UNIVERSIDAD MIGUEL HERNÁNDEZ
ESCUELA POLITÉCNICA SUPERIOR DE ELCHE GRADO EN INGENIERÍA MECÁNICA

ESTUDIO COMPARATIVO DE APLICACIONES CON ARDUINO PARA LA ESTIMACIÓN DE POSICIÓN Y VELOCIDAD

TRABAJO FIN DE GRADO

JUNIO - 2022

AUTOR: Daniel Sogorb Torres

DIRECTOR: David Valiente García

ÍNDICE

ÍNDICE DE IMÁGENES

ÍNDICE DE TABLAS

1 INTRODUCCIÓN

A modo de introducción desde los albores de la humanidad, el ser humano ha tenido la necesidad de cotejar distintos tipos de objetos en su entorno. Ya sea por necesidad o por puro conocimiento, con ello nace el concepto de medición.

La medición se trata de una expresión numérica la cual se relaciona con las dimensiones de un objeto en términos de longitud. Además, esta tiene como referencia una unidad de medición ya sean metros, centímetros, pulgadas, pies, etc. Pero no solo se trata de tener una referencia en distancia, sino también el de duración.

Por ello para el presente proyecto se tendrán en cuenta dichos factores con los cuales tendremos la base del estudio, la estimación de distancia y velocidad de los objetos.

Gracias al desarrollo tecnológico tenemos la capacidad de medir y comprobar diversas magnitudes físicas. En este caso se centrará el foco en los sensores, elementos que permiten medir cualquier tipo de magnitud física y química, y de las cuales nos centraremos en la obtención de longitudes para posteriormente dar un salto al cálculo de la velocidad.

Ahondando en la historia del sensor de proximidad, fue Pepperl Fush en 1958 quien creó el primer dispositivo de obtención de distancias. Dicho elemento es de carácter inductivo, esto quiere decir que puede captar elementos ferrosos.

Desde entonces, el desarrollo e innovación tecnológica ha ido en aumento dando paso a incorporar dichos elementos no solo en el tejido industrial sino también a nivel de usuario. Por lo que gracias a esto ha sido posible realizar dicho proyecto.

En concreto, se explotará la capacidad de diversos sensores de proximidad para lograr calcular la velocidad de los objetos a partir de la generación de los códigos que permitan dicha función. Todo ello será expuesto en los posteriores epígrafes donde aparecerán tanto sus características, fundamentos físicos, los objetivos a cumplir y el posterior análisis de los resultados dados.

Podremos definir por tanto la viabilidad de implementarlos en un entorno dinámico después de los resultamos obtenidos en el laboratorio.

2 JUSTIFICACIÓN

En el presente proyecto sobre la comparativa entre distintos sensores de proximidad, se afrontará la necesidad de hacer frente al estudio de la obtención de la velocidad. Debido a la carencia de estudios, dicho análisis aportará resultados de gran ayuda para situar a estos sensores como una alternativa a las ya vigentes en el campo de adquisición de la velocidad.

Los sensores para implementar este proyecto deberán ser comprobados con el fin de analizar su fiabilidad y precisión para cubrir las necesidades dadas en los diversos ambientes que requieran de su uso, para ello se realizarán sus pertinentes ajustes quedando así listos para cumplir con su funcionamiento.

Y, ¿a qué se alude en lo referente al campo de adquisición de la velocidad? Cuando nos referimos a esto pueden surgir ejemplos claros como los cinemómetros [6] utilizados por la DGT (Dirección General de Tráfico), [7] el sistema GPS (Global Positioning System), [8] sistemas PIV (Particle Image Velocimetry), entre otros.

Lo que se pretende realizar con dicho estudio es una propuesta más económica y simple ya que su implementación y montaje no requiere una gran cantidad de elementos, además su programación es relativamente sencilla y fácil de adaptar a las diversas necesidades que se presenten. Ya sean en entornos controlados o en vías públicas e incluso en vías ferroviarias. De esta manera los sensores actuarán de una forma más optima y minimizando así su error.

Lo que acontece en los siguientes epígrafes servirá de preámbulo para los posteriores estudios con dichos sensores u otros de la misma índole e incluso una mejora y puesta en escena, tal y como se comenta en el párrafo anterior. Todo ello seguirá una serie de objetivos para que su uso sea intuitivo y su error mínimo, ofreciendo al usuario una herramienta práctica y económica.

2.1 Objetivos

Los objetivos que plantea dicho estudio se recogen de manera particular en cada prueba a realizar, no obstante, a grandes rasgos lo que se pretende conseguir con el proyecto es:

- Dotar de una nueva alternativa para la medición de objetos móviles con la herramienta Arduino que aportará un manejo sencillo y económico. Tanto el material como la configuración de estos no requiere de una habilidad elevada para su comprensión.
- Obtener el código de programación en el leguaje C++, cuya modificación a través de unos pocos parámetros permitirá afrontar las distintas pruebas a realizar. Además del montaje que resulta ser compacto y fácilmente transportable a cualquier lugar que se necesite.

- Realizar pruebas reales comparativas, extensibles a un entorno con un mayor número de variables del laboratorio a un ambiente exterior con gran éxito.

3 ESTADO DEL ARTE

Para el actual punto se han consultado diversos artículos con la idea de aportar las primeras bases del estudio a realizar. Con ellos se dará paso a entender cuál es su fundamento teórico y de qué manera se utilizarán los sensores que posteriormente serán descritos.

A continuación, se exponen un par de artículos que pondrán en contexto los sistemas de medición que se van a utilizar en el presente proyecto.

En este primer artículo [1] se realiza un estudio de sistemas de detección con sensores ultrasónicos. En él, se muestran distintos métodos de medición para los cuales se pretende detectar objetos que se encuentran a largas y cortas distancias. Se pretende diseñar una solución dirigida a vehículos con un bajo coste económico para niños discapacitados.

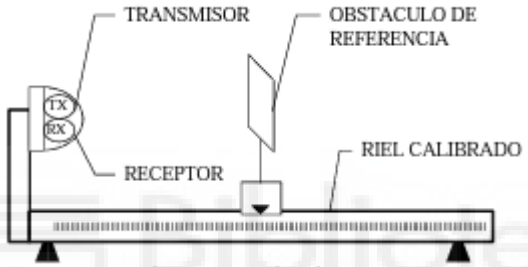


Imagen 1. Esquema detección de objetos

El fundamento para el cálculo de la distancia hace alusión a la expresión: $\boldsymbol{2 * X = v * t}$, donde X es la distancia a la cual se encuentran los objetos, v es la velocidad del sonido y t es el tiempo que tarda la onda en ser emitida y captada por el emisor y el receptor respectivamente.

Para el segundo artículo [2], se pretende poner en perspectiva los inicios del láser. Desde su propuesta teórica hasta su implementación en los primeros dispositivos, describiendo así los avances e investigaciones realizadas hasta la actualidad. Con ello se obtiene unas nociones de cómo funciona dicho dispositivo. Se trata de otro sensor capaz de determinar el tiempo transcurrido por una señal reflejada sobre un objeto, y a partir de ahí deducir su distancia y opcionalmente su velocidad.

A continuación, ahondando más en el tema del proyecto se referenciarán una serie de artículos relacionados con la adquisición de la velocidad por parte de sensores, además de otros elementos.

Comenzando por el tercer artículo [3] se investiga la posibilidad de realizar diversos esquemas ópticos utilizando fotodetectores con la finalidad de obtener sistemas de medición de velocidad. Por lo que con ello se pretende calcular a qué velocidad viaja una bala. Y desarrollando ecuaciones para definir los errores producidos en cada prueba. Los fotodetectores se utilizan para calcular el tiempo transcurrido por el objeto de manera que en el momento que la señal luminosa deja de ser captada por la fotocélula, se produce una variación de corriente eléctrica la cual queda refleja y tomando por tanto el instante de tiempo. Con ello se pretende estimar la velocidad de los objetos.

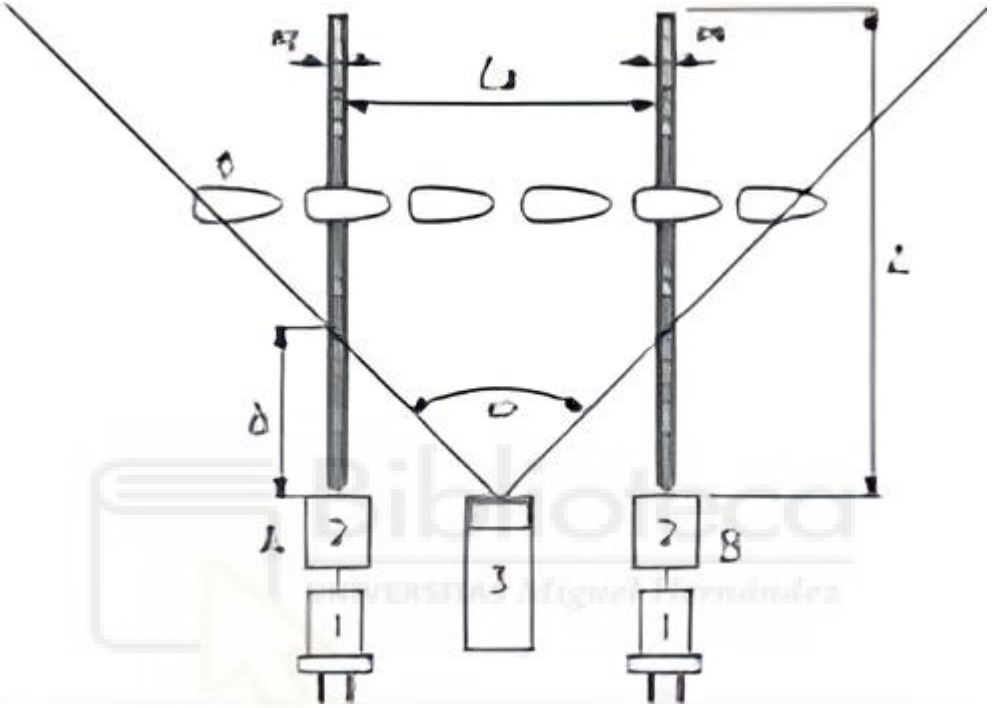


Imagen 2. Esquemas de sistemas de medición

En el siguiente artículo [4] se realiza un estudio comparativo para el cálculo de la velocidad entre el sistema GPS y un dispositivo de pistola láser. Se pretende realizar un estudio para un rango de valores entre 20 y 120 km/h. Además de ello los datos serán tratados a posteriori y se realizará un estudio estadístico para comprobar la fiabilidad de estos.

Característica	*Logger* GPS	Pistola Láser
Tipo de medición	Continua	Puntual
Rango de medición, km/h	0.01 – 1600	10 – 320
Precisión, km/h	0.2	2
Resolución de medición, km/h	0.01	1
Tiempo de captura de dato, s	0.1	0.3
Geo-referencia datos	si	no

Imagen 3. Características de los dispositivos

Finalizando el epígrafe se presenta el artículo [5] exponiendo un modelo de medición a raíz de un sistema basado en imágenes, en el que el cálculo se realiza de manera computacional. Las pruebas se ubicarán en una carretera transitada con un vehículo que incorpora un sistema GPS. Al parecer los

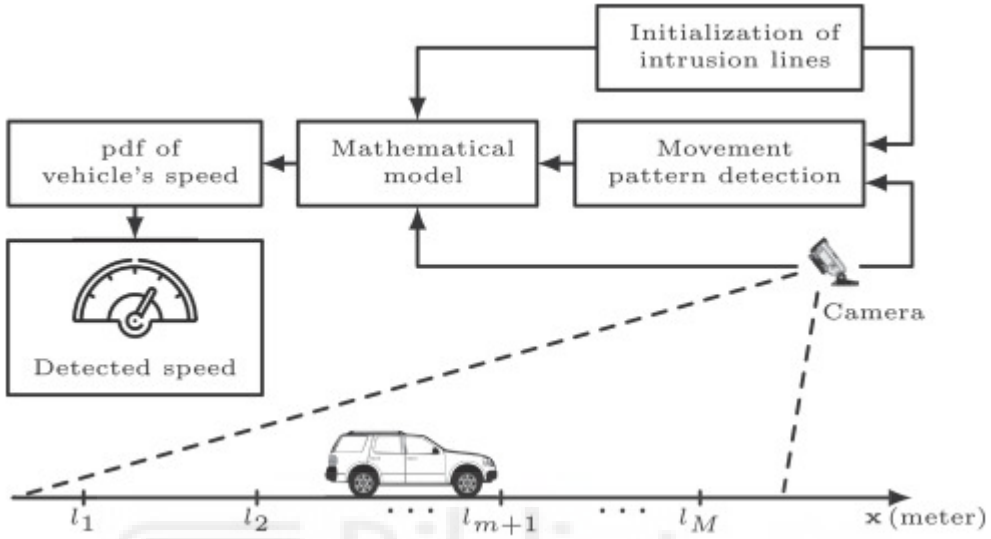


Imagen 4. Diseño experimental Artículo 5

resultados obtenidos son prometedores ya que apenas se presenta error en las mediciones.

3.1 Marco Teórico

Previo a los ensayos a realizar, se ha de tener en cuenta cuál es el funcionamiento y qué características poseen los elementos que se van a utilizar. Dicho esto, a continuación, se darán a conocer los dispositivos más relevantes utilizados para dicho estudio.

3.1.1 Sensor ultrasónico

Se define como sensor ultrasónico aquel dispositivo capaz de medir distancias a partir del uso de ondas ultrasónicas. En su estructura se encuentran los que denominaremos emisor y receptor. El primero de ellos el emisor, emitirá una onda como la mencionada anteriormente mientras que el segundo elemento el receptor recibirá dicha onda, sobre el cual esta será reflejada.

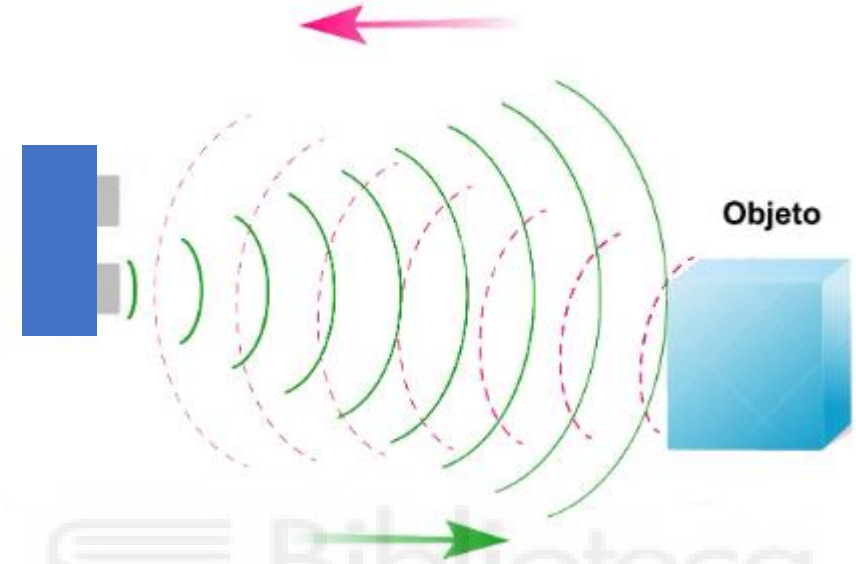


Imagen 5. Funcionamiento del HC-SR04

Por tanto, ¿cómo se obtiene la distancia de los objetos? Gracias a la expresión: $\boldsymbol{distancia} = \frac{\mathbf{1}}{\mathbf{2}} * \boldsymbol{c} * \boldsymbol{t}$, donde $\boldsymbol{t}$ es el tiempo transcurrido entre la emisión y recepción de la onda y $\boldsymbol{c}$ es la velocidad del sonido 343 m/s para 20°C. Como se observa en la expresión, está multiplicada por $\frac{\mathbf{1}}{\mathbf{2}}$, esto se debe a que el tiempo medido es el ida y vuelta de la onda.

Para las primeras 3 pruebas realizadas en este estudio se ha utilizado el sensor ultrasónico HC-SR04 [9]. Es necesario conocer cómo funcionan sus conexiones, las cuales seguidamente serán mostradas:

Imagen 6. Conexiones HC-SR04

3.1.2 Sensor láser

Se define como láser (*Light Amplification by Stimulated Emission of Radiation*) aquel dispositivo óptico capaz de generar un haz luminoso. Esto se produce debido por el bombeo óptico a los elementos dentro del medio activo. Se producirá por tanto una excitación de los fotones con lo generarán energía en forma de luz. Todo ello se observa en la siguiente imagen:

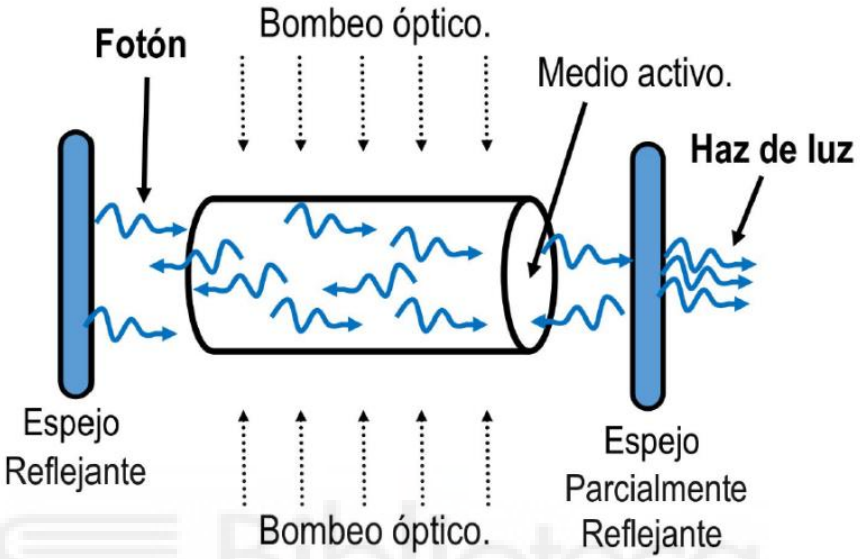


Imagen 7. Esquema del láser

Ahora bien, ¿cómo se obtiene la distancia de los objetos? Para obtener a qué distancias se encuentran los objetos se ha de utilizar nuevamente la siguiente expresión: $\boldsymbol{distancia} = \frac{1}{2} * \boldsymbol{c} * \boldsymbol{t}$, donde $\boldsymbol{t}$ es el tiempo transcurrido entre la ida y la vuelta desde el láser al objeto, por lo que tiene que ser dividido entre dos. Además, $\boldsymbol{c}$ hace referencia a la velocidad de la luz.

Para la realización de las últimas 3 pruebas utilizaremos el sensor laser Lidar Lite V3 [10]. Para el cual se tendrán en cuenta las múltiples conexiones y funcionamientos

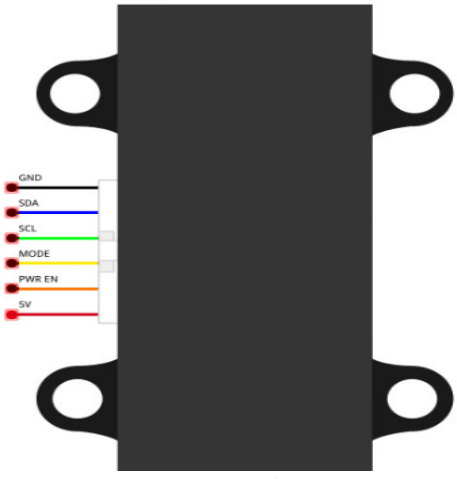


Imagen 8. Conexiones Lidar Lite V3

4 IMPLEMENTACIÓN

A continuación, se explicarán las diversas pruebas realizadas tanto con el sensor HC-SR04 como con el Lidar Lite V3. Además, se detallará su montaje así como la evolución que ha seguido el proyecto desde las primeras pruebas en un entorno controlado hasta su puesta en marcha en el taller.

También será necesario aclarar los sistemas de referencia utilizados para comprender el estado de las pruebas realizadas.

Por una parte, tendremos las pruebas **frontales** donde el objeto, el cual será un neumático para todas las pruebas, se aproximará frontalmente hacia los sensores como se muestra en los siguientes esquemas:

Imagen 9. Avance Frontal del neumático

Por otro lado, se realizarán además las pruebas cuando el neumático viaje en dirección **perpendicular** a la dirección de adquisición de los sensores como se muestra en la siguiente imagen:

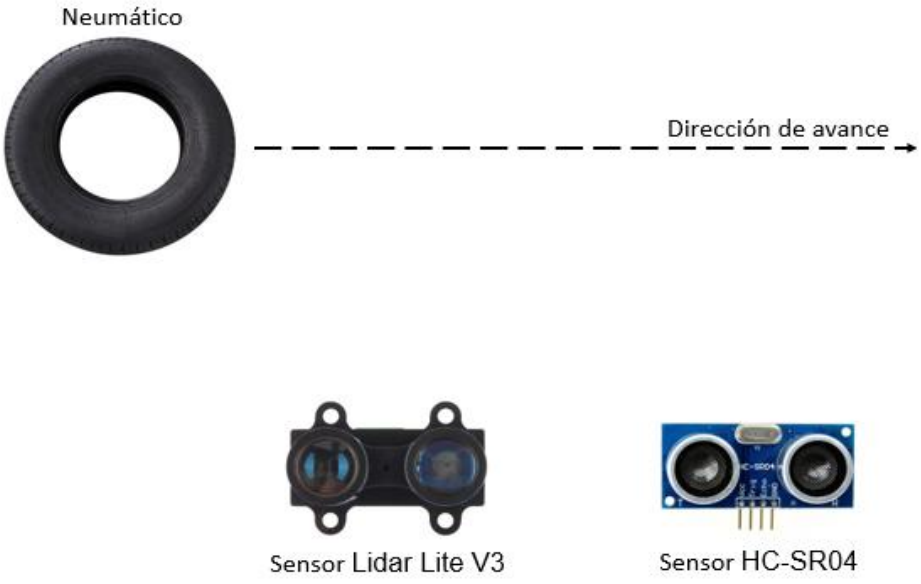


Imagen 10. Avance Perpendicular del neumático

4.1 Prueba 1. Rango del sensor HC-SR04

4.1.1 Objetivos

En esta primera prueba se pretende obtener el cumplimiento de una serie de objetivos:

- Mediante la utilización de un sensor HC-SR04, obtener a qué distancia se encuentran los objetos ubicados dentro de su rango.
- Conocer cómo funciona el sensor anteriormente mencionado.
- Programar un código en lenguaje C++ que nos permita adquirir dichos valores de distancia, visualizarlos a través del Monitor Serial.
- Examinar la detección del objeto en el momento que se adentra en un cierto rango de valores dado que posteriormente dicha información será útil para la realización de las consecutivas pruebas.

4.1.2 Componentes

A continuación, se detalla el equipamiento necesario para el primer montaje:

- **Arduino Mega 2560**

El Arduino Mega 2560 [11] es una placa de desarrollo basada en el microcontrolador ATmega2560. El microcontrolador establece la conexión entre las instrucciones y los elementos del circuito. Una vez el código esté en funcionamiento, el microprocesador será el encargado de controlar los elementos del circuito. Arduino Mega 2560 dispone de 54 entradas/salidas digitales, de las cuales 15 pueden ser usadas como salidas PWM, 16 entradas analógicas, 4 transmisores/receptores asíncrono universal, un cristal de 16Mhz, conexión USB, 1 conector para alimentación DC, conector ICSP, y un botón de reinicio. La placa Mega 2560 es compatible con la mayoría de las placas de expansión para Arduino UNO.

Imagen 11. Componente Placa de Arduino

- **Placa de pruebas o protoboard**

Es un elemento que sirve para realizar las conexiones de los elementos del circuito con la placa de Arduino sin la necesidad de soldadura. Se compone de líneas de conexión, buses de alimentación y un carril central.

