about 30mA.

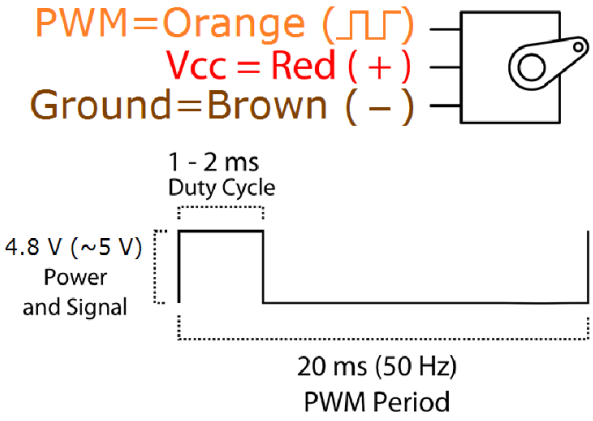
Calculating the Distance
The SRF04 provides an echo pulse proportional to distance. If the width of the pulse is measured in uS, then dividing by 58 will give you the distance in cm, or dividing by 148 will give the distance in inches. uS/58=cm or uS/148=inches.

ANEXO 4
DATASHEET SERVOMOTOR SG90
Technical Specification



Tiny and lightweight with high output power. Servo can rotate approximately 180 degrees (90 in each direction), and works just like the standard kinds but smaller. You can use any servo code, hardware or library to control these servos. Good for beginners who want to make stuff move without building a motor controller with feedback & gear box, especially since it will fit in small places. It comes with a 3 horns (arms) and hardware.

Specifications
- Weight: 9 g
- Dimension: 22.2 x 11.8 x 31 mm approx.
- Stall torque: 1.8 kgf·cm
- Operating speed: 0.1 s/60 degree
- Operating voltage: 4.8 V (~5V
- Dead band width: 10 µs
- Temperature range: 0 ºC – 55 ºC



Position "0" (1.5 ms pulse) is middle, "90" (~2 ms pulse) is all the way to the right, "-90" (~1 ms pulse) is all the way to the left

ANEXO 5
ARDUINO UNO
Technical Specification

Technical specs

Microcontroller	ATmega328P
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
PWM Digital I/O Pins	6
Analog Input Pins	6
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328P) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Clock Speed	16 MHz
Length	68.6 mm
Width	53.4 mm
Weight	25 g

Power

The board can operate on an external supply from 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may become unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.
The power pins are as follows:

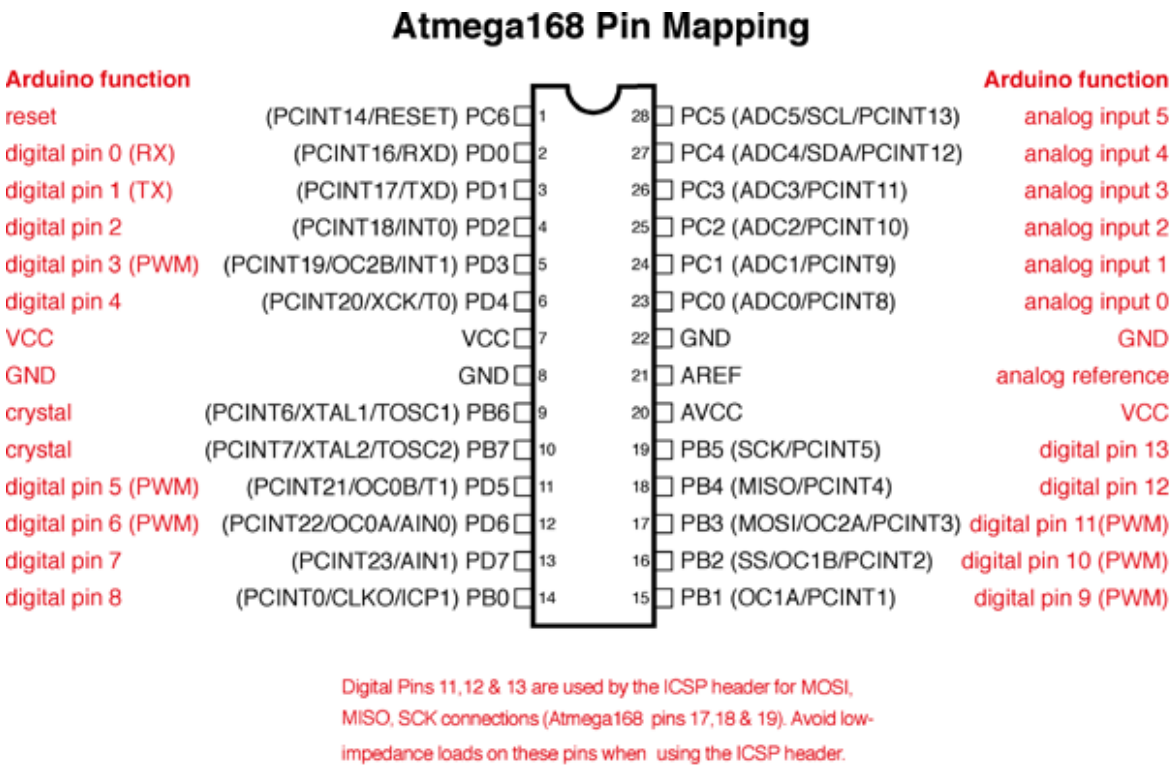
Vin. The input voltage to the Uno board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.
5V. This pin outputs a regulated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

Memory

The ATmega328 has 32 KB (with 0.5 KB occupied by the bootloader). It also has 2 KB of SRAM and 1 KB of EEPROM (which can be read and written with the EEPROM library).

Each of the 14 digital pins on the Uno can be used as an input or output, using pinMode(),digitalWrite(), and digitalRead() functions. They operate at 5 volts. Each pin can provide or receive 20 mA as recommended operating condition and has an internal pull-up resistor (disconnected by default) of 20-50k ohm. A maximum

of 40mA is the value that must not be exceeded on any I/O pin to avoid permanent damage to the microcontroller.



In addition, some pins have specialized functions:

Serial: 0 (RX) and 1 (TX). Used to receive (RX) and transmit (TX) TTL serial data. These pins are connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip.
External Interrupts: 2 and 3. These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the attachInterrupt() function for details.
PWM: 3, 5, 6, 9, 10, and 11. Provide 8-bit PWM output with the analogWrite() function.
SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). These pins support SPI communication using the SPI library.
LED: 13. There is a built-in LED driven by digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.
TWI: A4 or SDA pin and A5 or SCL pin. Support TWI communication using the Wire library.

The Uno has 6 analog inputs, labeled A0 through A5, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and the analogReference() function.
There are a couple of other pins on the board:

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

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