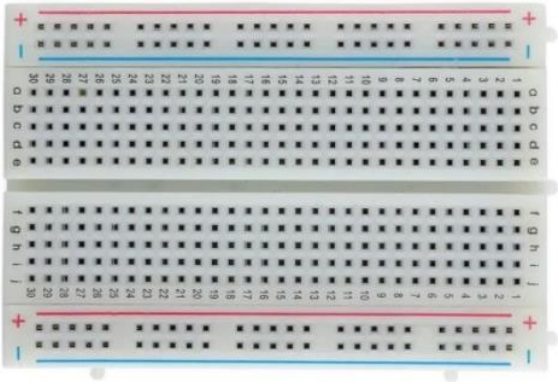


Imagen 12. Componente Protoboard

- **Led rojo**

Fuente de luz compuesto por dos terminales ánodo y cátodo. Cuando recibe una tensión de 2-3 voltios, emite luz debido a un salto de los electrones desde el ánodo hasta el cátodo.

Imagen 13. Componente Led

- **Resistencia 330Ω**

Oposición al flujo de corriente eléctrica en un circuito con el objetivo de modificar o alterar el paso de esta.

Imagen 14. Componente Resistencia

- **Sensor HC-SR04**

Sensor ultrasónico capaz de detectar objetos y medir la distancia a la que se encuentran estos. Posee un rango de acción de 2 a 400 cm, precisa de una alimentación de 5V, contiene 4 conexiones 2 de ellas van para el positivo y el negativo, Vcc y GND respectivamente. Además de 1 conexión trigger, encargada de emitir el pulso ultrasónico y finalmente, 1 conexión echo, la cual recibe la onda. Este sensor dispone de ángulo de apertura alrededor de los 15 grados.

Imagen 15. Componente Sensor ultrasónico

- **Cables dupont macho-macho**

Permiten la conexión entre la placa de Arduino con los sensores, servomotores, pantallas o protoboard.

Imagen 16. Componente Cables

4.1.3 Montaje

En cuanto al montaje se ha representado un esquema del circuito tanto en la plataforma online Tinkercad como de forma física.

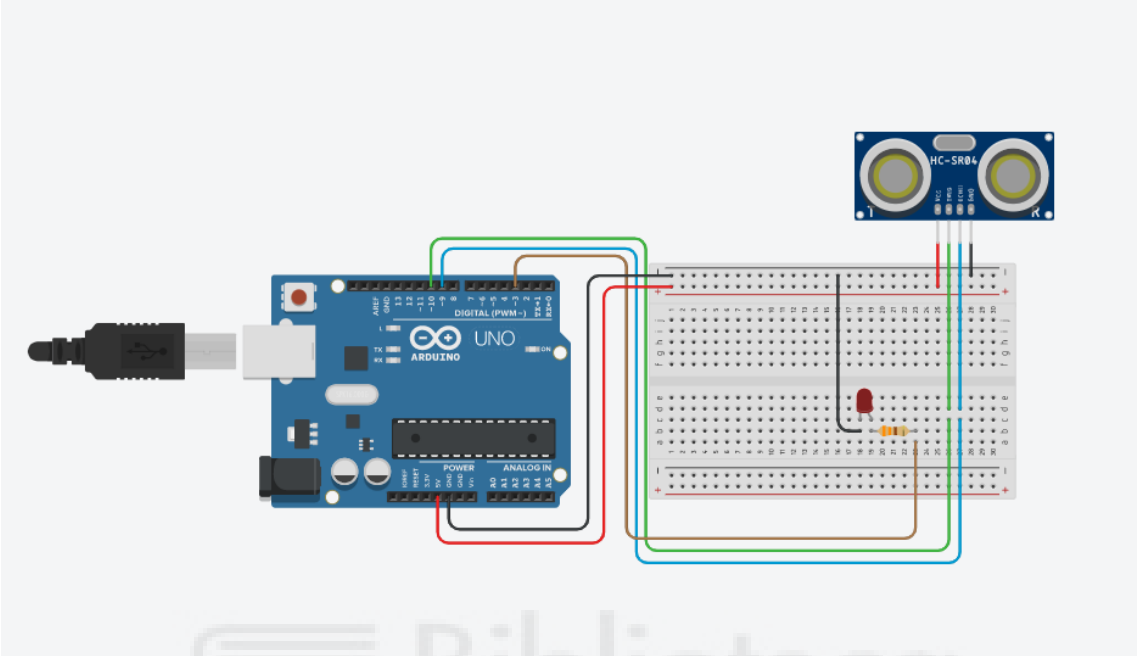


Imagen 17. Esquema Tinkercad Sensor Ultrasónico

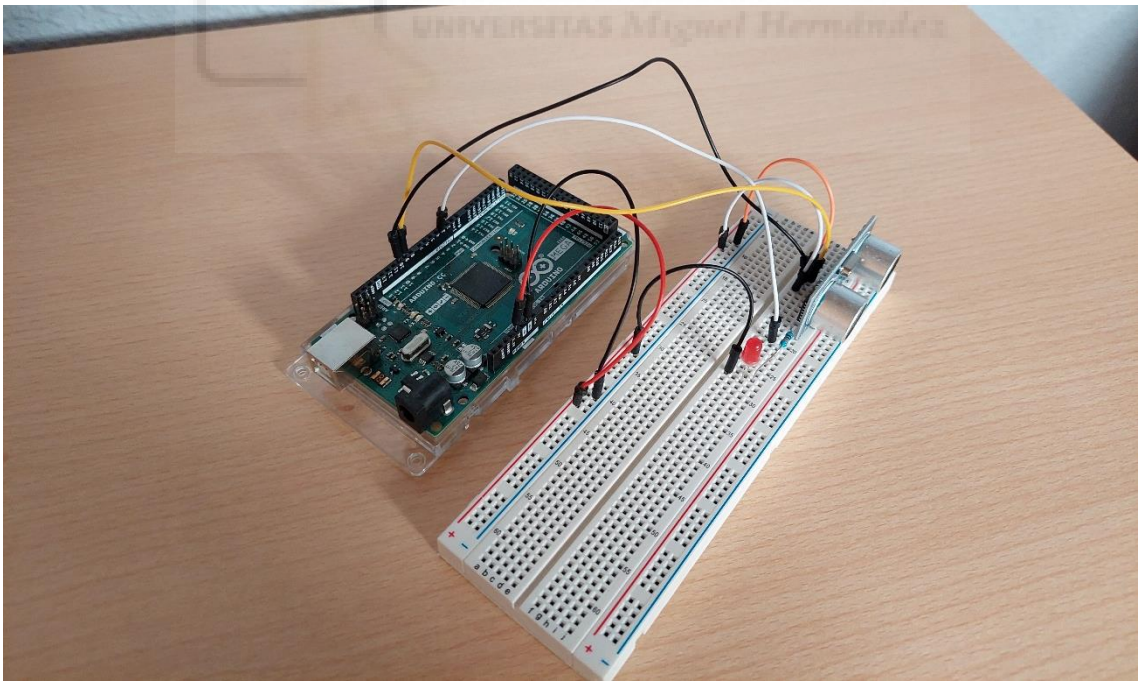


Imagen 18. Montaje Sensor Ultrasónico

Tal y como se muestra anteriormente en las imágenes, el sensor HC-RS04 estará conectado a los pines 9 y 10, estos estarán relacionados con las conexiones echo y trigger respectivamente. Además, el pin 3 activará el led

pasando antes por la resistencia de 330Ω. El funcionamiento queda explicado posteriormente junto al código.

El montaje presentado anteriormente será similar a los posteriores, **Prueba 2** y **Prueba 3**. Con la diferencia del código a implementar.

4.1.4 Código

Para la implementación del código en el sistema utilizaremos el entorno de Arduino IDE. Este entorno se empleará para el resto de las pruebas. Además, visualizaremos los resultados de una forma instantánea gracias al monitor serie.

En cuanto al código se compone de dos partes: `setup` y `loop`. La primera función `setup`, constituye el núcleo del programa, donde se localiza la configuración, la inicialización de variables y el envío de datos, entre otros. Estos comandos se ejecutarán una sola vez.

Por otro lado, se encuentra la función `loop`. Dicha función realiza en bucle todas las instrucciones que haya inmersas dentro de ella. Leyendo los datos percibidos por los elementos del circuito.

Una vez descrita la estructura genérica del código, a continuación, se mostrará el caso particular para la primera prueba realizada.

```
int TRIG = 10;     // trigger en pin 10
int ECO = 9;       // eco en pin 9
int LED = 3;       // LED en pin 3
int DURACION;
int DISTANCIA;
```

En estas mismas líneas, se observa la inicialización de las variables, las cuales se les asigna un pin dentro de la placa de Arduino. Además de que tipo de variable son, es te caso son todas de tipo entero (`int`).

```
void setup()
{
  pinMode(TRIG, OUTPUT);   // trigger como salida
  pinMode(ECO, INPUT);     // eco como entrada
  pinMode(LED, OUTPUT);    // LED como salida
  Serial.begin(9600);      // inicializacion de comunicacion serial a
9600 bps

}
```

Como se ha mencionado con anterioridad, se localizará la configuración de los distintos elementos del circuito al igual que la comunicación entre Arduino y el ordenador.

```
void loop(){

  digitalWrite(TRIG, HIGH);     // generacion del pulso a enviar del
sensor
  delay(1);
  digitalWrite(TRIG, LOW);

  DURACION = pulseIn(ECO, HIGH);  // alto en Eco

  DISTANCIA = DURACION / 58.4;    // distancia medida en centimetros
  Serial.println(DISTANCIA);      // envio de valor de distancia por
monitor serial
  delay(200);       // espera entre datos

  if (DISTANCIA <= 100 && DISTANCIA >= 0){ // si distancia entre 0 y
100 cms.
    digitalWrite(LED, HIGH);    // enciende LED
    delay(DISTANCIA * 10);      // espera proporcional a la distancia
    digitalWrite(LED, LOW);     // apaga LED
    }
}
```

Teniendo en cuenta el funcionamiento de la siguiente función, desglosaremos el bucle. Primeramente, activaremos el sensor emitiendo un pulso, este se enviará por el `TRIG`. Posteriormente, desactivaremos el `TRIG` y activaremos el `ECO`. Con ello se obtendrá la duración que tarda la onda en ser captada. Una vez adquirida se calculará la distancia a la cual se encuentra el objeto en centímetros. Para ello se utiliza la expresión: $\boldsymbol{DISTANCIA = DURACION / 58.4}$, donde tenemos que la $\boldsymbol{DURACION}$ es el tiempo que tarda la onda en ser emitida y captada por el sensor, además se deberá dividir entre 2 ya que dicho valor de duración es de ida y vuelta. Continuando con la ecuación encontramos en el denominador el valor $\boldsymbol{58.4}$, dicho valor alude a la conversión de la velocidad del sonido pasando de 343m/s a 29,2cm/µs, eso sí multiplicado el termino convertido por 2 de la operación de la $\boldsymbol{DURACION}$. Seguidamente se esperan 200 milisegundos para o bien volver a calcular el valor de distancia o bien entrar a la condición. Esta función tiene como utilizada encender el led cuando el sensor detecte algún objeto dentro del rango de 0 a 100 centímetros. Con ellos se conocerá si se encuentra próximo o no al sensor. La expresión mencionada anteriormente viene dada por la introducida en el **epígrafe 3.1.1**.

4.2 Prueba 2. Velocidad frontal del sensor HC-SR04

4.2.1 Objetivos

En esta segunda prueba se pretende obtener el cumplimiento de una serie de objetivos:

- Por medio de la utilización del sensor HC-SR04, obtener a qué velocidad se desplazan los objetos que atraviesan el rango del sensor.

- Programar un código en lenguaje C++ que nos permita adquirir dichos valores de velocidad y distancia, visualizándolos a través del Monitor Serial.
- Examinar la detección del objeto en el momento que se adentra en el rango de medición. Para posteriormente analizar dichos valores, se contrastarán para comprobar su exactitud.

4.2.2 Componentes

Los componentes de esta prueba han sido mencionados y descritos con anterioridad en el **epígrafe 4.1.2** Componentes de la **Prueba 1**.

4.2.3 Montaje

El montaje de la actual prueba es idéntico al de la **Prueba 1** como se ha mencionado en dicho epígrafe. La única excepción es que en este montaje no se ha utilizado el led.

4.2.4 Código

Dicho código tendrá la misma estructura descrita en la **Prueba 1**. Con la diferencia del contenido de este. En dicho programa obtendremos a qué velocidad se aproximan o se alejan los objetos al sensor.

Dicho esto, se expondrá el código descrito brevemente con anterioridad.

```
int trigPin = 10;
int echoPin = 9;
long duracion;
int distancia1=0;
int distancia2=0;
double Velocidad=0;
int distancia=0;
```

En dichas líneas se observa las variables del programa. Anteriormente vimos el tipo de dato `int`, en este caso tendremos `double` y `long` relacionados con las variables velocidad y duración respectivamente. Este tipo de datos aportarán mayor precisión a la hora de realizar el cálculo de velocidad y duración.

```
void loop() {
  distancia1 = ultrasonicRead();

  delay(300);

  distancia2 = ultrasonicRead();

  Velocidad = (distancia1 - distancia2)/0.3;

 if(distancia1<200 && distancia2<170 && distancia>70){

  Serial.print("Velocidad en cm/s:");
  Serial.println(Velocidad);
 }
}
```

```
float ultrasonicRead (){
digitalWrite(trigPin, LOW);
delayMicroseconds(2);

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

duracion = pulseIn(echoPin, HIGH);

distancia= duracion*0.034/2;

return distancia;
}
```


Previo al `void loop` se encuentra el `void setup`. Dado que la estructura es similar a la anteriormente descrita en la **Prueba 1**, tan solo se explicará la modificación realizada para no ser tan repetitivo.

En dichas líneas se observa como las variables `distancia1` y `distancia2` llaman a la función `ultrasonicRead()`, cuya tarea es proporcionar la distancia que obtiene el sensor ultrasónico. Una vez se obtienen los valores de dichas variables, se realizará el cálculo de la `Velocidad`. En este caso los valores de distancia serán restados y posteriormente divididos en 0,3 s, ya que entre la toma del primer valor de distancia hasta el siguiente han trascurrido 300 milisegundos, un valor más que asumible para obtener un valor medio confiable para la obtención de la velocidad. A continuación, se observa la función condicional, cuyas restricciones impiden que se den valores atípicos que puedan darnos un valor fiable al imprimir por pantalla el valor de la velocidad.

4.3 Prueba 3. Velocidad perpendicular del sensor HC-SR04

4.3.1 Objetivos

En esta tercera prueba se pretende obtener el cumplimiento de una serie de objetivos:

- En esta prueba se pretende conseguir unos resultados similares a los de la **Prueba 2**. Debido a que ambas tienen la misma función, calcular la velocidad a la cual se desplazan los objetos que entren dentro de su rango de acción.
- En esta ocasión también se llevará a la práctica el uso del sensor HC-SR04. En cambio, esta vez se realizarán las mediciones de forma perpendicular al movimiento del objeto. Dado que el sensor posee un ángulo de acción se tendrá en cuanta a la hora de realizar los cálculos pertinentes para obtener una aproximación de la velocidad a la que transcurren los objetos.
- Una vez obtenido dichos valores, se realizarán los análisis oportunos para verificar si los datos obtenidos son del todo fiables.

4.3.2 Componentes

Los componentes de la actual prueba han sido mencionados y descritos con anterioridad en el **epígrafe 4.1.2** Componentes de la **Prueba 1**.

4.3.3 Montaje

El montaje actual prueba es idéntica al de la **Prueba 2** al igual que el de la **Prueba 1**. Dicho esto, al igual que en la **Prueba 1** la luz led estará en este montaje.

4.3.4 Código

El código que se mostrará a continuación tendrá la misma estructura que los anteriores. En este caso el objetivo del programa será similar al de la **Prueba 2**. Debido a que ambos tienen la finalidad de calcular la velocidad. Para el caso actual se tendrá en cuenta el ángulo de apertura del sensor indicado por el fabricante 15 grados. Como se puede observar en la siguiente imagen:

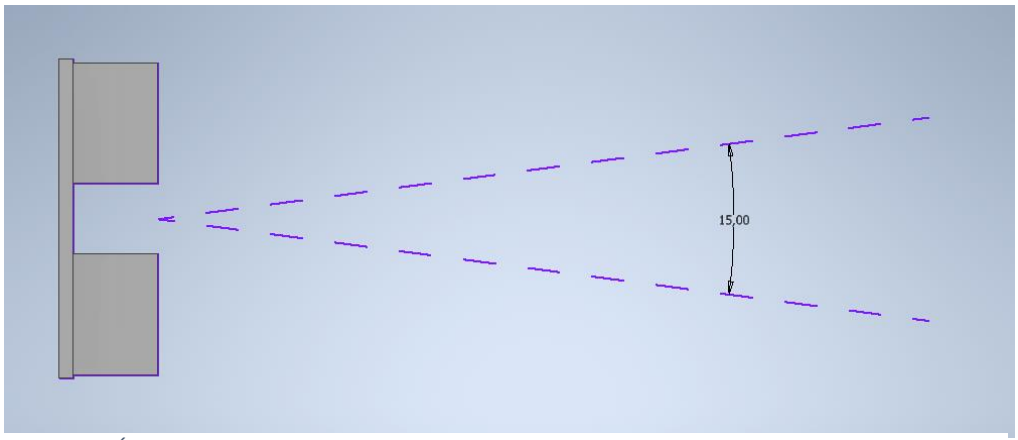


Imagen 19. Ángulo de apertura del sensor HC-SR04

Descrito como se realizarán las mediciones en dicha prueba, se observan las variables a utilizar. Pero antes de ello se mostrará una representación de los parámetros utilizados por lo que será más simple la comprensión del código.

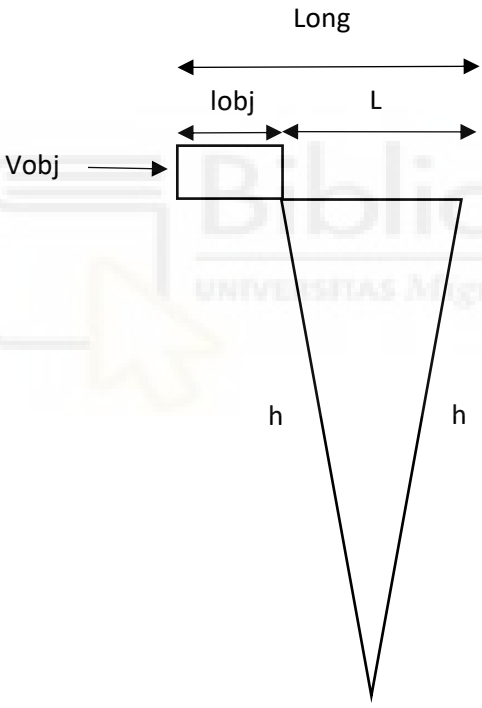


Imagen 20. Representación de las variables de la prueba 3

En dicha imagen se observa un triángulo que indica la amplitud de captación de datos del sensor, mientras que el rectángulo es una representación de un objeto cualquiera. Además de estas representaciones, se indican las distancias tanto del objeto, la longitud total recorrida y la longitud que capta el sensor. Estas son lobj=63.4cm, L=165; Long=lobj+L y h=117. Se considera además el ángulo de apertura del sensor el cual es de 15 grados.

```
int trigPin = 10;
int echoPin = 9;
int LED = 3;
float distancia;
long duracion;
float t;
float t1;
float t2;
float tiempo;
float Vobj; //cm/s
const float L = 165; //cm
const int h = 117; // cm
const float lobj = 63.4; //cm
float Long;
int aux = 0;
int sol = 0;
```

Como se observan en las anteriores líneas de código, las variables tienen ciertos valores de longitud. Esto se debe a que las mediciones están hechas a la distancia de 1 metro. Quiere decir que el sensor estará a una separación de 1 metro en dirección perpendicular al paso de los objetos como se muestra en la **Imagen 16**. Dicho sensor posee un rango de medición de 2 cm a 400 cm por lo que el valor dado para realizar la prueba se encuentra dentro del intervalo del sensor.

Además de las variables temporales, las del sensor y el led. Habrá dos variables auxiliares que ayudarán al desarrollo del código.

```
void loop() {
  t=millis();

  digitalWrite(trigPin, HIGH);
  delay(1);
  digitalWrite(trigPin, LOW);

  duracion = pulseIn(echoPin, HIGH);
  distancia = duracion / 58.4;
```

Previamente se iniciará con la introducción del `void setup` pero no será necesaria su mención dado que en las anteriores pruebas se ha descrito.

Por otro lado, como se lee en las primeras líneas del bucle se iniciará un contador de tiempo, además de comenzar el funcionamiento del sensor.

```
if(aux==0 && distancia>=80 && distancia<=h){
  digitalWrite(LED, HIGH);
  t1=t;
  aux=1;
}
if(aux==1 && distancia>h+30){
  digitalWrite(LED, LOW);
  t2=t;
  aux=0;
  sol=1;
}
if(sol==1 && aux==0){
  Long=L+lobj;
  tiempo=t2-t1;
  Vobj=Long/(tiempo/1000);
  Serial.print("Velocidad del objeto en cm/s :");
  Serial.println(Vobj);
  sol=0;
  }
}
```

A continuación, se muestra el grueso del programa. En función de si transcurre o no el objeto por el rango del sensor (el cual debe de pasar a 1 metro de separación), se activarán las condiciones del programa.

En primer lugar, una vez el objeto entre dentro del rango, se activará la primera condición. Con ello guardaremos el tiempo en `t1` y la variable `aux` pasará de 0 a 1. Además de encenderse la luz LED.

En segundo lugar, en el momento de la salida del objeto del rango del sensor, el LED se apagará, la variable `aux` volverá a su estado inicial, en su lugar la variable `sol` pasará de 0 a 1 y se captará el tiempo de salida del objeto.

Finalmente se ejecutará la última operación, la cual mostrará por pantalla la velocidad a la que se desplaza el objeto.

Y, ¿cómo se obtiene la velocidad? Partiendo de la variable `t`, activará el contador al inicio del `void loop`. Por lo tanto, en el momento de entrar a la primera y segunda condición dicho valor de `t` pasa a ser guardado en las variables `t1` y `t2`, dichos valores hacen referencia al momento de entrada y salida del objeto respectivamente. Dando paso a la tercera condición se pretende explicar que expresiones se han utilizado para obtener la velocidad. Primeramente $\boldsymbol{Long = L + lobj}$ para esta expresión lo que se pretende es calcular la longitud real qué ha recorrido el objeto, esto se debe a que el sensor capta en qué momento entra y sale el objeto por lo que no solo recorre una distancia `L`, sino que también habrá que tener en cuenta lo que mide el objeto `lobj`. Además $\boldsymbol{tiempo = t2 - t1}$ nos proporciona el tiempo que transcurre entre la entrada y la salida. Por tanto, gracias a la expresión: $\boldsymbol{Vobj = Long/(tiempo/1000)}$ obtendremos el valor de velocidad del objeto en las unidades de cm/s.

Para una mayor comprensión de la prueba se realiza a continuación una secuencia de imágenes.

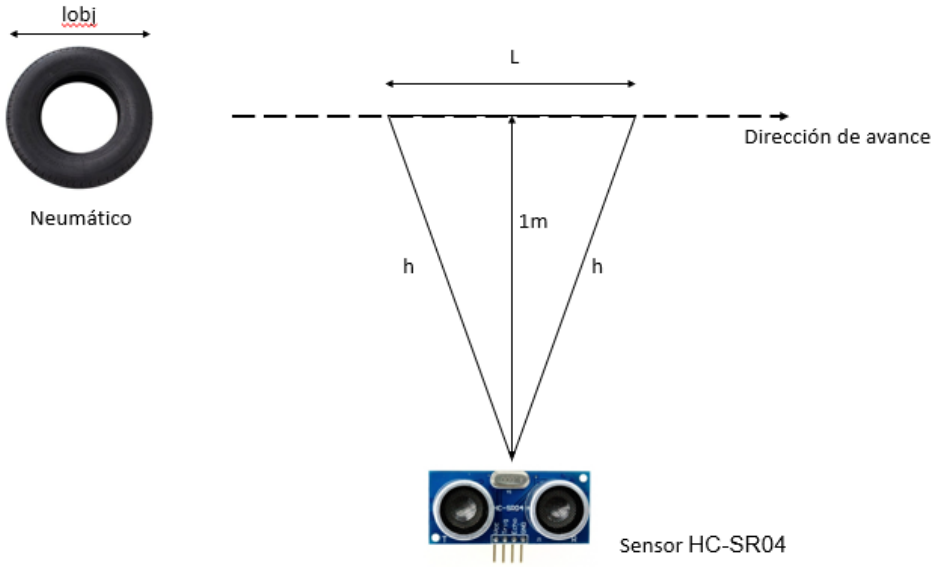


Imagen 21. Descripción Prueba 3 Paso 1

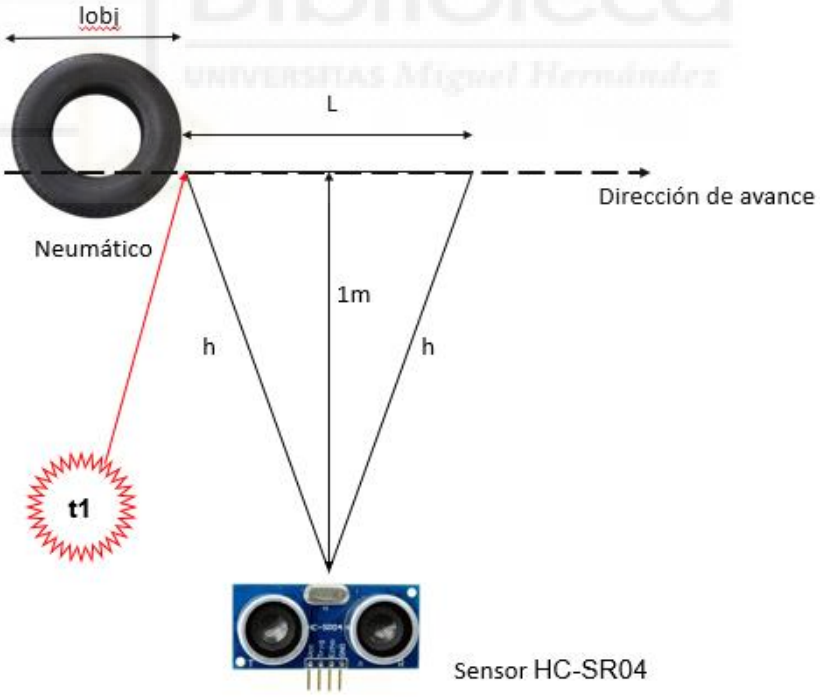


Imagen 22. Descripción Prueba 3 Paso 2

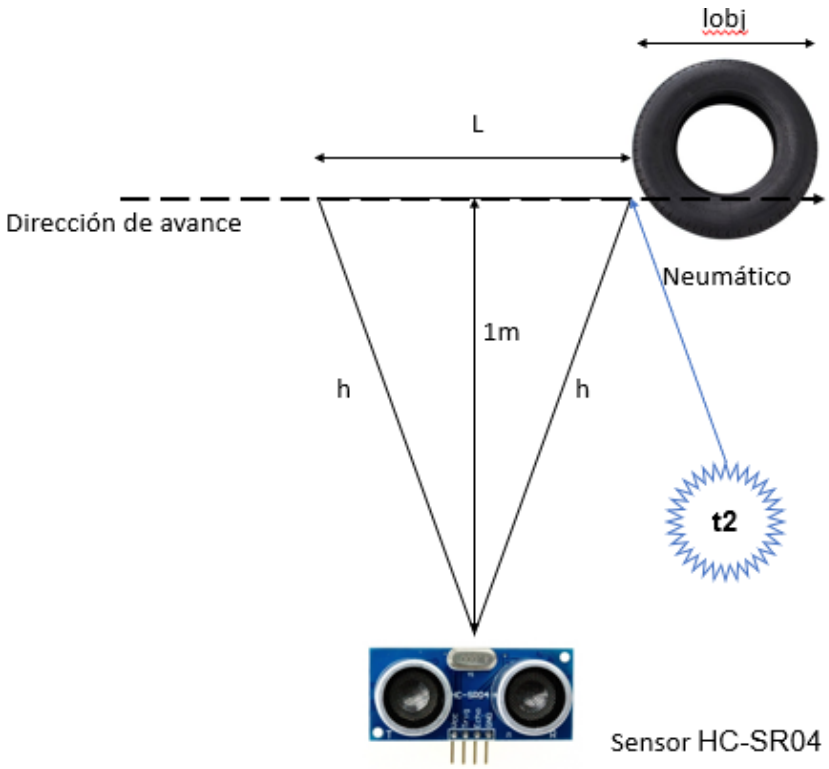


Imagen 23.. Descripción Prueba 3 Paso 3

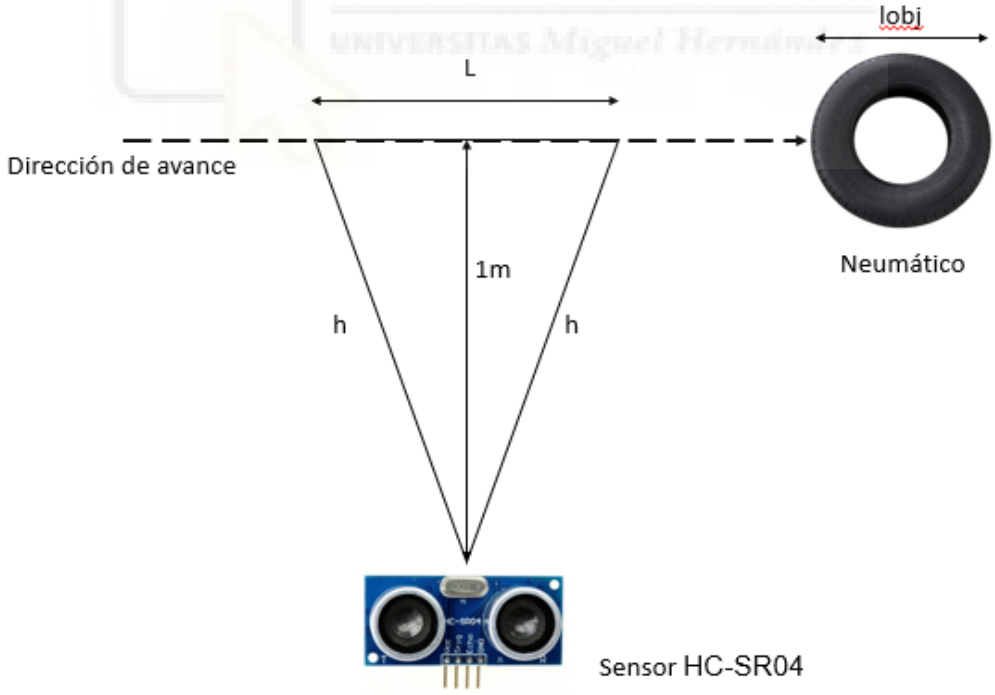


Imagen 24. Descripción Prueba 3 Paso 4

Cabe destacar que dicha prueba se realizó con una apertura del sensor de 15 grados como muestra en la ficha técnica dada por el fabricante.

Pero se detectó que dicho ángulo no coincidía con el del fabricante y en la web [12] se puede observar que la apertura real que se obtiene en este tipo de sensores puede llegar a los 30 grados e incluso 40.

Esto se debe a la directividad del sensor que como se muestra en la siguiente imagen se destaca un ángulo mayor al del fabricante. Además, se añaden en la imagen las condiciones descritas en el código a la hora de adquirir los datos por parte del sensor:

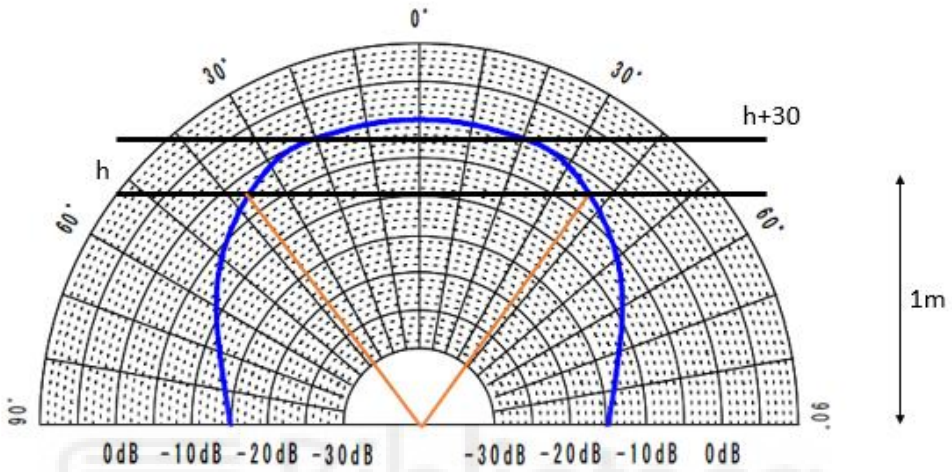


Imagen 25. Directividad sensor HC SR-04

4.4 Prueba 4. Rango lidar

4.4.1 Objetivos

En esta cuarta prueba se pretende obtener el cumplimiento de una serie de objetivos:

- Mediante la utilización del sensor láser Lidar Lite V3, obtener a qué distancia se encuentran los objetos ubicados dentro de su rango.
- Conocer cómo funciona el sensor anteriormente mencionado.
- Programar un código en lenguaje C++ que nos permita adquirir dichos valores de distancia, visualizarlos a través del Monitor Serial.
- Examinar la detección del objeto en el momento que se adentra en su rango de medición dado que posteriormente dicha información será útil para la realización de las consecutivas pruebas.

4.4.2 Componentes

Para dicha prueba serán necesarios ciertos componentes que se mencionarán a continuación. Además, hay que añadir que estos serán utilizados en las posteriores pruebas que vendrán (**Prueba 5 y 6**).

- **Sensor Lidar Lite V3**

Se trata de un sensor láser que permite calcular distancias. Posee una amplia gama de opciones para su uso, que permiten al usuario adaptar la precisión, el tiempo de medición y la distancia de funcionamiento del sensor. Dicho sensor tiene dos modos de comunicación I2C y PWM, su rango de funcionamiento puede ir dentro de los 5 cm a los 4 m. Además de no consumir una gran cantidad de energía y un tamaño compacto.

Imagen 26. Componente Lidar Lite V3

- **Condensador 1000µF**

Elemento eléctrico pasivo cuya función es la de almacenar energía para posteriormente ser liberada rápidamente. Está compuesto por dos partes metálicas que a su vez están separadas por un material dieléctrico o vacío. En este caso se usa para filtrar posibles espúreos en las señales eléctricas del circuito.

Imagen 27. Componente Condensador

Además, se necesitan los siguientes componentes:

- **Arduino Mega 2560**
- **Placa de pruebas o protoboard**
- **Cables dupont macho-macho**

Dichos elementos han sido descritos con anterioridad en el **epígrafe 4.1.2**.

4.4.3 Montaje

En cuanto al montaje se ha representado un esquema del circuito tanto en la plataforma online Tinkercad como de forma física.

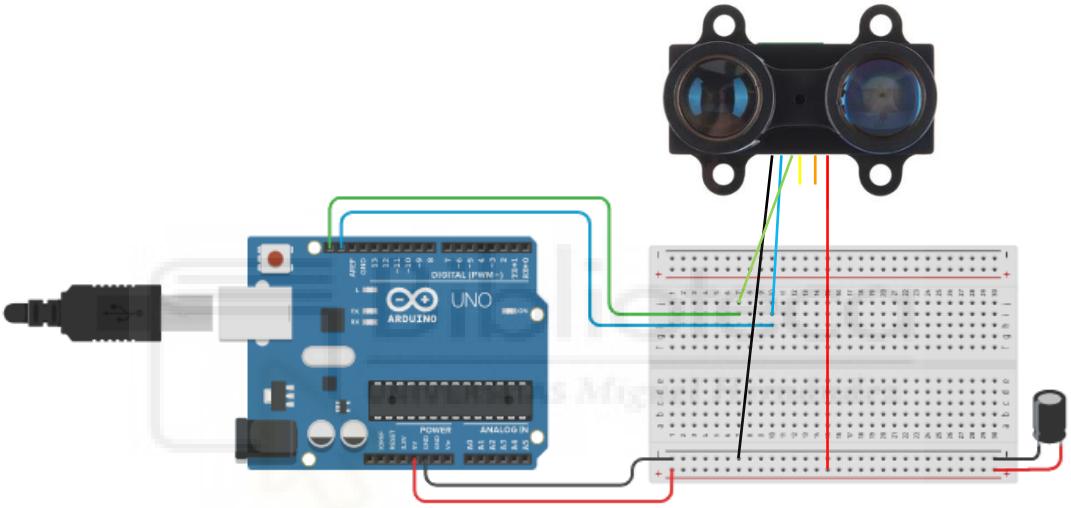


Imagen 29. Esquema Tinkercad Lidar

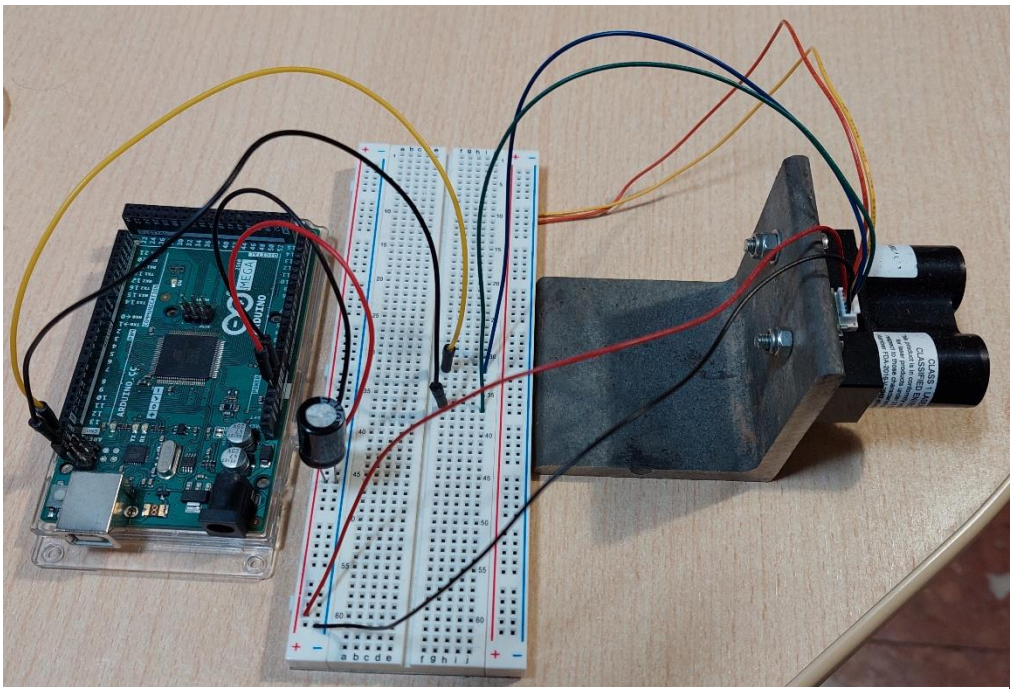


Imagen 28. Montaje Lidar

Como se ha mostrado en las anteriores imágenes el sensor laser está conectado por los pines SDA y SDL, además de los pines a toma de corriente y tierra. Se puede observar que hay dos conexiones del sensor. Estas están en aire, esto se debe a la configuración del sensor que se utilizará en las pruebas. La configuración es la I2C. También se añade un condensador de 1000 microfaradios.

El montaje presentado anteriormente será similar a los posteriores, **Prueba 5** y **Prueba 6**. Con la diferencia del código a implementar.

4.4.4 Código

En el presente código se busca el mismo objetivo que se mostró en la **Prueba 1**. Tan solo diferenciar que ahora nuestro elemento de medición será el sensor láser Lidar, además del código a implementar.

```
#include <Wire.h>
#include <LIDARLite.h>

LIDARLite myLidarLite;
```

Como se muestra previamente, se incluirán librearías acordes para el uso del sensor. Una de ellas es `Wire.h`, dicha librería facilita la comunicación entre Arduino y los dispositivos por I2C. Estos dispositivos utilizar las conexiones SDA y SCL que pertenecen a el envío de datos y al tiempo respectivamente.

Se incluirá la librería `LIDARLite.h` que permitirá la correcta utilización con el sensor a utilizar.

Además, se declara en la última línea el objeto `LIDARLite` para el control de dicho sensor.

```
void setup(){
 Serial.begin(115200);

 myLidarLite.begin(0, true);

 myLidarLite.configure(0);
}
```

Para el siguiente bloque de código se configura el sensor, ya que posee múltiples combinaciones. Donde aparece `myLidarLite.begin(0, true)` establecerá la configuración por defecto del modo I2C. En la siguiente línea se podrá modificar el modo de funcionamiento del sensor. Dichos valores junto con sus especificaciones quedan reflejados en el datasheet del sensor.

```
void loop(){
 Serial.println(myLidarLite.distance());

 for(int i = 0; i < 99; i++){
   Serial.println(myLidarLite.distance(false));
  }
}
```

Para finalizar el programa se mostrará por pantalla a que distancia se encuentran los objetos que transitan dentro del rango. Tomará una medida con corrección cada 100 mediciones, debido a las posibles variaciones de la luz ambiente. Dicho valor de distancia se obtiene gracias a la expresión descrita en el **epígrafe 3.1.2.**

4.5 Prueba 5. Velocidad frontal lidar

4.5.1 Objetivos

En esta quinta prueba se pretende obtener el cumplimiento de una serie de objetivos:

- Por medio de la utilización del sensor láser Lidar Lite v3, obtener a qué velocidad se desplazan los objetos que se aproximan al sensor.
- Programar un código en lenguaje C++ que nos permita adquirir dichos valores de velocidad y distancia, visualizándolos a través del Monitor Serial.
- Examinar la detección del objeto en el momento que se adentra en el rango de medición. Para posteriormente analizar dichos valores, se contrastarán para comprobar su exactitud.

4.5.2 Componentes

Mencionado en el **epígrafe 4.4.2**, se alude a dicha prueba ya que los componentes a utilizar son los mismo que se mencionaron de dicho epígrafe. Tan solo variará el objetivo de dicha prueba.

4.5.3 Montaje

Asimismo, como se cita en el epígrafe anterior, los componentes a utilizar serán los mismo y por tanto el montaje será similar.

4.5.4 Código

A continuación, se mostrará el código de la actual prueba. Para esta ocasión la base del programa será igual a la del **epígrafe 4.4.4**.

```
#include <Wire.h>
#include <LIDARLite.h>

LIDARLite myLidarLite;
int dist1;
int dist2;
float vel;
```

Como se puede observar se utilizan las mismas librearías, tan solo se añaden las variables de distancia y la de velocidad.

```
void setup(){

  Serial.begin(115200);
  myLidarLite.begin(0, true);
  myLidarLite.configure(0);
}
```

Al igual que en la **Prueba 4** el núcleo del programa será igual debido a que seguirá con la misma configuración y además del cálculo de la distancia.

```
void loop(){

    for(int i = 0; i < 99; i++){
    dist1 = myLidarLite.distance(false);
    delay(500);
    dist2 = myLidarLite.distance(false);

    vel=(dist2-dist1)/0.5;
    if(vel>=0){
    Serial.print(vel);
    Serial.println("cm/s");

    }else{
    vel=-vel;
    Serial.print(vel);
    Serial.println("cm/s");
    }
    }
    }
```

El principal cambio y el objetivo del código se basa en el cálculo de la velocidad a medida que se aproxima o se aleja el objeto del sensor de manera frontal. El cálculo se realizará por medio de la expresión: $\boldsymbol{vel} = (\boldsymbol{dist2} - \boldsymbol{dist1})/\mathbf{0.5}$, donde las variables `dist1` y `dist2` proporcionan la distancia a la cual se encuentra el objeto y todo ello dividido entre 0.5 segundos. Dicho valor de tiempo será necesario para la obtención de un valor medio aceptable.

Además del cálculo se tiene una función condicional para mostrar por pantalla dicho valor de velocidad tanto si se aproxima como si se aleja con signo positivo.

4.6 Prueba 6. Velocidad perpendicular lidar

4.6.1 Objetivos

En esta sexta prueba se pretende obtener el cumplimiento de una serie de objetivos:

- En esta prueba se pretende conseguir unos resultados similares a los de la **Prueba 5**. Debido a que ambas tienen la misma función, calcular la velocidad a la cual se desplazan los objetos que entren dentro de su rango de acción.
- En esta ocasión también se llevará a la práctica el uso del sensor láser Lidar Lite V3. En cambio, esta vez se realizarán las mediciones de forma perpendicular al movimiento del objeto, como también se hizo en la **Prueba 3**.
- Una vez obtenido dichos valores, se realizarán los análisis oportunos para verificar si los datos obtenidos son del todo fiables.

4.6.2 Componentes

Los componentes que aparecen en dicha prueba serán los mismo que en las pasadas **Pruebas 4 y 5**. Al igual que ocurre en la **Prueba 5** tan solo cambia el código.

4.6.3 Montaje

Como se mostró en el **epígrafe 4.4.3** dicho modelo será similar al de la prueba actual. Tan solo cambia la orientación a la cual realizará las mediciones.

4.6.4 Código

De acuerdo con los objetivos descritos en dicha prueba, esta tendrá la misma función a la descrita en la **Prueba 5**. En este caso se posicionará el sensor perpendicular al movimiento del objeto como se realizó en la **Prueba 3**, además de posicionar el sensor a la misma distancia del avance del objeto a 1 metro. Hay que comentar que, para el actual sensor, el rango de medición es mucho más amplio llegando hasta los 40 metros. En el **epígrafe 4.3.4** se puede

observar cómo se realiza la prueba para el sensor ultrasónico, en cambio, para el sensor láser la apertura no será la misma como refleja la siguiente imagen:

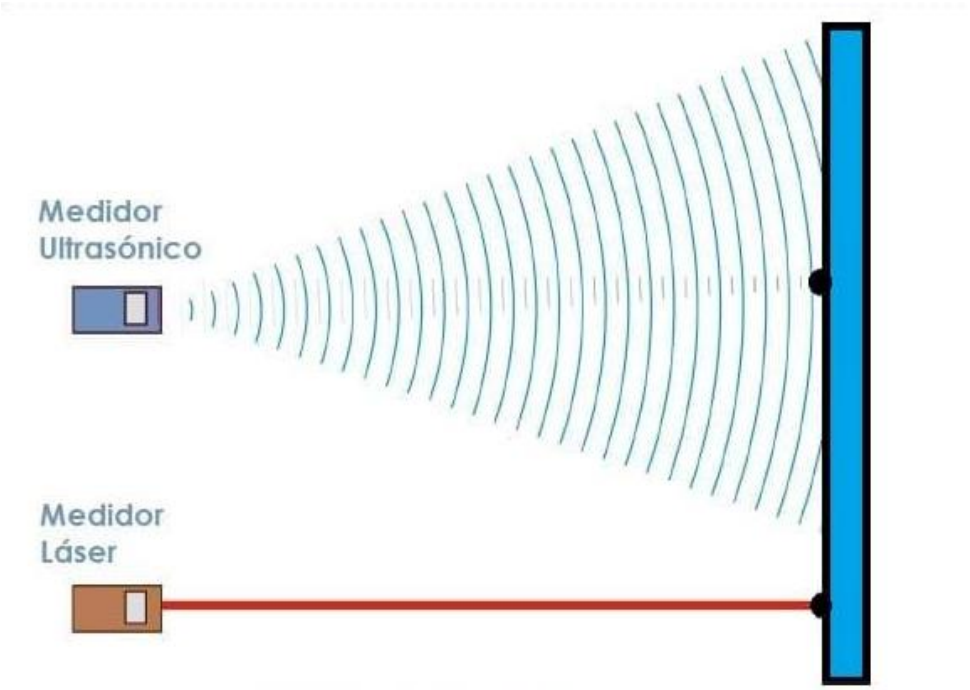


Imagen 30. Comparativo ángulo de acción sensor Ultrasónico vs Láser

Una vez entendido como se realiza la prueba se expondrá el código, para el cual la base de este será igual a los antes descritos en las **Pruebas 4 y 5**.

```
#include <Wire.h>
#include <LIDARLite.h>

LIDARLite myLidarLite;
float Vobj;
float t;
float t1;
float t2;
float tiempo;
float lobj=63;
int aux = 0;
int sol = 0;
```

Se observa que en este caso tenemos las mismas librerías, sin embargo, aparecen diversas variables de tiempo, auxiliares al igual que la longitud del objeto que transcurre en dirección perpendicular al sensor como aparecen en el **epígrafe 4.3.4**.

```
void setup() {
  Serial.begin(115200);
  myLidarLite.begin(0, true);
  myLidarLite.configure(0);
}
```

Al igual que en las anteriores pruebas con el sensor láser el núcleo del programa es igual.

```
void loop() {
  t=millis();
  for(int i = 0; i < 99; i++) {
  if(aux==0 && myLidarLite.distance(false)>=100 &&
myLidarLite.distance(false)<200){
  t1=t;
  aux=1;
}
  if(aux==1 && myLidarLite.distance(false)>220){
  t2=t;
  aux=0;
  sol=1;
}
  if(sol==1 && aux==0){
  tiempo=t2-t1;
  Vobj=lobj/(tiempo/1000);
  Serial.print("Velocidad del objeto en cm/s:");
  Serial.println(Vobj);
  sol=0;
}
}
}
```

Al comienzo del bucle se iniciará el contador `t=millis()` que permitirá posteriormente dar el tiempo en el momento que se detecta el objeto y a su salida del rango, esto ocurre de la misma manera en la **Prueba 3**. Se han establecido unos límites como se muestra en las condiciones para delimitar el rango de actuación del sensor, debido a que posicionaremos el sensor a la misma distancia que en la descrita en el **epígrafe 4.3.4**. Por lo tanto, cuando el objeto entre dentro de la primera condición se tomará la primera toma de tiempo `t1` y una vez salga y pase a la segunda condición se obtendrá esa segunda toma de tiempo `t2`. Hay que añadir que dicho sensor no posee la apertura angular del HC SR-04 por lo que la implementación del código y la toma de datos es algo más sencilla dado que no se requieren cálculos trigonométricos y por tanto un mayor número de variables a tener en consideración.

Una vez haya pasado el objeto saldrá por pantalla a qué velocidad ha transcurrido. La cual se calculará gracias a la expresión: $\boldsymbol{Vobj = lobj/(tiempo/1000)}$, se encuentra en el numerador el espacio recorrido en este caso es la longitud del objeto y en el denominador el tiempo entre la entrada y la salida en segundos.

5 RESULTADOS

Antes de comenzar con los resultados de las pruebas se han de mencionar dos medios más de medición, los cuales han sido utilizados con la finalidad de contrastar los datos con los obtenidos por los sensor ultrasónico y láser, por lo que asumiremos dichos medios (un par de fotocélulas y una aplicación móvil (Velocímetro)) como medidas de velocidad reales. Esto se debe a la no utilización de un sistema GPS ya que las pruebas han sido realizadas en interior. Para futuros trabajos se daría el salto al exterior utilizando por tanto elementos como el velocímetro de un vehículo o el propio GPS.

Dado que la finalidad del estudio será la comparativa del sensor HC-SR04 y Lidar Lite V3 y se deberá tener un primer valor "real" de la velocidad a la cual se mueve el objeto.

Comenzando por la obtención de datos por las fotocélulas y la aplicación móvil, se decidió no utilizar finalmente dicha aplicación debido a que a la hora de realizar el seguimiento del objeto se debe seguir con el dedo por la pantalla del móvil. Esta práctica no es muy robusta para la obtención de datos, puesto que no se podía seguir de formar constante al objeto. Su funcionamiento trata de estimar la velocidad del objeto mediante el procesamiento de imágenes.

Seguidamente se puede apreciar cómo es la interfaz de la aplicación:

Imagen 31. Aplicación móvil

Se decidió por tanto utilizar como medio para contrastar los resultados de los sensores, los datos obtenidos por las fotocélulas. Se puede observar a continuación como estaban posicionadas.

Imagen 32. Fotocélulas

En primera instancia para la realización de las pruebas se planteó inicialmente impulsar un neumático a lo largo de una superficie plana. Así pues, los sensores y las fotocélulas pudieran capturar el tiempo transcurrido en recorrer cierta distancia y por tanto a qué velocidad se desplazaba. No obstante, debido a la necesidad de tener unos valores más constantes, se decidió utilizar una plataforma de lanzamiento para realizar los ensayos. Debido a que estimar la velocidad mediante dinámica y cinemática resulta complejo por el gran número de variables externas, se decidió obtener la velocidad a partir de las fotocélulas comenzando el movimiento desde la plataforma de lanzamiento.

Se puede observar en las siguientes imágenes a qué dos alturas se iniciaba el movimiento del neumático además de un esquema global. Posteriormente se tomaron los datos pertinentes con las fotocélulas y se obtuvo una velocidad media a la cual se desplaza.

Dicha velocidad se calculó mediante la expresión: $\boldsymbol{Velocidad} = \boldsymbol{Espacio}/\boldsymbol{Tiempo}$. Donde el $\boldsymbol{Espacio}$ comprende la distancia entre las fotocélulas y el $\boldsymbol{Tiempo}$ se obtiene por la diferencia temporal entre la fotocélula 2 y la 1 como se muestra en el esquema.

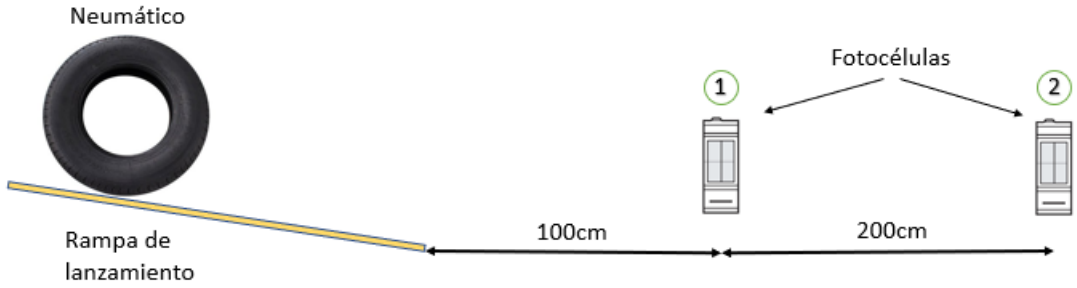


Imagen 33. Montaje Fotocélulas

Imagen 34. Salida de la rampa alta

Imagen 35. Salida de la rampa baja

Para tener unas medidas sobre la plataforma se sabe que esta mide 150 centímetros y uno de sus extremos, se encuentra elevado unos 16 centímetros respecto al otro. Por tanto, se obtiene un ángulo aproximado de 6 grados.

Para la obtención de los datos de las fotocélulas fue necesaria la utilización del software Test Xpress 7A para obtener los valores de las fotocélulas que posteriormente dichos datos fueron exportados y tratados en una hoja Excel para hallar el tiempo transcurrido, y posteriormente dicho valor de velocidad. Dado que los valores que proporciona el programa son los instantes en los que se corta la señal de las fotocélulas, se entiende que es el momento donde el objeto se sitúa entre las fotocélulas, y por tanto se activan los contadores de tiempo.

Dicho software lo que indica es el momento justo en el que algo corta la señal de una de las fotocélulas. Para este proyecto se han utilizado dos para marcar cuando entra y cuando sale el neumático del recorrido.

A continuación, se muestra la interfaz del programa y en qué momentos se detecta al objeto, puesto que una vez es detectado la señal desciende a 0 debido al corte de la señal.

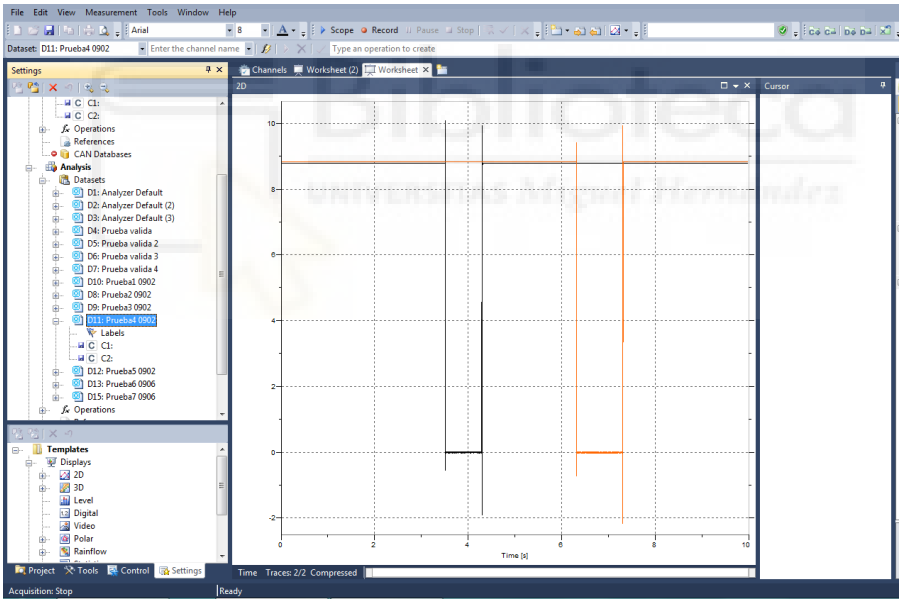


Imagen 36. Software empleado para las fotocélulas

Una vez extraídos los datos del programa se procede a su evaluación. Para ello se hará uso de una plantilla creada con el fin de obtener una serie de valores que permitirán obtener en qué momentos entró y salió el neumático del recorrido.

A continuación, se muestra dicha plantilla donde se localiza en la primera columna un contador de tiempo. Este contador no es más que la frecuencia de captación de los datos por parte del programa, la frecuencia es de 51200 Hz. Seguidamente se observan 4 columnas divididas en C1 y C2. Donde C1 y C2 son los canales que se utilizaron para los ensayos. La primera columna de dichos canales se tendrá el tiempo desde el inicio de la grabación de los datos. En la

segunda se observa la tensión que emite la fotocélula, una vez se interrumpa su señal interponiéndose en medio de ella, la tensión cae a 0.

Estas primeras columnas son los datos exportados del programa, ahora en las columnas F y G se formula una condición que cambiará cuando la tensión de las columnas C y E pasen de su estado alto hasta su estado más bajo. Manteniéndose en 0 y en el momento de cambio dará el dato de tiempo en el cual se atraviesa la fotocélula.

Ese valor de tiempo se reflejará en las celdas H2 e I2 gracias a la fórmula que aparece en la fila 9. Con ello tan solo queda realizar una simple ecuación, $\boldsymbol{Velocidad = Distancia / (Tiempo\ C2 - Tiempo\ C1)}$ con lo que se halla la velocidad del objeto. Se refleja en la celda L2.

Hay que mencionar la distancia de separación de las fotocélulas, ya que será la que recorra el neumático. Esta es de 200 cm, la que aparece en la celda L2.

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Tiempo (s)	C1: Time[s]	C1: []	C2: Time[s]	C2: []			Tiempo C1	Tiempo C2				
2	1,95313E-05	0	9,15	0	9,28			2,4553516	4,4822852		Distancia (cm)	200	
3	3,90625E-05	0	9,15	0	9,28	0	0				Tiempo (s)	2,026934	
4	5,85938E-05	0	9,15	0	9,28	0	0				Velocidad (cm/s)	98,67121	
5	0,000078125	0	9,15	0	9,28	0	0						
6	9,76563E-05	0	9,15	0	9,28	0	0						
7	0,000117188	0	9,15	0	9,28	0	0						
8	0,000136719	0	9,15	0	9,28	0	0						
9	0,00015625	0	9,15	0	9,28	0	0	=INDICE(A2:A11,COINCIDIR(VERDADERO,A2:A11>0,0))					
10	0,000175781	0	9,16	0	9,28	0	0						
11	0,000195313	0	9,15	0	9,28	0	0						
12	0,000214844	0	9,16	0	9,28	0	0						
13	0,000234375	0	9,15	0	9,28	0	0						
14	0,000253906	0	9,15	0	9,28	0	0						
15	0,000273438	0	9,15	0	9,28	0	0						
16	0,000292969	0	9,15	0	9,28	0	0						
17	0,0003125	0	9,15	0	9,28	0	0						
18	0,000332031	0	9,15	0	9,28	0	0						
19	0,000351563	0	9,15	0	9,28	0	0						
20	0,000371094	0	9,15	0	9,28	0	0			Inicio Rampa	V media (cm/s)		
21	0,000390625	0	9,15	0	9,28	0	0			Alta	136,465		
22	0,000410156	0	9,15	0	9,28	0	0			Media	97,79		

Imagen 37. Plantilla de resultados ensayos

Gracias a la plantilla se puede obtener los valores que se muestran en las celdas K21 y K22. Estos datos servirán de contraste a los que se obtengan de los sensores.

Se obtuvo por tanto que el neumático viaja a una velocidad de 136,465 cm/s cuando su movimiento se inicia en la parte alta de la rampa. Por otro lado, al iniciar su movimiento desde una posición media de la rampa, alcanzará una velocidad de 97,79 cm/s.

Tanto la velocidad al comienzo del punto alto como del punto medio de la rampa han sido halladas a raíz del valor medio de los lanzamientos realizados desde dichos puntos.

Con todo ello se pasará a cotejar los resultados obtenidos por los sensores.

5.1 Resultados Velocidad Frontal HC-SR04 y Lidar Lite V3

En este epígrafe se realizará una comparativa entre los resultados obtenidos por ambos sensores a la hora de medir la velocidad del neumático a medida que se acerca frontalmente a los sensores. Estos resultados hacen alusión a las **Pruebas 2 y 5**.

Primeramente, se expondrán los resultados de los sensores cuando el neumático inicia su descenso desde lo alto de la rampa de lanzamiento.

VELOCIDAD FRONTAL INICIO ALTO			
Lidar Lite V3		**HC-SR04**	
Velocidad Sensor (cm/s)	**Error**	**Velocidad Sensor (cm/s)**	**Error**
138	-1,12%	132	3,27%
136	0,34%	134	1,81%
134	1,81%	130	4,74%
138	-1,12%	134	1,81%
136	0,34%	136	0,34%
136	0,34%	134	1,81%
136	0,34%	132	3,27%
138	-1,12%	132	3,27%
136	0,34%	130	4,74%
138	-1,12%	132	3,27%
Velocidad Media	**Error Velocidad Media**	**Velocidad Media**	**Error Velocidad Media**
136,6	-0,10%	132,6	2,83%
Varianza	1,82	**Varianza**	3,60

Tabla 1. Datos de Sensores para Ensayo Frontal Inicio Alto

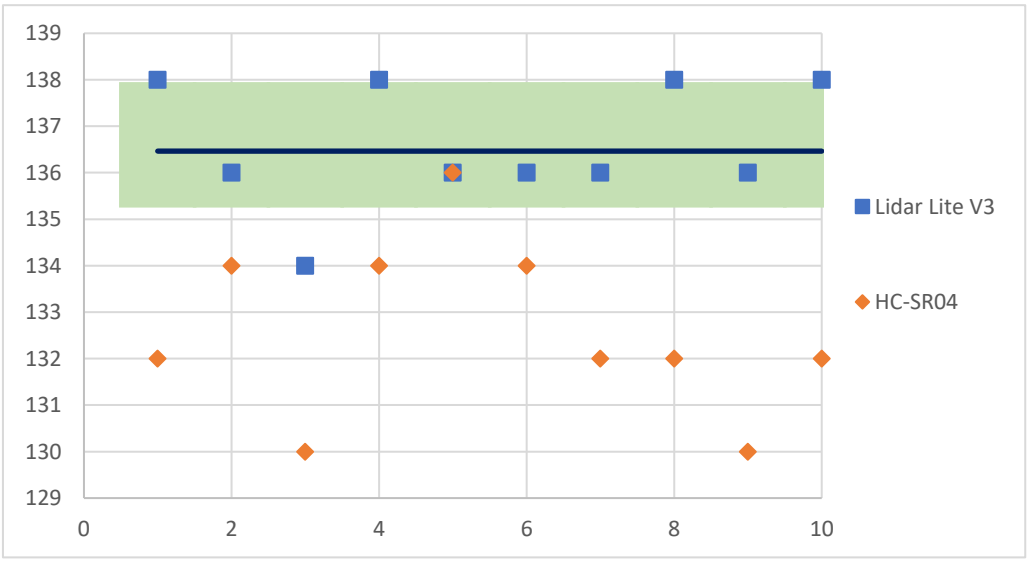


Tabla 2. Gráfica Desviación Ensayo Frontal Inicio Alto

En relación con los datos de la anterior tabla, se pueden ver los resultados del ensayo por parte del sensor láser como por el ultrasónico. En dicha prueba los valores dados por el sensor láser muestran un grado de proximidad mayor al del sensor ultrasónico, con relación al valor obtenido por las fotocélulas.

Por otro lado, si se observan los datos del sensor ultrasónico se refleja una variación entorno a un 3%. Donde sus valores muestran un mayor desviación típica y error en relación con los del láser como se muestra en la gráfica actual.

A continuación, se muestran las mediciones tomadas una vez el neumático inicia su movimiento desde la mitad de la rampa.

VELOCIDAD FRONTAL INICIO MEDIO			
Lidar Lite V3		**HC-SR04**	
Velocidad Sensor (cm/s)	**Error**	**Velocidad Sensor (cm/s)**	**Error**
94	3,88%	94	3,88%
100	-2,26%	96	1,83%
94	3,88%	96	1,83%
96	1,83%	94	3,88%
96	1,83%	96	1,83%
102	-4,31%	94	3,88%
98	-0,21%	94	3,88%
100	-2,26%	98	-0,21%
98	-0,21%	94	3,88%
98	-0,21%	96	1,83%
Velocidad Media	**Error Velocidad Media**	**Velocidad Media**	**Error Velocidad Media**
97,6	0,19%	95,2	2,65%
Varianza	6,93	**Varianza**	1,96

Tabla 3. Datos de Sensores para Ensayo Frontal Inicio Medio

Del mismo modo como se mostró en la pasada **Tabla 1**, se contemplan los resultados obtenidos en el ensayo y como ha variado a la hora de disminuir la velocidad del neumático.

Se puede comprobar que los resultados actuales tienen cierta similitud con los de la **Tabla 1**. La media del sensor láser está próxima a la obtenida por las fotocélulas, en cambio la del sensor ultrasónico sigue estando por debajo.

En esta ocasión la desviación de los datos por parte del Lidar sigue manteniéndose próxima, en cambio se observa que los datos del HC-SR04 se encuentran alejados, pero a la vez muy próximos entre ellos, concentrándose a 95 cm/s.

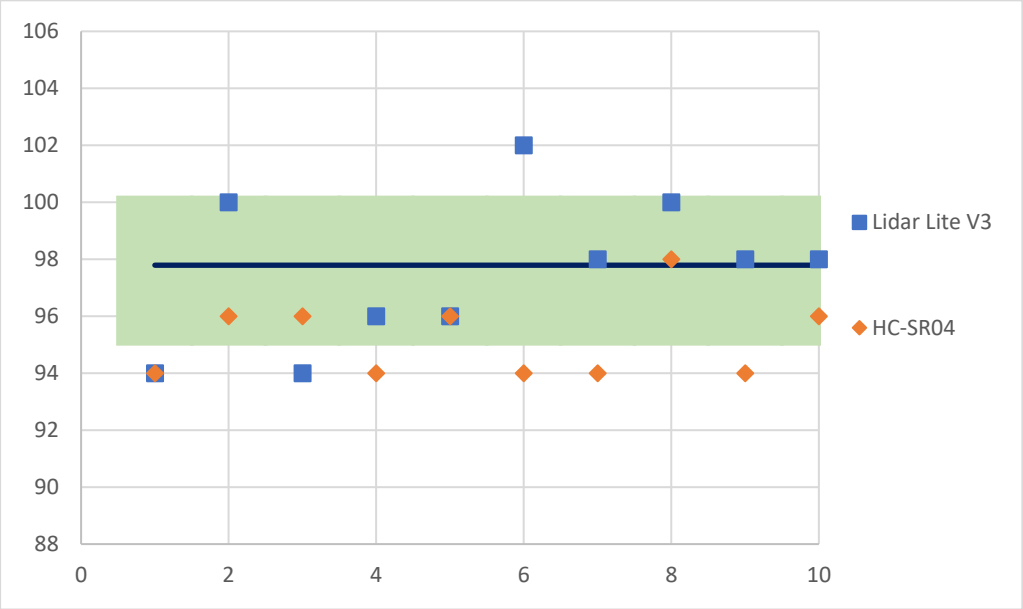


Tabla 4. Gráfica Desviación Ensayo Frontal Inicio Medio

A raíz de los resultados obtenidos en estas pruebas, se puede ultimar una serie de sucesos que influyen en los resultados.

En un primer vistazo los resultados del sensor Lidar son mucho más precioso para el cálculo de la velocidad en comparación con los obtenidos por el sensor HC-SR04 para dichas pruebas.

Centrando la atención en el sensor ultrasónico, dicho sensor no posee una capacidad de obtención de datos como la que tiene el sensor láser. Por tanto, los datos que se extraen no son del todo precisos. Sobre todo, cuando la velocidad del objeto aumenta. Esto queda representado en la **Tabla 2** donde la nube de puntos es mucho más dispersa. En cambio, en la **Tabla 4** aun estando los valores por debajo de la media, sí se observa una mayor correlación de los datos manteniéndose más próximos entre ellos.

Teniendo en cuenta los datos y la capacidad de adquisición de los sensores, en la actual prueba tanto a la altura media y alta, el sensor Lidar sería la mejor opción por su aproximación a los valores de las fotocélulas.

5.2 Resultados Velocidad Perpendicular HC-SR04 y Lidar Lite V3

En este epígrafe se realizará una comparativa entre los resultados obtenidos por ambos sensores a la hora de medir la velocidad del neumático a medida que se

acerca lateralmente a los sensores. Estos resultados hacen alusión a las **Pruebas 3 y 6**.

Antes de exponer los resultados hay que comentar un suceso ocurrido al extraer los datos para la **Prueba 3**. Como se muestra en la **Imagen 15**, el ángulo de apertura para la adquisición de datos es mayor para el sensor ultrasónico. Por tanto, como se expone en la prueba aludida, el objetivo será detectar el neumático una vez entre y salga de su área de acción.

Se comprobó que el programa realizaba dicha función. No obstante, esto ocurría a velocidades para las cuales el neumático viajaba lentamente. Cuando se menciona esta velocidad quiere decirse que el neumático ha comenzado su movimiento al inicio del rango de captación de datos por parte del sensor.

Debido a este suceso, realizar los ensayos a una mayor velocidad no sería conveniente puesto que no se extraerían datos concluyentes como los que se muestran a continuación una vez el neumático empieza en lo alto de la rampa. Es decir, el sensor no detectaría correctamente la presencia del neumático. En cierta medida, la frecuencia de adquisición del sensor, así como su apertura angular, son los causantes de esta problemática.

RESULTADOS PRUEBA 3 INICIO ALTO

Velocidad Sensor (cm/s)	Error
184,34	-35,08%
177,6	-30,14%
143,2	-4,94%
126,4	7,38%
187,83	-37,64%
187,52	-37,41%
153,8	-12,70%
171,09	-25,37%
134,59	1,37%
185,69	-36,07%
Velocidad Media	**Error Velocidad Media**
165,21	-21,06%
Varianza	560,14

Tabla 5. Revisión resultados Prueba 3

Se conoce además que el sensor HC-SR04 no tiene la misma capacidad de adquirir datos como la posee el Lidar Lite V3, esto se comprobó en el **epígrafe 5.1**.

Por tanto, se procedió a modificar la prueba de forma que ahora el neumático no pasaría de forma perpendicular al sensor. El sensor se posicionaría con un cierto ángulo en dirección al avance del neumático.

En cierta medida esta propuesta surge de la idea de extender esta implementación a la medida de velocidad de vehículos circulando en vía pública.

La imposibilidad de situar el sensor frontalmente a los vehículos (en la mayoría de los casos), hizo plantear esta alternativa lateral.

A continuación, se observa la modificación para posteriormente realizar la comparativa de dicho punto.

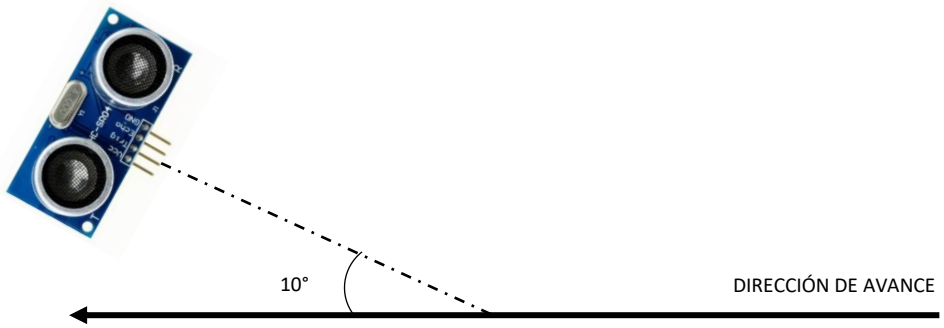


Imagen 38. Modificación Prueba 3

En relación con lo expuesto anteriormente, se procede a mostrar los resultados obtenidos en dicha prueba primeramente arriba de la plataforma de lanzamiento y posteriormente a media altura.

VELOCIDAD PERPENDICULAR INICIO ALTO			
Lidar Lite V3		**HC-SR04**	
Velocidad Sensor (cm/s)	**Error**	**Velocidad Sensor (cm/s)**	**Error**
140,31	-2,82%	130	4,74%
135,58	0,65%	134	1,81%
139,79	-2,44%	132	3,27%
132,59	2,84%	132	3,27%
136,94	-0,35%	134	1,81%
134,80	1,22%	136	0,34%
136,18	0,21%	130	4,74%
138,33	-1,37%	130	4,74%
132,91	2,61%	130	4,74%
134,54	1,41%	132	3,27%
Velocidad Media	**Error Velocidad Media**	**Velocidad Media**	**Error Velocidad Media**
136,20	0,20%	132	3,27%
Varianza	7,10	**Varianza**	4,44

Tabla 6. Datos de Sensores para Ensayo Lateral Inicio Alto

Expuestos los datos en la anterior tabla, se pueden ver los resultados del ensayo por parte del sensor láser como por el ultrasónico. En dicha prueba los valores dados por el sensor láser muestran un grado de proximidad mayor al del sensor ultrasónico, con relación al valor obtenido por las fotocélulas.

Se puede apreciar un aumento en el error porcentual por parte de ambos sensores en comparación a los resultados obtenidos para la misma altura de la

rampa en el **epígrafe 5.1**. Además, como se muestra en la siguiente gráfica los resultados se encuentran algo más dispersos.

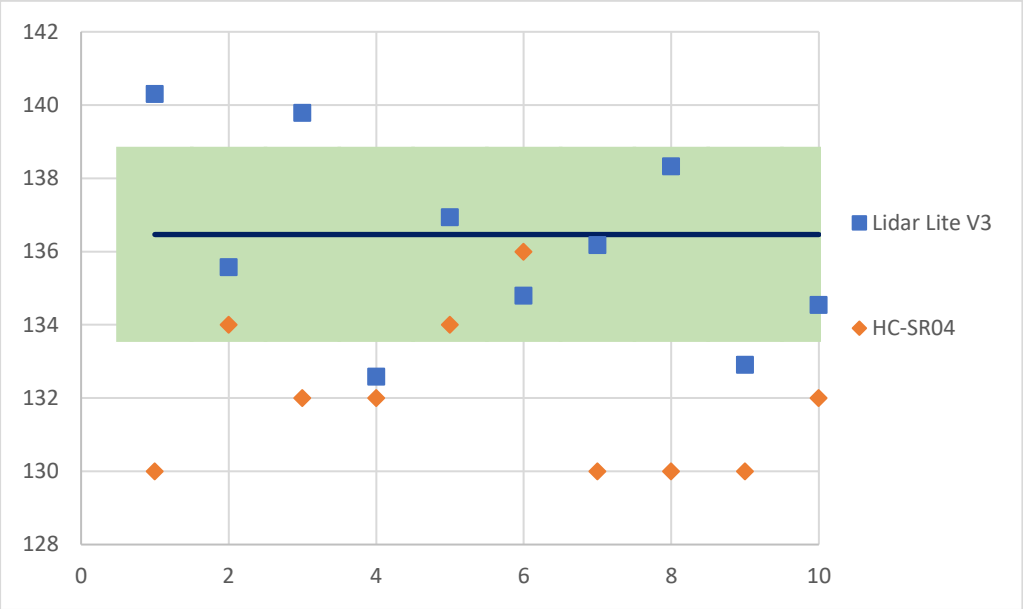


Tabla 7. Gráfica Desviación Ensayo Lateral Inicio Alto

A continuación, se muestran las mediciones tomadas una vez el neumático inicia su movimiento desde la mitad de la rampa.

VELOCIDAD PERPENDICULAR INICIO MEDIO			
Lidar Lite V3		**HC-SR04**	
Velocidad Sensor (cm/s)	**Error**	**Velocidad Sensor (cm/s)**	**Error**
96,36	1,46%	96,67	1,15%
97,48	0,32%	96,67	1,15%
96,74	1,07%	96,67	1,15%
97,02	0,79%	93,33	4,56%
95,33	2,52%	93,33	4,56%
97,43	0,37%	94,5	3,36%
96,92	0,89%	96,67	1,15%
98,27	-0,49%	93,33	4,56%
96,74	1,07%	93,33	4,56%
98,33	-0,55%	96,67	1,15%
Velocidad Media	**Error Velocidad Media**	**Velocidad Media**	**Error Velocidad Media**
97,06	0,74%	95,12	2,73%
Varianza	0,79	**Varianza**	2,80

Tabla 8. Datos de Sensores para Ensayo Lateral Inicio Medio

Como se puede comprobar en esta ocasión los resultados indican un aumento leve del error porcentual por parte del sensor láser. Por otro lado, dicho error se disminuyó en el sensor ultrasónico respecto a los resultados mostrados en la **Tabla 5**.

Para esta prueba la desviación de los resultados muestra que los obtenidos por el sensor láser se mantiene próximos a los obtenidos por las fotocélulas, pero cayendo por debajo de estos. Mientras los obtenidos por el ultrasónico se mantienen en un rango entorno a los 93/96,67 cm/s.

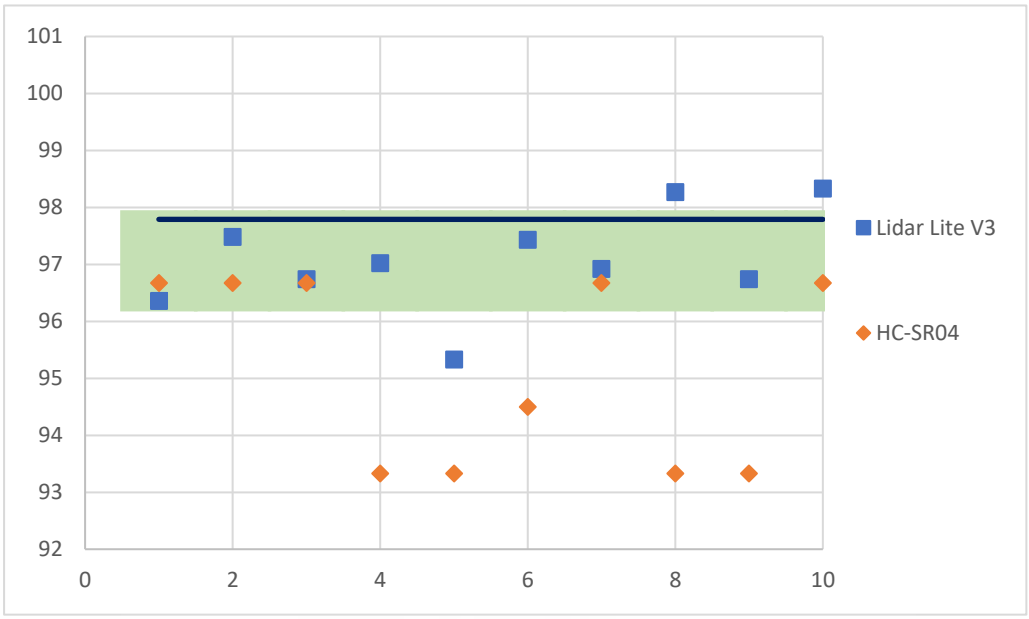


Tabla 9. Gráfica Desviación Ensayo Lateral Inicio Medio

Concluyendo con lo anterior, los resultados muestran una gran precisión por parte del Lidar, en contra posición los resultados del HC-SR04 son algo menos precisos.

Además de lo mencionado al final del **epígrafe 5.1**. Se observa una correlación en los resultados de las pruebas, por una parte a medida que disminuye la velocidad en las pruebas (inicio alto → inicio medio) los datos obtenidos por el sensor ultrasónico se van aproximando poco a poco a los reales.

Por otro lado los resultados por parte del sensor Lidar muestran una gran exactitud en ambas pruebas siendo la de mayor velocidad con una aproximación frontal del neumático, la mas precisa.

6 PRESUPUESTO

Se procede en dicho epígrafe a dar una estimación del posible coste a la hora de realizar dicho proyecto.

COMPONENTES				
	Uds	**€/Ud**		
Placa Arduino Mega 2560	1	46,00		46,00 €
Protoboard	1	1,00		1,00 €
Led Rojo	1	0,20		0,20 €
Resistencia 330Ω	1	0,10		0,10 €
Sensor HC-SR04	1	1,00		1,00 €
Cables Dupont	8	0,05		0,40 €
Sensor Lidar Lite V3	1	155,00		155,00 €
Condensador 1000µF	1	1,00		1,00 €
			Subtotal	**204,70 €**
SOFTWARE				
Arduino IDE	1	0		- €
Microsotf Excel	1	0		- €
Test Xpress 7A	1	0		- €
Aplicación Móvil*	1	0		- €
			Subtotal	- €
TAREAS REALIZADAS				
	Horas	**€/Hora**		
Adquisición de componentes	2	30,00		60,00 €
Montaje	1	30,00		30,00 €
Programación	50	30,00		1.500,00 €
Búsqueda de información	60	30,00		1.800,00 €
Realización de pruebas	15	30,00		450,00 €
Resolución de errores	10	30,00		300,00 €
Redacción del proyecto	90	30,00		2.700,00 €
			Subtotal	**6.840,00 €**

COSTE TOTAL	**7.044,70 €**

*Se dispone de dicho software, el cual se hace alusión al inicio del **epígrafe 5 Resultados**

7 CONCLUSIONES Y TRABAJO FUTURO

En el presente proyecto se ha realizado un estudio comparativo mediante Arduino de dos tipos de sensores de proximidad, para los que se han elaborado ciertas pruebas para evaluar su precisión a la hora de medir la distancia a la cual se encuentran los objetos y su exactitud en el momento del cálculo de la velocidad de los objetos que atraviesen su rango.

Atendiendo a los resultados obtenidos a través de este proyecto, se puede concluir que los datos recogidos a través de los sensores HC-SR04 y Lidar Lite V3 resultan ser prometedores tanto en las pruebas frontales como perpendiculares. Por otro lado, el sensor láser proporciona resultados más próximos a la realidad en comparación con los que obtuvo el sensor ultrasónico. No obstante, esto se debe a la capacidad de adquisición de datos por parte de los sensores puesto que el Lidar posee una mayor frecuencia de adquisición y directividad en comparación con el HC-SR04.

En esta misma línea, añadir a lo mencionado anteriormente que también encontramos una adecuada consecución de los objetivos que se planteaban. Por una parte, el sensor láser ha obtenido los resultados que se esperaban sin embargo el sensor ultrasónico debido a su baja capacidad de adquisición de datos ha resultado ser eficiente en la **Prueba 2** por el contrario en la **Prueba 3** no obtuvo los mejores resultados y por tanto se decidió modificar su esquema como se muestra en la **Imagen 33**, es decir, dado que en la **Prueba 2** el neumático se dirige frontalmente al sensor no se aprecia en gran medida la diferencia de adquisición de datos entre el sensor láser y el ultrasónico, por lo que los resultados son favorables. Sin embargo, esto no ocurre en la **Prueba 3** ya que el neumático se dirige con una dirección perpendicular a la dirección de adquisición del sensor, esquema que se muestra al comienzo del **epígrafe 4**. Debido a los resultados obtenidos en dicha prueba se pasó a modificar el esquema y en lugar de posicionarlo de manera perpendicular al avance del neumático, se posicionó de manera que el sensor se encuentra con un ángulo de 10 grados respecto a la trayectoria del neumático como se muestra en la **Imagen 33**.

Además, cabe señalar que, debido a sus resultados en la posición frontal, ambos sensores pueden ser una buena opción a nivel económico y podrá ser equiparable a otras con un valor más elevado. Por el contrario, en las pruebas perpendiculares el sensor ultrasónico no ha sido preciso en sus cálculos como se ha mencionado anteriormente, sin embargo, cabe destacar que el sensor Lidar ha obtenido unos resultados equiparables a los obtenidos por las fotocélulas.

Teniendo en cuenta estas consideraciones, las posibilidades de mejora y desarrollo del presente trabajo dan pie a la propuesta de las pruebas fuera del laboratorio con el fin de realizar ensayos en entornos reales. En futuras investigaciones sería conveniente realizar los ensayos utilizando un vehículo dotado con un dispositivo GPS que además del cuentakilómetros y un cámara captando el recorrido del vehículo. Estos datos servirían de referencia para el posterior contraste con los sensores, al igual que ha ocurrido con las fotocélulas.

8 BIBLIOGRAFÍA

[1] Ferdeghini, F., Brengi, D., & Lupi, D. (1998, August). Sistema de detección combinado para sensores ultrasónicos. In *XVI Congreso Argentino de Control Automático, AADECA* (Vol. 2, pp. 514-519). https://sistemamid.com/panel/uploads/biblioteca/2015-01-25_08-52-27113879.pdf

[2] Ibarra Villalón, H. E., Pottiez, O., & Gómez Vieyra, A. (2018). El camino hacia la luz láser. *Revista Mexicana de Fisica E. Publicacion de Ensenanza, Historia y Filosofia de la Sociedad Mexicana de Fisica*, *64*(2 Jul-Dec), 100–107. https://doi.org/10.31349/revmexfise.64.100

[3] Musayev, E. (2007). Laser-based large detection area speed measurement methods and systems. *Optics and Lasers in Engineering*, *45*(11), 1049–1054. https://doi.org/10.1016/j.optlaseng.2007.03.007

[4] Echaveguren, Tomás, Díaz, Álvaro, & Arellano, Daniela. (2013). Comparación entre mediciones de velocidad obtenidas con los equipos GPS y Pistola Láser. *Obras y proyectos*, (14), 47-55. https://dx.doi.org/10.4067/S0718-28132013000200004

[5] Javadi, S., Dahl, M., & Pettersson, M. I. (2019). Vehicle speed measurement model for video-based systems. Computers & Electrical Engineering: An International Journal, 76, 238–248. https://doi.org/10.1016/j.compeleceng.2019.04.001

[6] Plaza, D. (2020, May 28). *¿Qué es un cinemómetro? Tipos y umbrales de tolerancia*. Motor.es. https://www.motor.es/que-es/cinemometro

[7] *El Sistema de Posicionamiento Global*. (n.d.). Gps.gov. Retrieved June 20, 2022, from https://www.gps.gov/systems/gps/spanish.php

[8] Wikipedia contributors. (2022, May 24). *Particle image velocimetry*. Wikipedia, The Free Encyclopedia.

https://en.wikipedia.org/w/index.php?title=Particle_image_velocimetry&oldid=1089633520

[9] Luis. (2015, June 16). *Medir distancia con Arduino y sensor de ultrasonidos HC-SR04*. Luis Llamas; Luis. https://www.luisllamas.es/medir-distancia-con-arduino-y-sensor-de-ultrasonidos-hc-sr04/

[10] Garmin, & Garmin Ltd. or its subsidiaries. (n.d.). *LIDAR-Lite v3*. Garmin. Retrieved June 22, 2022, from https://www.garmin.com/es-ES/p/557294

[11] Aguayo, P. (2019, enero 15). *Arduino Mega 2560*. Arduino.cl - Compra tu Arduino en Línea. https://arduino.cl/arduino-mega-2560/

[12] Semanticscholar.Org. from https://www.semanticscholar.org/paper/Ultrasonic-Sonar-Object-and-Range-Detection-Display-Sansury/68a092be44eecf6b9ae6f5bb1e1f2b7b95a2d57c/figure/1

9 ANEXOS

9.1 Código Prueba 1

```
int TRIG = 10;      // trigger en pin 10
int ECO = 9;      // eco en pin 9
int LED = 3;      // LED en pin 3
int DURACION;
int DISTANCIA;

void setup()
{
  pinMode(TRIG, OUTPUT);  // trigger como salida
  pinMode(ECO, INPUT);    // eco como entrada
  pinMode(LED, OUTPUT);   // LED como salida
  Serial.begin(9600);     // inicializacion de comunicacion serial a
9600 bps

}

void loop()
{

  digitalWrite(TRIG, HIGH);     // generacion del pulso a enviar del
sensor
  delay(1);
  digitalWrite(TRIG, LOW);

  DURACION = pulseIn(ECO, HIGH);  // alto en Echo

  DISTANCIA = DURACION / 58.4;    // distancia medida en centimetros
  Serial.println(DISTANCIA);    // envio de valor de distancia por
monitor serial
  delay(200);       // espera entre datos

  if (DISTANCIA <= 100 && DISTANCIA >= 0){ // si distancia entre 0 y
100 cms.
    digitalWrite(LED, HIGH);      // enciende LED
    delay(DISTANCIA * 10);      // espera proporcional a la distancia
    digitalWrite(LED, LOW);     // apaga LED
    }

}
```

9.2 Código Prueba 2

```
int trigPin = 10;
int echoPin = 9;
long duracion;
int distancia1=0;
int distancia2=0;
double Velocidad=0;
int distancia=0;

void setup()
{
pinMode(trigPin, OUTPUT);
pinMode(echoPin, INPUT);
Serial.begin(9600);
}
void loop() {

   distancia1 = ultrasonicRead();

   delay(300);

   distancia2 = ultrasonicRead();


   Velocidad = (distancia1 - distancia2)/0.3

 if(distancia1<200 && distancia2<170 && distancia>70){

  Serial.print("Velocidad en cm/s  :");
  Serial.println(Velocidad);
 }
}
float ultrasonicRead (){
// Clears the trigPin
digitalWrite(trigPin, LOW);
delayMicroseconds(2);

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

duracion = pulseIn(echoPin, HIGH);

distancia= duracion*0.034/2;

//Serial.print("Distancia en cm : ");
//Serial.println(distancia);
return distancia;

}
```

9.3 Código Prueba 3

```
const int trigPin = 10;       // trigger en pin 10
const int echoPin = 9;       // echo en pin 9
int LED = 3;
float distancia;
long duracion;
float t;
float t1;
float t2;
float tiempo;
float Vobj; //cm/s
const float L = 165; //cm
const int h = 117; // cm
const float lobj = 63.4; //cm     Diametro de la rueda 63.4
float Long;
int aux = 0;
int sol = 0;

void setup() {
  pinMode(trigPin, OUTPUT);  // trigger como salida
  pinMode(echoPin, INPUT);    // echo como entrada
  pinMode(LED, OUTPUT);   // LED como salida
  Serial.begin(9600);     // inicializacion de comunicacion serial a
9600 bps
}

void loop() {
  t=millis();

  digitalWrite(trigPin, HIGH);     // generacion del pulso a enviar
  delay(1);        // al pin conectado al trigger
  digitalWrite(trigPin, LOW);    // del sensor

  duracion = pulseIn(echoPin, HIGH);  // con funcion pulseIn se espera
un pulso
            // alto en Echo
  distancia = duracion / 58.4;    // distancia medida en centimetros
  //Serial.print("Distancia al objeto : ");
  // Serial.println(distancia);

if(aux==0 && distancia>=80 && distancia<=h){
  digitalWrite(LED, HIGH);      // enciende LED
  t1=t;
  aux=1;
  Serial.print("Distancia al objeto 1 : ");
  Serial.println(distancia);
  //Serial.print("Tiempo 1 de ejecucion :");
  //Serial.println(t1);
}
if(aux==1 && distancia>h+30){
  digitalWrite(LED, LOW);     // apaga LED
  t2=t;
  aux=0;
  sol=1;
  Serial.print("Distancia al objeto 2 : ");
  Serial.println(distancia);
  //Serial.print("Tiempo 2 de ejecucion :");
  //Serial.println(t2);
}
```

```
if(sol==1 && aux==0){
  Long=L+lobj;
  tiempo=t2-t1;
  Vobj=Long/(tiempo/1000);
  Serial.print("Velocidad del objeto en cm/s :");
  Serial.println(Vobj);
  sol=0;
  }
}
```

9.4 Código Prueba 4

```
#include <Wire.h>
#include <LIDARLite.h>

LIDARLite myLidarLite;

void setup()
{
  Serial.begin(115200);

 myLidarLite.begin(0, true);


  myLidarLite.configure(0);
}

void loop()
{

  Serial.println(myLidarLite.distance());

  for(int i = 0; i < 99; i++)
  {
    Serial.println(myLidarLite.distance(false));
  }
}
```

9.5 Código Prueba 5

```
LIDARLite myLidarLite;
int dist1;
int dist2;
float vel;

void setup()
{
  Serial.begin(115200);

 myLidarLite.begin(0, true);

 myLidarLite.configure(0);
}

void loop()
{
 for(int i = 0; i < 99; i++)
  {
    dist1 = myLidarLite.distance(false);
   // Serial.print(dist1);
    //Serial.println("cm d1");
    delay(500);
    dist2 = myLidarLite.distance(false);
    //Serial.print(dist2);
    //Serial.println("cm d2");

    vel=(dist2-dist1)/0.5;
     if(vel>=0)
    {
    Serial.print(vel);
    Serial.println("cm/s");

    }else{
      vel=-vel;
    Serial.print(vel);
    Serial.println("cm/s");
    }
  }

    }
```

9.6 Código Prueba 6

```
#include <Wire.h>
#include <LIDARLite.h>

LIDARLite myLidarLite;
//int LED = 3;
float Vobj;
float t;
float t1;
float t2;
float tiempo;
float lobj=63;        //Silla 34cm
int aux = 0;
int sol = 0;

void setup() {
  //pinMode(LED, OUTPUT);
  Serial.begin(115200);
  myLidarLite.begin(0, true);
  myLidarLite.configure(0);
}

void loop() {
  t=millis();
  //Serial.println(myLidarLite.distance());
  for(int i = 0; i < 99; i++)
  {
  //Serial.println(myLidarLite.distance(false));
  if(aux==0 && myLidarLite.distance(false)>=100 &&
myLidarLite.distance(false)<200){
  //digitalWrite(LED, HIGH);      // enciende LED
  t1=t;
  aux=1;
}

  if(aux==1 && myLidarLite.distance(false)>220){
  //digitalWrite(LED, LOW);     // apaga LED
  t2=t;
  aux=0;
  sol=1;
}

  if(sol==1 && aux==0){
  tiempo=t2-t1;
  Vobj=lobj/(tiempo/1000);
  Serial.print("Velocidad del objeto en cm/s :");
  Serial.println(Vobj);
  sol=0;
}
}
}
```

9.7 Datasheet

Arduino® MEGA 2560 Rev3

Product Reference Manual
SKU: A000067

Description

Arduino® Mega 2560 is an exemplary development board dedicated for building extensive applications as compared to other maker boards by Arduino. The board accommodates the ATmega2560 microcontroller, which operates at a frequency of 16 MHz. The board contains 54 digital input/output pins, 16 analog inputs, 4 UARTs (hardware serial ports), a USB connection, a power jack, an ICSP header, and a reset button.

Target Areas

3D Printing, Robotics, Maker

Features

- **ATmega2560 Processor**
  - Up to 16 MIPS Throughput at 16MHz
  - 256k bytes (of which 8k is used for the bootloader)
  - 4k bytes EEPROM
  - 8k bytes Internal SRAM
  - 32 × 8 General Purpose Working Registers
  - Real Time Counter with Separate Oscillator
  - Four 8-bit PWM Channels
  - Four Programmable Serial USART
  - Controller/Peripheral SPI Serial Interface
- **ATmega16U2**
  - Up to 16 MIPS Throughput at 16 MHz
  - 16k bytes ISP Flash Memory
  - 512 bytes EEPROM
  - 512 bytes SRAM
  - USART with SPI master only mode and hardware flow control (RTS/CTS)
  - Master/Slave SPI Serial Interface
- **Sleep Modes**
  - Idle
  - ADC Noise Reduction
  - Power-save
  - Power-down
  - Standby
  - Extended Standby
- **Power**
  - USB Connection
  - External AC/DC Adapter
- **I/O**
  - 54 Digital
  - 16 Analog
  - 15 PWM Output

Contents

1 The Board

Arduino® Mega 2560 is a successor board of Arduino Mega, it is dedicated to applications and projects that require large number of input output pins and the use cases which need high processing power. The Arduino® Mega 2560 comes with a much larger set of IOs when we compare it with traditional Uno board considering the form factor of both the boards.

1.1 Application Examples

- **Robotics**: Featuring the high processing capacitity, the Arduino Mega 2560 can handle the extensive robotic applications. It is compatible with the motor controller shield that enables it to control multiple motors at an instance, thus making it perfect of robotic applications. The large number of I/O pins can accommodate many robotic sensors as well.
- **3D Printing**: Algorithms play a significant role in implementation of 3D printers. Arduino Mega 2560 has the power to process these complex algorithms required for 3D printing. Additionally, the slight changes to the code is easily possible with the Arduino IDE and thus 3D printing programs can be customized according to user requirements.
- **Wi-Fi**: Integrating wireless functionality enhances the utility of the applications. Arduino Mega 2560 is compatible with WiFi shields hence allowing the wireless features for the applications in 3D printing and Robotics.

1.2 Accessories

1.3 Related Products

- Arduino® Uno Rev 3
- Arduino® Nano
- Arduino® DUE without headers

2 Ratings

2.1 Recommended Operating Conditions

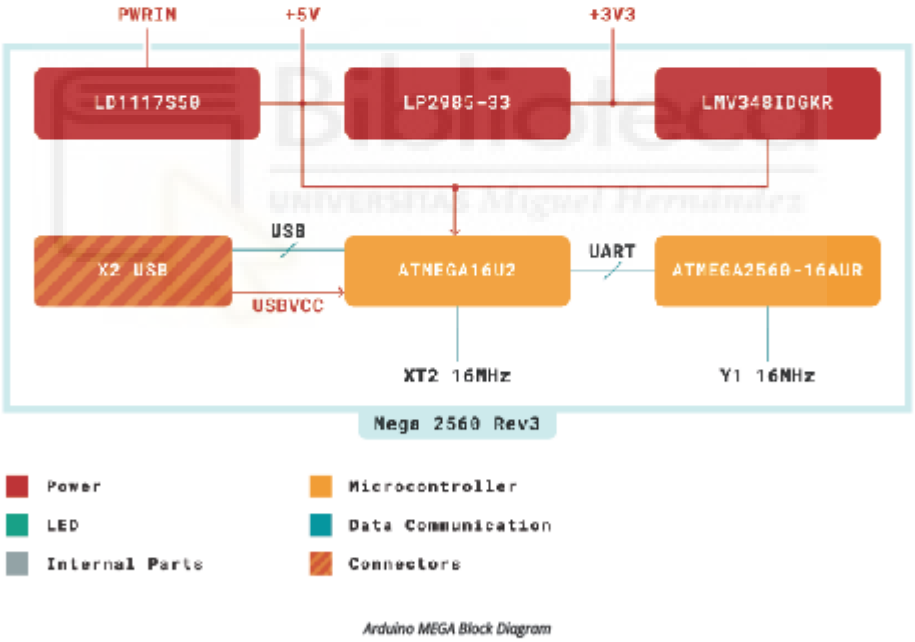
Symbol	Description	Min	Max
	Conservative thermal limits for the whole board:	-40 °C	85 °C

2.2 Power Consumption

Symbol	Description	Min	Typ	Max	Unit
PWRIN	Input supply from power jack		TBC		mW
USB VCC	Input supply from USB		TBC		mW
VIN	Input from VIN pad		TBC		mW

3 Functional Overview

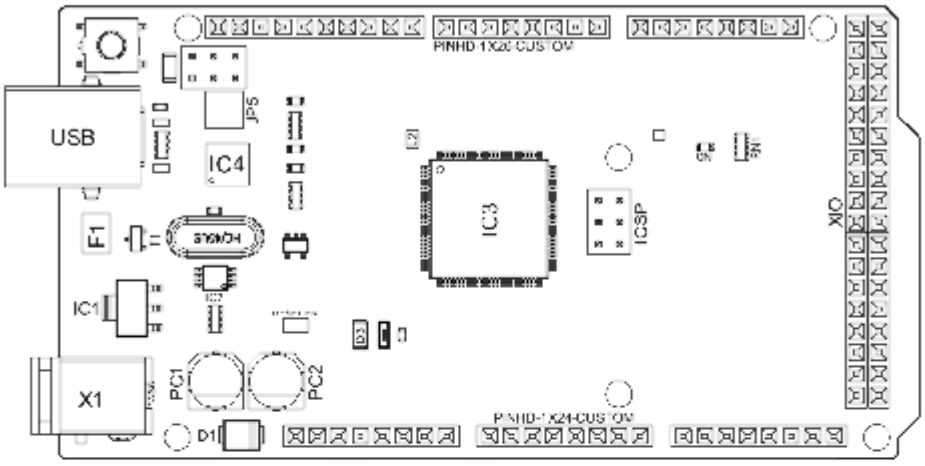
3.1 Block Diagram



Arduino MEGA Block Diagram

3.2 Board Topology

Front View



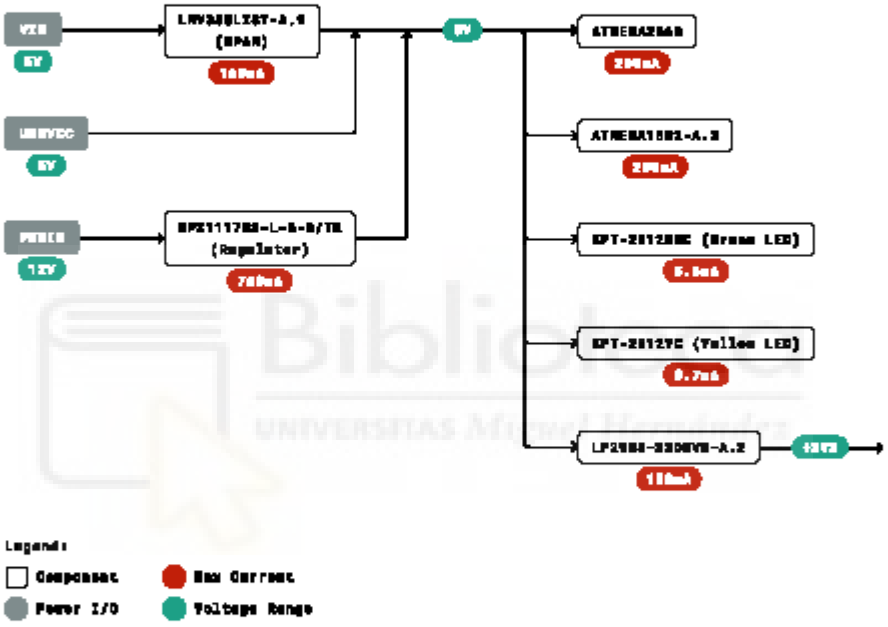
Arduino MEGA Top View

Ref.	Description	Ref.	Description
USB	USB B Connector	F1	Chip Capacitor
IC1	5V Linear Regulator	X1	Power Jack Connector
JP5	Plated Holes	IC4	ATmega16U2 chip
PC1	Electrolytic Alumninum Capacitor	PC2	Electrolytic Alumninum Capacitor
D1	General Purpose Rectifier	D3	General Purpose Diode
L2	Fixed Inductor	IC3	ATmega2560 chip
ICSP	Connector Header	ON	Green LED
RN1	Resistor Array	XIO	Connector

3.3 Processor

Primary processor of Arduino Mega 2560 Rev3 board is ATmega2560 chip which operates at a frequency of 16 MHz. It accommodates a large number of input and output lines which gives the provision of interfacing many external devices. At the same time the operations and processing is not slowed due to its significantly larger RAM than the other processors. The board also features a USB serial processor ATmega16U2 which acts an interface between the USB input signals and the main processor. This increases the flexibility of interfacing and connecting peripherals to the Arduino Mega 2560 Rev 3 board.

3.4 Power Tree



Power Tree

4 Board Operation

4.1 Getting Started - IDE

If you want to program your Arduino® MEGA 2560 while offline you need to install the Arduino® Desktop IDE **[1]** To connect the Arduino® MEGA 2560 to your computer, you'll need a Type-B USB cable. This also provides power to the board, as indicated by the LED.

4.2 Getting Started - Arduino Web Editor

All Arduino® boards, including this one, work out-of-the-box on the Arduino® Web Editor **[2]**, by just installing a simple plugin.

The Arduino® Web Editor is hosted online, therefore it will always be up-to-date with the latest features and support for all boards. Follow **[3]** to start coding on the browser and upload your sketches onto your board.

4.3 Sample Sketches

Sample sketches for the Arduino® MEGA 2560 can be found either in the "Examples" menu in the Arduino® IDE

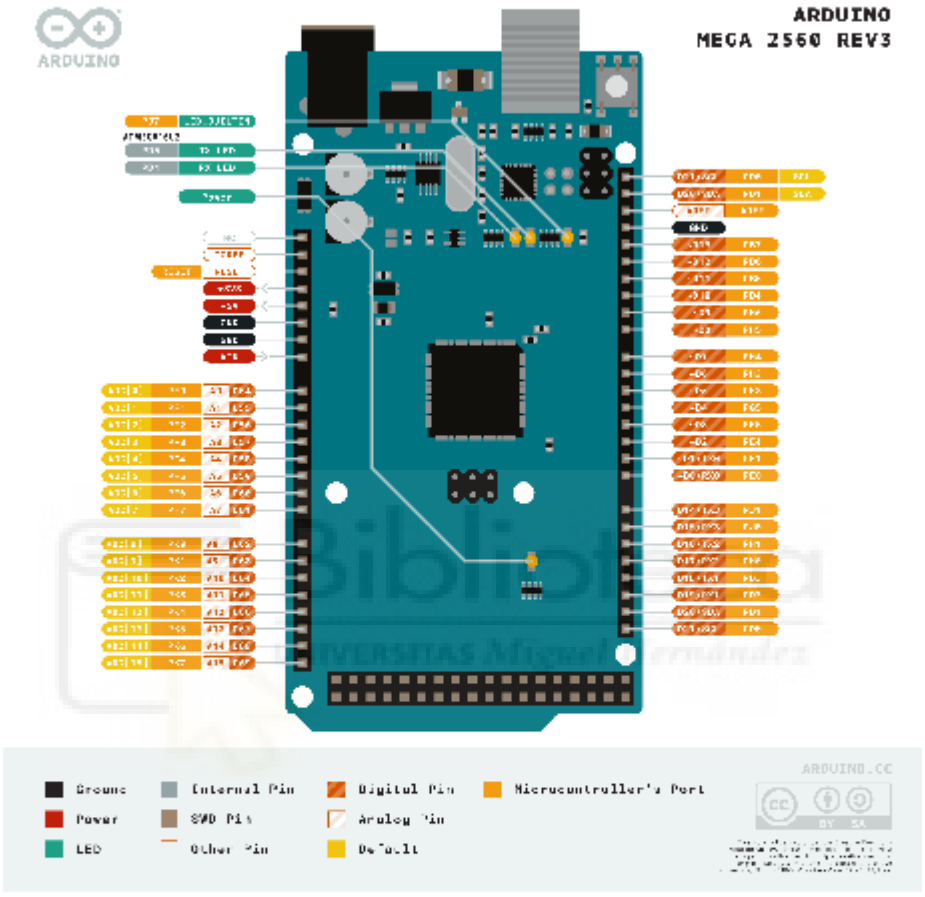
4.4 Online Resources

Now that you have gone through the basics of what you can do with the board you can explore the endless possibilities it provides by checking exciting projects on ProjectHub **[5]**, the Arduino® Library Reference **[6]** and the online store **[7]** where you will be able to complement your board with sensors, actuators and more.

4.5 Board Recovery

All Arduino boards have a built-in bootloader which allows flashing the board via USB. In case a sketch locks up the processor and the board is not reachable anymore via USB it is possible to enter bootloader mode by double-tapping the reset button right after power up.

5 Connector Pinouts



Arduino Mega Pinout

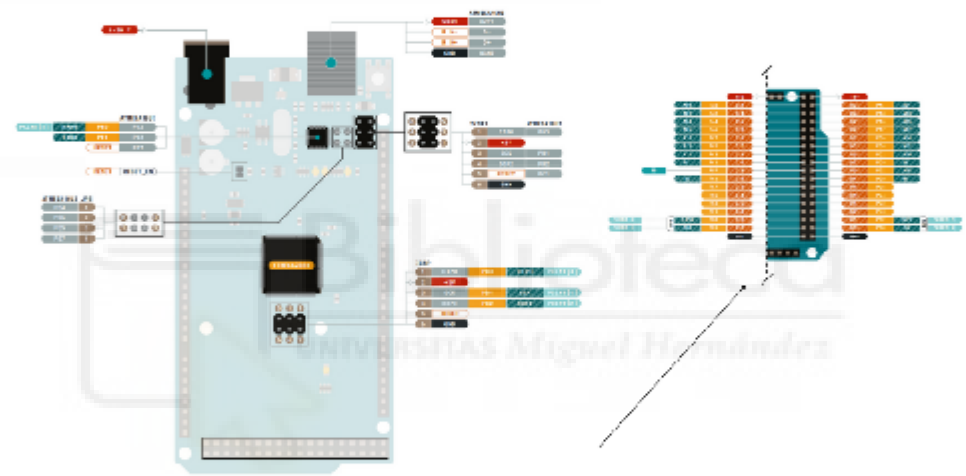
5.1 Analog

Pin	Function	Type	Description
1	NC	NC	Not Connected
2	IOREF	IOREF	Reference for digital logic V - connected to 5V
3	Reset	Reset	Reset
4	+3V3	Power	+3V3 Power Rail
5	+5V	Power	+5V Power Rail
6	GND	Power	Ground
7	GND	Power	Ground
8	VIN	Power	Voltage Input
9	A0	Analog	Analog input 0 /GPIO
10	A1	Analog	Analog input 1 /GPIO
11	A2	Analog	Analog input 2 /GPIO
12	A3	Analog	Analog input 3 /GPIO
13	A4	Analog	Analog input 4 /GPIO
14	A5	Analog	Analog input 5 /GPIO
15	A6	Analog	Analog input 6 /GPIO
16	A7	Analog	Analog input 7 /GPIO
17	A8	Analog	Analog input 8 /GPIO
18	A9	Analog	Analog input 9 /GPIO
19	A10	Analog	Analog input 10 /GPIO
20	A11	Analog	Analog input 11 /GPIO
21	A12	Analog	Analog input 12 /GPIO
22	A13	Analog	Analog input 13 /GPIO
23	A14	Analog	Analog input 14 /GPIO
24	A15	Analog	Analog input 15 /GPIO

5.2 Digital

Pin	Function	Type	Description
1	D21/SCL	Digital Input/I2C	Digital input 21/I2C Dataline
2	D20/SDA	Digital Input/I2C	Digital input 20/I2C Dataline
3	AREF	Digital	Analog Reference Voltage
4	GND	Power	Ground
5	D13	Digital/GPIO	Digital input 13/GPIO
6	D12	Digital/GPIO	Digital input 12/GPIO
7	D11	Digital/GPIO	Digital input 11/GPIO
8	D10	Digital/GPIO	Digital input 10/GPIO
9	D9	Digital/GPIO	Digital input 9/GPIO
10	D8	Digital/GPIO	Digital input 8/GPIO
11	D7	Digital/GPIO	Digital input 7/GPIO
12	D6	Digital/GPIO	Digital input 6/GPIO
13	D5	Digital/GPIO	Digital input 5/GPIO
14	D4	Digital/GPIO	Digital input 4/GPIO

Pin	Function	Type	Description
15	D3	Digital/GPIO	Digital input 3/GPIO
16	D2	Digital/GPIO	Digital input 2/GPIO
17	D1/TX0	Digital/GPIO	Digital input 1 /GPIO
18	D0/Tx1	Digital/GPIO	Digital input 0 /GPIO
19	D14	Digital/GPIO	Digital input 14 /GPIO
20	D15	Digital/GPIO	Digital input 15 /GPIO
21	D16	Digital/GPIO	Digital input 16 /GPIO
22	D17	Digital/GPIO	Digital input 17 /GPIO
23	D18	Digital/GPIO	Digital input 18 /GPIO
24	D19	Digital/GPIO	Digital input 19 /GPIO
25	D20	Digital/GPIO	Digital input 20 /GPIO
26	D21	Digital/GPIO	Digital input 21 /GPIO



Arduino Mega Pinout

5.3 ATMEGA16U2 JP5

Pin	Function	Type	Description
1	PB4	Internal	Serial Wire Debug
2	PB6	Internal	Serial Wire Debug
3	PB5	Internal	Serial Wire Debug
4	PB7	Internal	Serial Wire Debug

5.4 ATMEGA16U2 ICSP1

Pin	Function	Type	Description
1	CIPO	Internal	Controller In Peripheral Out
2	+5V	Internal	Power Supply of 5V
3	SCK	Internal	Serial Clock
4	COPI	Internal	Controller Out Peripheral In
5	RESET	Internal	Reset
6	GND	Internal	Ground

5.5 Digital Pins D22 - D53 LHS

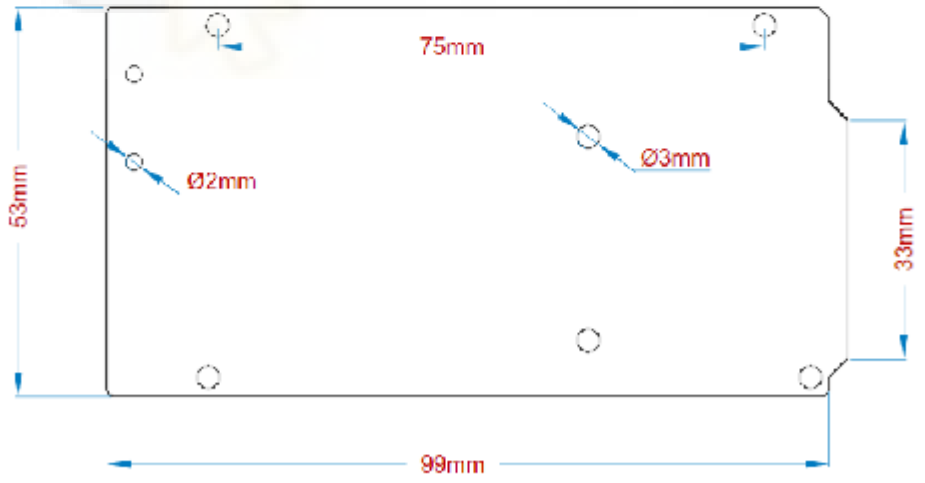
Pin	Function	Type	Description
1	+5V	Power	Power Supply of 5V
2	D22	Digital	Digital input 22/GPIO
3	D24	Digital	Digital input 24/GPIO
4	D26	Digital	Digital input 26/GPIO
5	D28	Digital	Digital input 28/GPIO
6	D30	Digital	Digital input 30/GPIO
7	D32	Digital	Digital input 32/GPIO
8	D34	Digital	Digital input 34/GPIO
9	D36	Digital	Digital input 36/GPIO
10	D38	Digital	Digital input 38/GPIO
11	D40	Digital	Digital input 40/GPIO
12	D42	Digital	Digital input 42/GPIO
13	D44	Digital	Digital input 44/GPIO
14	D46	Digital	Digital input 46/GPIO
15	D48	Digital	Digital input 48/GPIO
16	D50	Digital	Digital input 50/GPIO
17	D52	Digital	Digital input 52/GPIO
18	GND	Power	Ground

5.6 Digital Pins D22 - D53 RHS

Pin	Function	Type	Description
1	+5V	Power	Power Supply of 5V
2	D23	Digital	Digital input 23/GPIO
3	D25	Digital	Digital input 25/GPIO
4	D27	Digital	Digital input 27/GPIO
5	D29	Digital	Digital input 29/GPIO
6	D31	Digital	Digital input 31/GPIO
7	D33	Digital	Digital input 33/GPIO
8	D35	Digital	Digital input 35/GPIO
9	D37	Digital	Digital input 37/GPIO
10	D39	Digital	Digital input 39/GPIO
11	D41	Digital	Digital input 41/GPIO
12	D43	Digital	Digital input 43/GPIO
13	D45	Digital	Digital input 45/GPIO
14	D47	Digital	Digital input 47/GPIO
15	D49	Digital	Digital input 49/GPIO
16	D51	Digital	Digital input 51/GPIO
17	D53	Digital	Digital input 53/GPIO
18	GND	Power	Ground

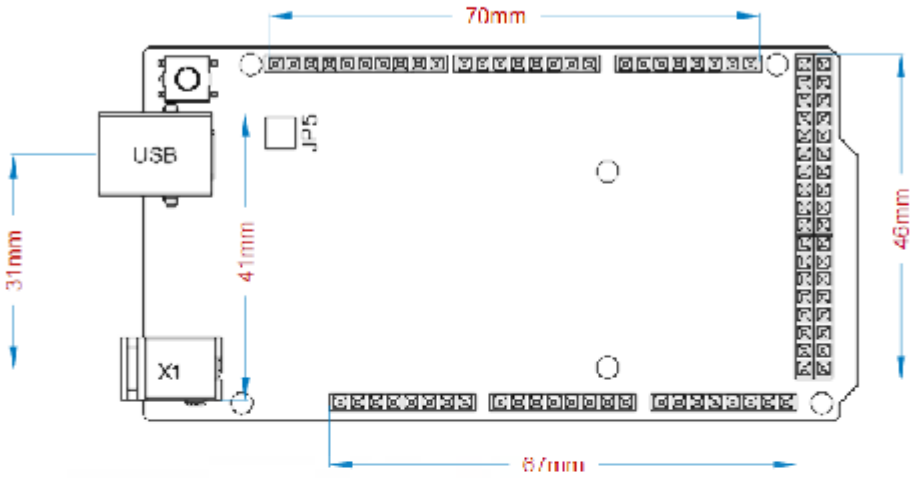
6 Mechanical Information

6.1 Board Outline



Arduino Mega Outline

6.2 Board Mount Holes



Arduino Mega Mount Holes

Certifications

7 Declaration of Conformity CE DoC (EU)

We declare under our sole responsibility that the products above are in conformity with the essential requirements of the following EU Directives and therefore qualify for free movement within markets comprising the European Union (EU) and European Economic Area (EEA).

8 Declaration of Conformity to EU RoHS & REACH 211 01/19/2021

Arduino boards are in compliance with RoHS 2 Directive 2011/65/EU of the European Parliament and RoHS 3 Directive 2015/863/EU of the Council of 4 June 2015 on the restriction of the use of certain hazardous substances in electrical and electronic equipment.

Substance	Maximum Limit (ppm)
Lead (Pb)	1000
Cadmium (Cd)	100
Mercury (Hg)	1000
Hexavalent Chromium (Cr6+)	1000
Poly Brominated Biphenyls (PBB)	1000
Poly Brominated Diphenyl ethers (PBDE)	1000
Bis(2-Ethylhexyl) phthalate (DEHP)	1000
Benzyl butyl phthalate (BBP)	1000
Dibutyl phthalate (DBP)	1000
Diisobutyl phthalate (DIBP)	1000

Exemptions : No exemptions are claimed.

Arduino Boards are fully compliant with the related requirements of European Union Regulation (EC) 1907 /2006 concerning the Registration, Evaluation, Authorization and Restriction of Chemicals (REACH). We declare none of the SVHCs (https://echa.europa.eu/web/guest/candidate-list-table), the Candidate List of Substances of Very High Concern for authorization currently released by ECHA, is present in all products (and also package) in quantities totaling in a concentration equal or above 0.1%. To the best of our knowledge, we also declare that our products do not contain any of the substances listed on the "Authorization List" (Annex XIV of the REACH regulations) and Substances of Very High Concern (SVHC) in any significant amounts as specified by the Annex XVII of Candidate list published by ECHA (European Chemical Agency) 1907 /2006/EC.

9 Conflict Minerals Declaration

As a global supplier of electronic and electrical components, Arduino is aware of our obligations with regards to laws and regulations regarding Conflict Minerals, specifically the Dodd-Frank Wall Street Reform and Consumer Protection Act, Section 1502. Arduino does not directly source or process conflict minerals such as Tin, Tantalum, Tungsten, or Gold. Conflict minerals are contained in our products in the form of solder, or as a component in metal alloys. As part of our reasonable due diligence Arduino has contacted component suppliers within our supply chain to verify their continued compliance with the regulations. Based on the information received thus far we declare that our products contain Conflict Minerals sourced from conflict-free areas.

10 FCC Caution

Any Changes or modifications not expressly approved by the party responsible for compliance could void the user's authority to operate the equipment.

This device complies with part 15 of the FCC Rules. Operation is subject to the following two conditions:

(1) This device may not cause harmful interference

(2) this device must accept any interference received, including interference that may cause undesired operation.

FCC RF Radiation Exposure Statement:

1. This Transmitter must not be co-located or operating in conjunction with any other antenna or transmitter.
2. This equipment complies with RF radiation exposure limits set forth for an uncontrolled environment.
3. This equipment should be installed and operated with minimum distance 20cm between the radiator & your body.

English: User manuals for licence-exempt radio apparatus shall contain the following or equivalent notice in a conspicuous location in the user manual or alternatively on the device or both. This device complies with Industry Canada licence-exempt RSS standard(s). Operation is subject to the following two conditions:

(1) this device may not cause interference

(2) this device must accept any interference, including interference that may cause undesired operation of the device.

French: Le présent appareil est conforme aux CNR d'Industrie Canada applicables aux appareils radio exempts de licence. L'exploitation est autorisée aux deux conditions suivantes :

(1) l' appareil nedoit pas produire de brouillage

(2) l'utilisateur de l'appareil doit accepter tout brouillage radioélectrique subi, même si le brouillage est susceptible d'en compromettre le fonctionnement.

IC SAR Warning:

English This equipment should be installed and operated with minimum distance 20 cm between the radiator and your body.

French: Lors de l' installation et de l' exploitation de ce dispositif, la distance entre le radiateur et le corps est d 'au moins 20 cm.

Important: The operating temperature of the EUT can't exceed 85°C and shouldn't be lower than -40°C.

Hereby, Arduino S.r.l. declares that this product is in compliance with essential requirements and other relevant provisions of Directive 201453/EU. This product is allowed to be used in all EU member states.

11 Company Information

Company name	Arduino S.r.l.
Company Address	Arduino SRL, Via Andrea Appiani 25, 20900 Monza MB, Italy

12 Reference Documentation

Ref	Link
Arduino IDE (Desktop)	https://www.arduino.cc/en/Main/Software
Arduino IDE (Cloud)	https://create.arduino.cc/editor
Cloud IDE Getting Started	https://create.arduino.cc/projecthub/Arduino_Genuino/getting-started-with-arduino-web-editor-4b3e4a
Arduino Pro Website	https://www.arduino.cc/pro
Project Hub	https://create.arduino.cc/projecthub?by=part&part_id=11332&sort=trending
Library Reference	https://www.arduino.cc/reference/en/libraries/
Online Store	https://store.arduino.cc/

13 Revision History

Date	Revision	Changes
29/09/2020	1	First Release

Ultrasonic Ranging Module HC - SR04

Product features:

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

Wire connecting direct as following:

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

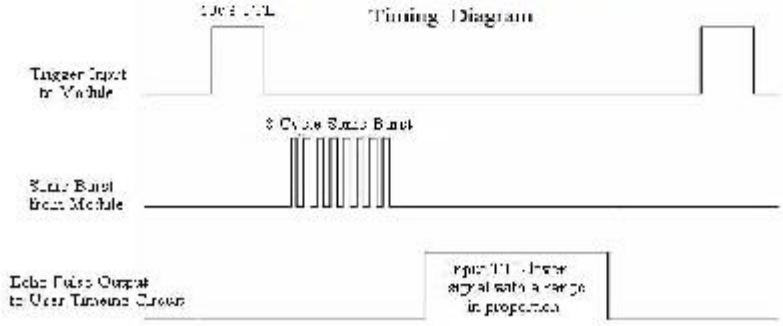
Electric Parameter

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

Vcc Trig Echo GND

Timing diagram

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



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

GARMIN.

Lidar Lite v3 Operation Manual and Technical Specifications

Laser Safety

⚠WARNING

This device requires no regular maintenance. In the event that the device becomes damaged or is inoperable, repair or service must be handled by authorized, factory-trained technicians only. Attempting to repair or service the unit on your own can result in direct exposure to laser radiation and the risk of permanent eye damage. For repair or service, contact your dealer or Garmin® for more information. This device should not be modified or operated without its housing or optics. Operating this device without a housing and optics, or operating this device with modified housing or optics that expose the laser source, may result in direct exposure to laser radiation and the risk of permanent eye damage. Removal or modification of the diffuser in front of the laser optic may result in the risk of permanent eye damage.

Use of controls or adjustments or performance of procedures other than those specified in this documentation may result in hazardous radiation exposure. Garmin is not responsible for injuries caused through the improper use or operation of this product.

⚠CAUTION

This device emits laser radiation. This Laser Product is designated Class 1 during all procedures of operation. This designation means that the laser is safe to look at with the unaided eye, however it is advisable to avoid looking into the beam when operating the device and to turn off the module when not in use.

Documentation Revision Information

Rev	Date	Changes
0A	09/2016	Initial release

Table of Contents

Specifications

Physical

Specification	Measurement
Size (LxWxH)	20 × 48 × 40 mm (0.8 × 1.9 × 1.6 in.)
Weight	22 g (0.78 oz.)
Operating temperature	-20 to 60°C (-4 to 140°F)

Electrical

Specification	Measurement
Power	5 Vdc nominal 4.5 Vdc min., 5.5 Vdc max.
Current consumption	105 mA idle 135 mA continuous operation

Performance

Specification	Measurement
Range (70% reflective target)	40 m (131 ft)
Resolution	+/- 1 cm (0.4 in.)
Accuracy < 5 m	±2.5 cm (1 in.) typical*
Accuracy ≥ 5 m	±10 cm (3.9 in.) typical Mean ±1% of distance maximum Ripple ±1% of distance maximum
Update rate (70% Reflective Target)	270 Hz typical 650 Hz fast mode** >1000 Hz short range only
Repetition rate	~50 Hz default 500 Hz max

*Nonlinearity present below 1 m (39.4 in.)

**Reduced sensitivity

Interface

Specification	Measurement
User interface	I2C PWM External trigger
I2C interface	Fast-mode (400 kbit/s) Default 7-bit address 0x62 Internal register access & control
PWM interface	External trigger input PWM output proportional to distance at 10 µs/cm

Laser

Specification	Measurement
Wavelength	905 nm (nominal)
Total laser power (peak)	1.3 W
Mode of operation	Pulsed (256 pulse max. pulse train)
Pulse width	0.5 µs (50% duty Cycle)
Pulse train repetition frequency	10-20 KHz nominal
Energy per pulse	<280 nJ
Beam diameter at laser aperture	12 × 2 mm (0.47 × 0.08 in.)
Divergence	8 mRadian

Connections

Wiring Harness

Wire Color	Function
Red	5 Vdc (+)
Orange	Power enable (internal pull-up)
Yellow	Mode control
Green	I2C SCL
Blue	I2C SDA
Black	Ground (-)

There are two basic configurations for this device:

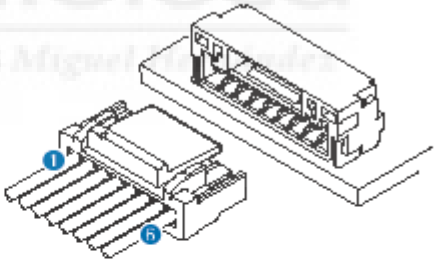
- **I2C (Inter-Integrated Circuit)**—a serial computer bus used to communicate between this device and a microcontroller, such as an Arduino board ("I2C Interface", page 4).
- **PWM (Pulse Width Modulation)**—a bi-directional signal transfer method that triggers acquisitions and returns distance measurements using the mode-control pin ("Mode Control Pin", page 4).

Connector

You can create your own wiring harness if needed for your project or application. The needed components are readily available from many suppliers.

Part	Description	Manufacturer	Part Number
Connector housing	6-position, rectangular housing, latch-lock connector receptacle with a 1.25 mm (0.049 in.) pitch.	JST	GHR-06V-S
Connector terminal	26-30 AWG crimp socket connector terminal (up to 6)	JST	SSHL-002T-P0.2
Wire	UL 1061 26 AWG stranded copper	N/A	N/A

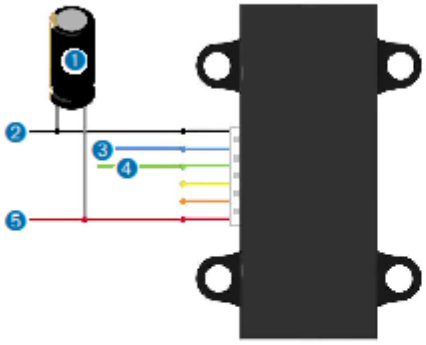
Connector Port Identification



Item	Pin	Function
1	1	5 Vdc (+)
	2	Power enable (internal pull-up)
	3	Mode control
	4	I2C SCL
	5	I2C SDA
6	6	Ground (-)

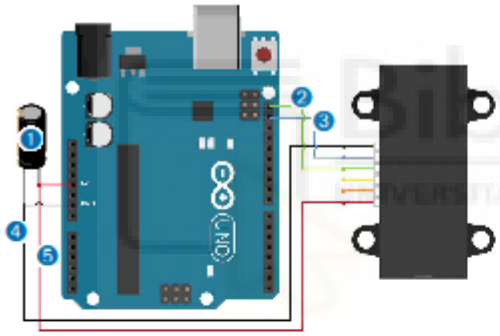
I2C Connection Diagrams

Standard I2C Wiring



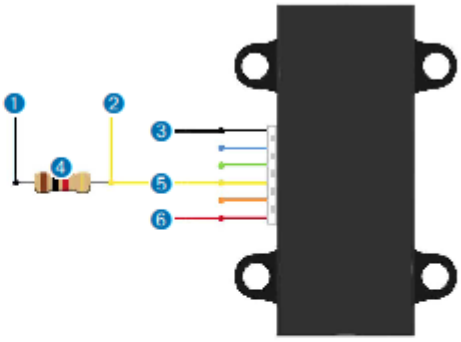
Item	Description	Notes
1	680µF electrolytic capacitor	You must observe the correct polarity when installing the capacitor.
2	Power ground (-) connection	Black wire
3	I2C SDA connection	Blue wire
4	I2C SCA connection	Green wire
5	5 Vdc power (+) connection	Red wire The sensor operates at 4.75 through 5.5 Vdc, with a max. of 6 Vdc.

Standard Arduino I2C Wiring



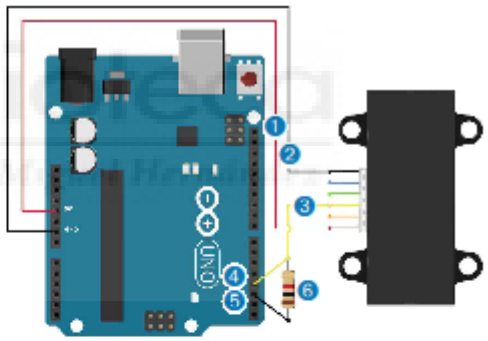
Item	Description	Notes
1	680µF electrolytic capacitor	You must observe the correct polarity when installing the capacitor.
2	I2C SCA connection	Green wire
3	I2C SDA connection	Blue wire
4	Power ground (-) connection	Black wire
5	5 Vdc power (+) connection	Red wire The sensor operates at 4.75 through 5.5 Vdc, with a max. of 6 Vdc.

PWM Wiring



Item	Description	Notes
1	Trigger pin on microcontroller	Connect the other side of the resistor to the trigger pin on your microcontroller.
2	Monitor pin on microcontroller	Connect one side of the resistor to the mode-control connection on the device, and to a monitoring pin on your microcontroller.
3	Power ground (-) connection	Black Wire
4	1kΩ resistor	
5	Mode-control connection	Yellow wire
6	5 Vdc power (+) connection	Red wire The sensor operates at 4.75 through 5.5 Vdc, with a max. of 6 Vdc.

PWM Arduino Wiring



Item	Description	Notes
1	5 Vdc power (+) connection	Red wire The sensor operates at 4.75 through 5.5 Vdc, with a max. of 6 Vdc.
2	Power ground (-) connection	Black Wire
3	Mode-control connection	Yellow wire
4	Monitor pin on microcontroller	Connect one side of the resistor to the mode-control connection on the device, and to a monitoring pin on your microcontroller.
5	Trigger pin on microcontroller	Connect the other side of the resistor to the trigger pin on your microcontroller.
6	1kΩ resistor	

Operational Information

Technology

This device measures distance by calculating the time delay between the transmission of a Near-Infrared laser signal and its reception after reflecting off of a target. This translates into distance using the known speed of light. Our unique signal processing approach transmits a coded signature and looks for that signature in the return, which allows for highly effective detection with eye-safe laser power levels. Proprietary signal processing techniques are used to achieve high sensitivity, speed, and accuracy in a small, low-power, and low-cost system

Theory of Operation

To take a measurement, this device first performs a receiver bias correction routine, correcting for changing ambient light levels and allowing maximum sensitivity.

Then the device sends a reference signal directly from the transmitter to the receiver. It stores the transmit signature, sets the time delay for "zero" distance, and recalculates this delay periodically after several measurements.

Next, the device initiates a measurement by performing a series of acquisitions. Each acquisition is a transmission of the main laser signal while recording the return signal at the receiver. If there is a signal match, the result is stored in memory as a correlation record. The next acquisition is summed with the previous result. When an object at a certain distance reflects the laser signal back to the device, these repeated acquisitions cause a peak to emerge, out of the noise, at the corresponding distance location in the correlation record.

The device integrates acquisitions until the signal peak in the correlation record reaches a maximum value. If the returned signal is not strong enough for this to occur, the device stops at a predetermined maximum acquisition count.

Signal strength is calculated from the magnitude of the signal record peak and a valid signal threshold is calculated from the noise floor. If the peak is above this threshold the measurement is considered valid and the device will calculate the distance, otherwise it will report 1 cm. When beginning the next measurement, the device clears the signal record and starts the sequence again.

Interface

Initialization

On power-up or reset, the device performs a self-test sequence and initializes all registers with default values. After roughly 22 ms distance measurements can be taken with the I2C interface or the Mode Control Pin.

Power Enable Pin

The enable pin uses an internal pullup resistor, and can be driven low to shut off power to the device.

I2C Interface

This device has a 2-wire, I2C-compatible serial interface (refer to I2C-Bus Specification, Version 2.1, January 2000, available from Philips Semiconductor). It can be connected to an I2C bus as a slave device, under the control of an I2C master device. It supports 400 kHz Fast Mode data transfer.

The I2C bus operates internally at 3.3 Vdc. An internal level shifter allows the bus to run at a maximum of 5 Vdc. Internal 3k ohm pullup resistors ensure this functionality and allow for a simple connection to the I2C host.

The device has a 7-bit slave address with a default value of 0x62. The effective 8-bit I2C address is 0xC4 write and 0xC5 read. The device will not respond to a general call. Support is not provided for 10-bit addressing.

Setting the most significant bit of the I2C address byte to one triggers automatic incrementing of the register address with successive reads or writes within an I2C block transfer. This is commonly used to read the two bytes of a 16-bit value within one transfer and is used in the following example.

The simplest method of obtaining measurement results from the I2C interface is as follows:

1. Write 0x04 to register 0x00.
2. Read register 0x01. Repeat until bit 0 (LSB) goes low.
3. Read two bytes from 0x8f (High byte 0x0f then low byte 0x10) to obtain the 16-bit measured distance in centimeters.

A list of all available control resisters is available on page 7.

For more information about the I2C protocol, see I2C Protocol Operation (page 7).

Mode Control Pin

The mode control pin provides a means to trigger acquisitions and return the measured distance via Pulse Width Modulation (PWM) without having to use the I2C interface.

The idle state of the mode control pin is high impedance (High-Z). Pulling the mode control pin low will trigger a single measurement, and the device will respond by driving the line high with a pulse width proportional to the measured distance at 10 µs/cm. A 1k ohm termination resistance is required to prevent bus contention.

The device drives the mode control pin high at 3.3 Vdc. Diode isolation allows the pin to tolerate a maximum of 5 Vdc.

As shown in the diagram PWM Arduino Wiring (page 3), a simple triggering method uses a 1k ohm resistor in series with a host output pin to pull the mode control pin low to initiate a measurement, and a host input pin connected directly to monitor the low-to-high output pulse width.

If the mode control pin is held low, the acquisition process will repeat indefinitely, producing a variable frequency output proportional to distance.

The mode control pin behavior can be modified with the ACQ_CONFIG_REG (0x04) I2C register as detailed in 0x04 (page 8).

Settings

The device can be configured with alternate parameters for the distance measurement algorithm. This can be used to customize performance by enabling configurations that allow choosing between speed, range and sensitivity. Other useful features are also detailed in this section. See the full register map (Control Register List (page 7) for additional settings not mentioned here.

Receiver Bias Correction

Address	Name	Description	Initial Value
0x00	ACQ_COMMAND	Device command	–

- Write 0x00: Reset device, all registers return to default values
- Write 0x03: Take distance measurement without receiver bias correction
- Write 0x04: Take distance measurement with receiver bias correction

Faster distance measurements can be performed by omitting the receiver bias correction routine. Measurement accuracy and sensitivity are adversely affected if conditions change (e.g. target distance, device temperature, and optical noise). To achieve good performance at high measurement rates receiver bias correction must be performed periodically. The recommended method is to do so at the beginning of every 100 sequential measurement commands.

Maximum Acquisition Count

Address	Name	Description	Initial Value
0x02	SIG_COUNT_VAL	Maximum acquisition count	0x80

The maximum acquisition count limits the number of times the device will integrate acquisitions to find a correlation record peak (from a returned signal), which occurs at long range or with low target reflectivity. This controls the minimum measurement rate and maximum range. The unit-less relationship is roughly as follows: rate = 1/n and range = n^(1/4), where n is the number of acquisitions.

Measurement Quick Termination Detection

Address	Name	Description	Initial Value
0x04	ACQ_CONFIG_REG	Acquisition mode control	0x08

You can enable quick-termination detection by clearing bit 3 in this register. The device will terminate a distance measurement early if it anticipates that the signal peak in the correlation record will reach maximum value. This allows for faster and slightly less accurate operation at strong signal strengths without sacrificing long range performance.

Detection Sensitivity

Address	Name	Description	Initial Value
0x1c	THRESHOLD_ BYPASS	Peak detection threshold bypass	0x00

The default valid measurement detection algorithm is based on the peak value, signal strength, and noise in the correlation record. This can be overridden to become a simple threshold criterion by setting a non-zero value. Recommended non-default values are 0x20 for higher sensitivity with more frequent erroneous measurements, and 0x60 for reduced sensitivity and fewer erroneous measurements.

Burst Measurements and Free Running Mode

Address	Name	Description	Initial Value
0x04	ACQ_CONFIG_REG	Acquisition mode control	0x08
0x11	OUTER_LOOP_ COUNT	Burst measurement count control	0x00
0x45	MEASURE_DELAY	Delay between automatic measurements	0x14

The device can be configured to take multiple measurements for each measurement command or repeat indefinitely for free running mode.

OUTER_LOOP_COUNT (0x11) controls the number of times the device will retrigger itself. Values 0x00 or 0x01 result in the default one measurement per command. Values 0x02 to 0xfe directly set the repetition count. Value 0xff will enable free running mode after the host device sends an initial measurement command.

The default delay between automatic measurements corresponds to a 10 Hz repetition rate. Set bit 5 in ACQ_CONFIG_REG (0x04) to use the delay value in MEASURE_DELAY (0x45) instead. A delay value of 0x14 roughly corresponds to 100Hz.

The delay is timed from the completion of each measurement. The means that measurement duration, which varies with returned signal strength, will affect the repetition rate. At low repetition rates (high delay) this effect is small, but for lower delay values it is recommended to limit the maximum acquisition count if consistent frequency is desired.

Velocity

Address	Name	Description	Initial Value
0x09	VELOCITY	Velocity measurement output	--

The velocity measurement is the difference between the current measurement and the previous one, resulting in a signed (2's complement) 8-bit number in cm. Positive velocity is away from the device. This can be combined with free running mode for a constant measurement frequency. The default free running frequency of 10 Hz therefore results in a velocity measurement in .1 m/s.

Configurable I2C Address

Address	Name	Description	Initial Value
0x16	UNIT_ID_HIGH	Serial number high byte	Unique
0x17	UNIT_ID_LOW	Serial number low byte	Unique
0x18	I2C_ID_HIGH	Write serial number high byte for I2C address unlock	--
0x19	I2C_ID_LOW	Write serial number low byte for I2C address unlock	--
0x1a	I2C_SEC_ADDR	Write new I2C address after unlock	--
0x1e	I2C_CONFIG	Default address response control	0x00

The I2C address can be changed from its default value. Available addresses are 7-bit values with a '0' in the least significant bit (even hexadecimal numbers).

To change the I2C address, the unique serial number of the unit must be read then written back to the device before setting the new address. The process is as follows:

1. Read the two byte serial number from 0x96 (High byte 0x16 and low byte 0x17).
2. Write the serial number high byte to 0x18.
3. Write the serial number low byte to 0x19.
4. Write the desired new I2C address to 0x1a.
5. Write 0x08 to 0x1e to disable the default address.

This can be used to run multiple devices on a single bus, by enabling one, changing its address, then enabling the next device and repeating the process.

The I2C address will be restored to default after a power cycle.

Power Control

Address	Name	Description	Initial Value
0x65	POWER_CONTROL	Power state control	0x80

NOTE: The most effective way to control power usage is to utilize the enable pin to deactivate the device when not in use.

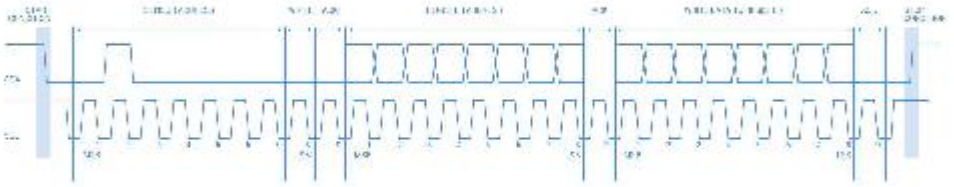
Another option is to set bit 0 in this register which disables the receiver circuit, saving roughly 40mA. After being re-enabled, the receiver circuit stabilizes by the time a measurement can be performed. Setting bit 2 puts the device in sleep mode until the next I2C transaction, saving 20mA. **Since the wake-up time is only around 2 m/s shorter than the full power-on time, and both will reset all registers, it is recommended to use the enable pin instead.**

I2C Protocol Information

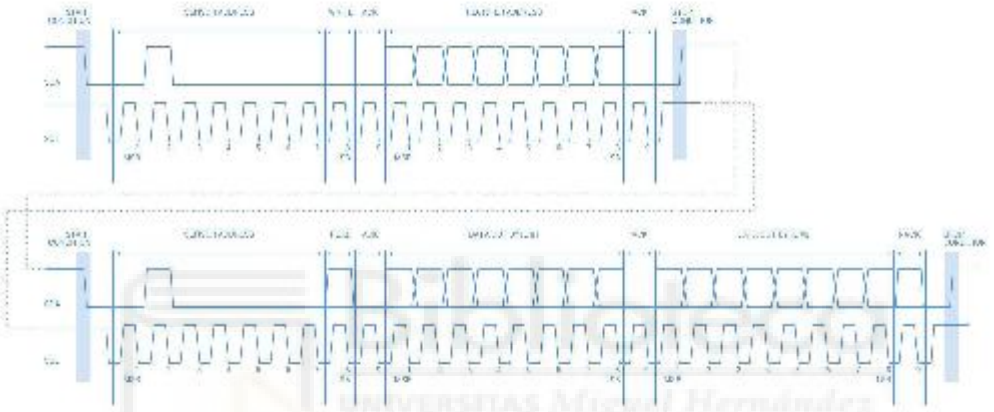
This device has a 2-wire, I2C-compatible serial interface (refer to I2C-Bus Specification, Version 2.1, January 2000, available from Philips Semiconductor). It can be connected to an I2C bus as a slave device, under the control of an I2C master device. It supports standard 400 kHz data transfer mode. Support is not provided for 10-bit addressing.

The Sensor module has a 7-bit slave address with a default value of 0x62 in hexadecimal notation. The effective 8 bit I2C address is: 0xC4 write, 0xC5 read. The device will not presently respond to a general call.

Write



Read



Notes:

- This device does not work with repeated START conditions. It must first receive a STOP condition before a new START condition.
- The ACK and NACK items are responses from the master device to the slave device.
- The last NACK in the read is technically optional, but the formal I2C protocol states that the master shall not acknowledge the last byte.

I2C Protocol Operation

The I2C serial bus protocol operates as follows:

1. The master initiates data transfer by establishing a start condition, which is when a high-to-low transition on the SDA line occurs while SCL is high. The following byte is the address byte, which consists of the 7-bit slave address followed by a read/write bit with a zero state indicating a write request. A write operation is used as the initial stage of both read and write transfers. If the slave address corresponds to the module's address the unit responds by pulling SDA low during the ninth clock pulse (this is termed the acknowledge bit). At this stage, all other devices on the bus remain idle while the selected device waits for data to be written to or read from its shift register.
2. Data is transmitted over the serial bus in sequences of nine clock pulses (eight data bits followed by an acknowledge bit). The transitions on the SDA line must occur during the low period of SCL and remain stable during the high period of SCL.
3. An 8 bit data byte following the address loads the I2C control register with the address of the first control register to be read along with flags indicating if auto increment of the addressed control register is desired with successive reads or writes; and if access to the internal micro or external correlation processor register space is requested. Bit locations 5:0 contain the control register address while bit 7 enables the automatic incrementing of control register with successive data blocks. Bit position 6 selects correlation memory external to the microcontroller if set. (Presently an advanced feature)
4. If a read operation is requested, a stop bit is issued by the master at the completion of the first data frame followed by the initiation of a new start condition, slave address with the read bit set (one state). The new address byte is followed by the reading of one or more data bytes succession. After the slave has acknowledged receipt of a valid address, data read operations proceed by the master releasing the I2C data line SDA with continuing clocking of SCL. At the completion of the receipt of a data byte, the master must strobe the acknowledge bit before continuing the read cycle.
5. For a write operation to proceed, Step 3 is followed by one or more 8 bit data blocks with acknowledges provided by the slave at the completion of each successful transfer. At the completion of the transfer cycle a stop condition is issued by the master terminating operation.

Register Definitions

Control Register List

Address	R/W	Name	Description	Intial Value	Details
0x00	W	ACQ_COMMAND	Device command	--	page 8
0x01	R	STATUS	System status	--	page 8
0x02	R/W	SIG_COUNT_VAL	Maximum acquisition count	0x80	page 8
0x04	R/W	ACQ_CONFIG_REG	Acquisition mode control	0x08	page 8
0x09	R	VELOCITY	Velocity measurement output	--	page 8
0x0c	R	PEAK_CORR	Peak value in correlation record	--	page 8
0x0d	R	NOISE_PEAK	Correlation record noise floor	--	page 8
0x0e	R	SIGNAL_STRENGTH	Received signal strength	--	page 9
0x0f	R	FULL_DELAY_HIGH	Distance measurement high byte	--	page 9
0x10	R	FULL_DELAY_LOW	Distance measurement low byte	--	page 9
0x11	R/W	OUTER_LOOP_COUNT	Burst measurement count control	0x01	page 9
0x12	R/W	REF_COUNT_VAL	Reference acquisition count	0x05	page 9
0x14	R	LAST_DELAY_HIGH	Previous distance measurement high byte	--	page 9
0x15	R	LAST_DELAY_LOW	Previous distance measurement low byte	--	page 9
0x16	R	UNIT_ID_HIGH	Serial number high byte	Unique	page 9
0x17	R	UNIT_ID_LOW	Serial number low byte	Unique	page 9
0x18	W	I2C_ID_HIGH	Write serial number high byte for I2C address unlock	--	page 9
0x19	W	I2C_ID_LOW	Write serial number low byte for I2C address unlock	--	page 9
0x1a	R/W	I2C_SEC_ADDR	Write new I2C address after unlock	--	page 9
0x1c	R/W	THRESHOLD_BYPASS	Peak detection threshold bypass	0x00	page 9
0x1e	R/W	I2C_CONFIG	Default address response control	0x00	page 9
0x40	R/W	COMMAND	Make command	--	page 10
0x45	R/W	MEASURE_DELAY	Delay between automatic measurements	0x14	page 10
0x4c	R	PEAK_BCK	Second largest peak value in correlation record	--	page 10
0x52	R	CORR_DATA	Correlation record data low byte	--	page 10
0x53	R	CORR_DATA_SIGN	Correlation record data high byte	--	page 10
0x5d	R/W	ACQ_SETTINGS	Correlation record memory bank select	--	page 10
0x65	R/W	POWER_CONTROL	Power state control	0x80	page 10

Detailed Control Register Definitions

NOTE: Unless otherwise noted, all registers contain one byte and are read and write.

0x00

R/W	Name	Description	Initial Value
W	ACQ_COMMAND	Device command	–

Bit	Function
7:0	Write 0x00: Reset FPGA, all registers return to default values Write 0x03: Take distance measurement without receiver bias correction Write 0x04: Take distance measurement with receiver bias correction

0x01

R/W	Name	Description	Initial Value
R	STATUS	System status	–

Bit	Function
6	Process Error Flag 0: No error detected 1: System error detected during measurement
5	Health Flag 0: Error detected 1: Reference and receiver bias are operational
4	Secondary Return Flag 0: No secondary return detected 1: Secondary return detected in correlation record
3	Invalid Signal Flag 0: Peak detected 1: Peak not detected in correlation record, measurement is invalid
2	Signal Overflow Flag 0: Signal data has not overflowed 1: Signal data in correlation record has reached the maximum value before overflow. This occurs with a strong received signal strength
1	Reference Overflow Flag 0: Reference data has not overflowed 1: Reference data in correlation record has reached the maximum value before overflow. This occurs periodically
0	Busy Flag 0: Device is ready for new command 1: Device is busy taking a measurement

0x02

R/W	Name	Description	Initial Value
R/W	SIG_COUNT_VAL	Maximum acquisition count	0x80

Bit	Function
7:0	Maximum number of acquisitions during measurement

0x04

R/W	Name	Description	Initial Value
R/W	ACQ_CONFIG_REG	Acquisition mode control	0x08

Bit	Function
6	0: Enable reference process during measurement 1: Disable reference process during measurement
5	0: Use default delay for burst and free running mode 1: Use delay from MEASURE_DELAY (0x45) for burst and free running mode
4	0: Enable reference filter, averages 8 reference measurements for increased consistency 1: Disable reference filter
3	0: Enable measurement quick termination. Device will terminate distance measurement early if it anticipates that the signal peak in the correlation record will reach maximum value. 1: Disable measurement quick termination.
2	0: Use default reference acquisition count of 5. 1: Use reference acquisition count from REF_COUNT_VAL (0x12).
1:0	Mode Select Pin Function Control 00: Default PWM mode. Pull pin low to trigger measurement, device will respond with an active high output with a duration of 10us/cm. 01: Status output mode. Device will drive pin active high while busy. Can be used to interrupt host device. 10: Fixed delay PWM mode. Pulling pin low will not trigger a measurement. 11: Oscillator output mode. Nominal 31.25 kHz output. The accuracy of the silicon oscillator in the device is generally within 1% of nominal. This affects distance measurements proportionally and can be measured to apply a compensation factor.

0x09

R/W	Name	Description	Initial Value
R	VELOCITY	Velocity measurement output	–

Bit	Function
7:0	Velocity measurement output. The difference between the current measurement and the previous one, signed (2's complement) value in centimeters.

0x0c

R/W	Name	Description	Initial Value
R	PEAK_CORR	Peak value in correlation record	–

Bit	Function
7:0	The value of the highest peak in the correlation record.

0x0d

R/W	Name	Description	Initial Value
R	NOISE_PEAK	Correlation record noise floor	–

Bit	Function
7:0	A measure of the noise in the correlation record. Will be slightly above the third highest peak.

0x0e

R/W	Name	Description	Initial Value
R	SIGNAL_STRENGTH	Received signal strength	--

Bit	Function
7:0	Received signal strength calculated from the value of the highest peak in the correlation record and how many acquisitions were performed.

0x0f

R/W	Name	Description	Initial Value
R	FULL_DELAY_HIGH	Distance measurement high byte	--

Bit	Function
7:0	Distance measurement result in centimeters, high byte.

0x10

R/W	Name	Description	Initial Value
R	FULL_DELAY_LOW	Distance measurement low byte	--

Bit	Function
7:0	Distance measurement result in centimeters, low byte.

0x11

R/W	Name	Description	Initial Value
R/W	OUTER_LOOP_COUNT	Burst measurement count control	0x01

Bit	Function
7:0	0x00-0x01: One measurement per distance measurement command. 0x02-0xfe: Repetition count per distance measurement command. 0xff: Indefinite repetitions after initial distance measurement command. See ACQ_CONFIG_REG (0x04) and MEASURE_DELAY (0x45) for non-default automatic repetition delays.

0x12

R/W	Name	Description	Initial Value
R/W	REF_COUNT_VAL	Reference acquisition count	0x05

Bit	Function
7:0	Non-default number of reference acquisitions during measurement. ACQ_CONFIG_REG (0x04) bit 2 must be set.

0x14

R/W	Name	Description	Initial Value
R	LAST_DELAY_HIGH	Previous distance measurement high byte	--

Bit	Function
7:0	Previous distance measurement result in centimeters, high byte.

0x15

R/W	Name	Description	Initial Value
R	LAST_DELAY_LOW	Previous distance measurement low byte	--

Bit	Function
7:0	Previous distance measurement result in centimeters, low byte.

0x16

R/W	Name	Description	Initial Value
R	UNIT_ID_HIGH	Serial number high byte	Unique

Bit	Function
7:0	Unique serial number of device, high byte.

0x17

R/W	Name	Description	Initial Value
R	UNIT_ID_LOW	Serial number low byte	Unique

Bit	Function
7:0	Unique serial number of device, high byte.

0x18

R/W	Name	Description	Initial Value
W	I2C_ID_HIGH	Write serial number high byte for I2C address unlock	--

Bit	Function
7:0	Write the value in UNIT_ID_HIGH (0x16) here as part of enabling a non-default I2C address. See I2C_ID_LOW (0x19) and I2C_SEC_ADDR (0x1a).

0x19

R/W	Name	Description	Initial Value
W	I2C_ID_LOW	Write serial number low byte for I2C address unlock	--

Bit	Function
7:0	Write the value in UNIT_ID_LOW (0x17) here as part of enabling a non-default I2C address. See I2C_ID_HIGH (0x18) and I2C_SEC_ADDR (0x1a).

0x1a

R/W	Name	Description	Initial Value
R/W	I2C_SEC_ADDR	Write new I2C address after unlock	--

Bit	Function
7:0	Non-default I2C address. Available addresses are 7-bit values with a '0' in the least significant bit (even hexadecimal numbers). I2C_ID_HIGH (0x18) and I2C_ID_LOW (0x19) must have the correct value for the device to respond to the non-default I2C address.

0x1c

R/W	Name	Description	Initial Value
R/W	THRESHOLD_BYPASS	Peak detection threshold bypass	0x00

Bit	Function
7:0	0x00: Use default valid measurement detection algorithm based on the peak value, signal strength, and noise in the correlation record. 0x01-0xff: Set simple threshold for valid measurement detection. Values 0x20-0x60 generally perform well.

0x1e

R/W	Name	Description	Initial Value
R/W	I2C_CONFIG	Default address response control	0x00

Bit	Function

3	0: Device will respond to I2C address 0x62. Device will also respond to non-default address if configured successfully. See I2C_ID_HIGH (0x18), I2C_ID_LOW (0x19), and I2C_SEC_ADDR (0x1a). 1: Device will only respond to non-default I2C address. It is recommended to configure the non-default address first, then use the non-default address to write to this register, ensuring success.

0x40

R/W	Name	Description	Initial Value
R/W	COMMAND	State command	–

Bit	Function
2:0	000: Test mode disable, resume normal operation 111: Test mode enable, allows download of correlation record Select correlation memory bank in ACQ_SETTINGS (0x5d) prior to enabling test mode. Once test mode is enabled, read CORR_DATA (0x52) and CORR_DATA_SIGN (0x53) in one transaction (read from 0xd2). The memory index is incremented automatically and successive reads produce sequential data.

0x45

R/W	Name	Description	Initial Value
R/W	MEASURE_DELAY	Delay between automatic measurements	0x14

Bit	Function
7:0	Non-default delay after completion of measurement before automatic retrigger, in burst and continuous modes. ACQ_CONFIG_REG (0x04) bit 5 must be set. Value 0xc8 corresponds to 10 Hz repetition rate and 0x14 to roughly 100 Hz.

0x4c

R/W	Name	Description	Initial Value
R	PEAK_BCK	Second largest peak value in correlation record	–

Bit	Function
7:0	The value of the second highest peak in the correlation record.

0x52

R/W	Name	Description	Initial Value
R	CORR_DATA	Correlation record data low byte	–

Bit	Function
7:0	Correlation record data low byte. See CORR_DATA_SIGN (0x53), ACQ_SETTINGS (0x5d), and COMMAND (0x40).

0x53

R/W	Name	Description	Initial Value
R	CORR_DATA_SIGN	Correlation record data high byte	–

Bit	Function
7:0	Correlation record data high byte. Correlation record data is a 2's complement 9-bit value, and must be sign extended to be formatted as a 16-bit 2's complement value. Thus when repacking the two bytes obtained for the I2C transaction, set the high byte to 0xff if the LSB of the high byte is one.

0x5d

R/W	Name	Description	Initial Value
R/W	ACQ_SETTINGS	Correlation record memory bank select	–

Bit	Function
7:6	11: Access correlation memory bank. Write prior to test mode enable, see COMMAND (0x40).

0x65

R/W	Name	Description	Initial Value
R/W	POWER_CONTROL	Power state control	0x80

Bit	Function
2	1: Device Sleep, wakes upon I2C transaction. Registers are reinitialized, wakeup time similar to full reset using enable pin. 0: Device awake
0	1: Disable receiver circuit 0: Enable receiver circuit. Receiver circuit stabilizes by the time a measurement can be performed.

11

Frequently Asked Questions

Must the device run on 5 Vdc? Can it run on 3.3 Vdc instead?

The device requires 5 Vdc to run properly, so this specification is recommended and supported.

What is the spread of the laser beam?

At very close distances (less than 1 m) the beam diameter is about the size of the aperture (lens). For distances greater than 1 m, you can estimate the beam diameter using this equation:

Distance/100 = beam diameter at that distance (in whatever units you measured the distance).

The actual spread is ~6 milli-radians or ~1/2 degree.

How do distance, target size, aspect, and reflectivity effect returned signal strength?

The device transmits a focused infrared beam that reflects off of a target, and a portion of that reflected signal returns to the receiver. The distance is calculated by taking the difference between the moment of signal transmission to the moment of signal reception. Successfully receiving a reflected signal is heavily influenced by several factors. These factors include:

- Target Distance

  The relationship of distance (D) to returned signal strength is an inverse square. So, with increase in distance, returned signal strength decreases by 1/D^2 or the square root of the distance.

- Target Size

  The relationship of a target's Cross Section (C) to returned signal strength is an inverse power of four. The device transmits a focused near-infrared laser beam that spreads at a rate of approximately 0.5º as distance increases. Up to 1 m it is approximately the size of the lens. Beyond 1 m, the approximate beam spread in degrees can be estimated by dividing the distance by 100, or ~8 miliradians. When the beam overfills (is larger than) the target, the signal returned decreases by 1/C^4 or the fourth root of the target's cross section.

- Aspect

  The aspect of the target, or its orientation to the sensor, affects the observable cross section and, therefore, the amount of returned signal decreases as the aspect of the target varies from the normal.

- Reflectivity

  Reflectivity characteristics of the target's surface also affect the amount of returned signal. In this case, we concern ourselves with reflectivity of near infrared wavelengths ("How does the device work with reflective surfaces?", page 12).

In summary, a small target can be very difficult to detect if it is distant, poorly reflective, and its aspect is away from the normal. In such cases, the returned signal strength may be improved by attaching infrared reflectors to the target, increasing the size of the target, modifying its aspect, or reducing distance from the sensor.

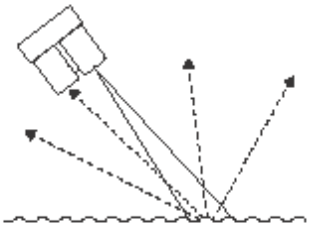
How does the device work with reflective surfaces?

Reflective characteristics of an object's surface can be divided into three categories (in the real world, a combination of characteristics is typically present):

- Diffuse Reflective
- Specular
- Retro-reflective

Diffuse Reflective Surfaces

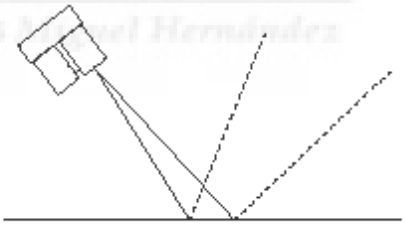
Purely diffuse surfaces are found on materials that have a textured quality that causes reflected energy to disperse uniformly. This tendency results in a relatively predictable percentage of the dispersed laser energy finding its way back to the device. As a result, these materials tend to read very well.



Materials that fall into this category are paper, matte walls, and granite. It is important to note that materials that fit into this category due to observed reflection at visible light wavelengths may exhibit unexpected results in other wavelengths. The near infrared range used by the device may detect them as nearly identical. For example, a black sheet of paper may reflect a nearly identical percentage of the infrared signal back to the receiver as a white sheet.

Specular Surfaces

Specular surfaces, are found on materials that have a smooth quality that reflect energy instead of dispersing it. It is difficult or impossible for the device to recognize the distance of many specular surfaces. Reflections off of specular surfaces tend to reflect with little dispersion which causes the reflected beam to remain small and, if not reflected directly back to the receiver, to miss the receiver altogether. The device may fail to detect a specular object in front of it unless viewed from the normal.



Examples of specular surfaces are mirrors and glass viewed off-axis.

How does liquid affect the signal?

There are a few considerations to take into account if your application requires measuring distances to, or within, liquid:

- Reflectivity and other characteristics of the liquid itself
- Reflectivity characteristics of particles suspended in the liquid
- Turbidity
- Refractive characteristics of the liquid

Reflectivity of the liquid is important when measuring distance to the surface of a liquid or if measuring through liquid to the bottom of a container("How does the device work with reflective surfaces?", page 12).

It is important to note that measuring distance with the device depends on reflected energy from the transmitted signal being detected by the receiver in the sensor. For that reason, the surface condition of the liquid may play an important role in the overall reflectivity and detectability of the liquid. In the case of a flat, highly reflective liquid surface, the laser's reflected energy may not disperse adequately to allow detection unless viewed from the normal. By contrast, small surface ripples may create enough dispersion of the reflected energy to allow detection of the liquid without the need to position the sensor so that the transmitted beam strikes the liquid's surface from the normal.

Reflectivity of suspended particles is a characteristic that may help or hinder depending on the application.

Turbidity, or the clarity of a liquid created by the presence or absence of suspended particles, can similarly help or hinder measurement efforts. If the application requires detecting the surface of the liquid, then suspended particles may help by reflecting more of the transmitted beam back to the receiver, increasing detectability and permitting measurements to be taken.

It is important to note that, attempting to measure through suspended particles in a liquid will only be successful if the transmitted beam is allowed to reflect off of the desired target without first being absorbed or reflected by the suspended particles.

When the near infrared energy transmitted by the device transitions from the atmosphere to a liquid, the energy may be bent, or refracted, and absorbed in addition to being dispersed. The degree to which the transmitted beam is refracted and absorbed is defined by its refraction index. That being said, the most important criteria impacting successful measurement through a liquid is the amount of dispersion of the transmitted beam and whether any of the dispersed beam makes its way back to the receiver on the device.

Remember that electromagnetic energy travels slower through a liquid and may affect accuracy of the final measurement output.

For the latest free software updates (excluding map data) throughout the life of your Garmin products, visit the Garmin Web site at www.garmin.com.

GARMIN.

Garmin International, Inc.
1200 East 151st Street, Olathe, Kansas 66062, USA

Garmin (Europe) Ltd.
Liberty House, Hounsdown Business Park, Southampton, Hampshire, SO40 9LR UK

Garmin Corporation
No. 68, Zhangshu 2nd Road, Xizhi Dist., New Taipei City, 221, Taiwan (R.O.C.)

www.garmin.com

September 2016 190-02088-00_0A Printed in Taiwan