

Instituto Tecnológico de Celaya



Sistemas Programables

Robot explorador “Seeker”

Integrantes:
Cid Soto Pablo Emilio
García Escoto Julio César
Rico Gómez Eduardo Daniel
Rincón Laguna María Lizeth

Martes 19 de enero de 2021

ÍNDICE

ANTECEDENTES	5
DESCRIPCIÓN DEL PROYECTO	5
PROBLEMÁTICA	5
ALCANCES	5
LIMITACIONES	6
OBJETIVOS	6
Objetivo general	6
Objetivos específicos	6
HIPÓTESIS	6
JUSTIFICACIÓN DEL PROYECTO	6
MARCO TEÓRICO	7
Software	7
Arduino IDE	7
Processing	8
SolidWorks	9
Ultimaker Cura	9
Lenguajes de programación	10
Wiring (Arduino).	10
Lenguaje C/C++	10
Processing	11
Módulos y componentes	12
Arduino Mega 2560	12
Bateria 9.6v 1600-2000mAh genérica	13
Regulador Voltaje LM2596	14
Motores DC 12V	14
Puente H L298n	15
Cámara WIFI ML8-004	15
Motor A Pasos 5v 28byj-48	16
Driver Uln2003	16
Módulo ESP-01 ESP8266 WiFi-Serial	17
Sensor DHT11	17
Sensor MQ-9	18
Módulo VI53I0x	18
Sensor Ultrasonico Hc-sr04	19
METODOLOGÍA EXPERIMENTAL	19
CRONOGRAMA	20

PRESUPUESTO	20
ESTRATEGIAS DE COMPORTAMIENTO DEL PROTOTIPO	21
Modo de comportamiento	21
Errores	22
Obtención de mediciones	22
COMPONENTES ADQUIRIDOS	24
Material	24
ANÁLISIS Y PRUEBAS DEL SOFTWARE	27
Librerías	27
Stepper.h library	27
LiquidCrystal	27
Librería ESP8266 WiFi	29
Librería DHT11	30
Biblioteca del VL53L0X	32
Ejemplos de códigos	35
Manejo de motor a pasos 28BYJ-48 con driver UNL2003	35
Medición con sensor ultrasónico	37
Creación de servidor TCP en ESP8266	39
Conexión como cliente a servidor TCP desde processing	43
Medición de temperatura y humedad con DHT11	44
Medición del nivel de monóxido de carbono y gases inflamables	46
Medición de distancia con VL53L0X	47
DISEÑO DEL PROTOTIPO	49
Boceto base	49
Diseño del chasis del robot	50
Diseño de los engranajes de las orugas	50
Diseño de los eslabones de la oruga	51
Diseño base de motores	52
Diseño brida para motor	52
Diseño final	53
INTERFAZ DE USUARIO	53
Código fuente	54
Arduino	54
ESP	59
Processing	62
Pruebas de armado y operación	69
Armado y prueba de ESP	69
Conexión de los componentes suministrados por la batería	70
Interfaz funcional con los componentes	70

Armado motores a paso junto con la cámara	71
Impresión y armado del prototipo	71
PROSPECCIÓN COMERCIAL	74
AJUSTES FINALES	76
TRABAJOS FUTUROS	76
VIDEOS DEL PROTOTIPO	77
BIBLIOGRAFÍA	81

DEFINICIÓN DEL PROYECTO

ANTECEDENTES

Los robots exploradores son los dispositivos robotizados que han sido creados con el fin de reconocer y explorar un lugar o terreno siendo capaces de moverse de forma autónoma o controlados por personas a control remoto. Su objetivo es evitar poner en riesgo la vida de los humanos, ya sea debido a que el lugar es inaccesible o porque se encuentra en una zona contaminada.

Tienen como finalidad hacer reconocimiento allí en donde el hombre no puede llegar por ser una zona inaccesible o porque supondría un peligro para la salud. También son utilizados en lugares de difícil acceso, a donde sí que podría llegar una persona solo empleando más tiempo y recursos económicos.

Los robots exploradores disponen de cámaras integradas que pueden sacar fotografías y videos, así como poder rastrear la zona por sistemas infrarrojos, etc... Esa información es retransmitida y enviada en tiempo real a la base de control.

(ROBOT EXPLORADOR, 2019)

DESCRIPCIÓN DEL PROYECTO

El proyecto a desarrollar consiste en el diseño, elaboración e implementación de un prototipo físico de un robot explorador mediante arduino y el uso de diferentes sensores para su manejo remoto así como cámara para poder visualizar lo que está viendo el robot y poder realizar un mejor manejo del mismo.

El prototipo podrá realizar la siguientes funciones:

- Movimientos controlables desde la computadora por medio de Wi-Fi
- Cámara que permita la visualización de la vista del robot desde el computador
- Brazos movibles
- Compartimiento para almacenar objetos

PROBLEMÁTICA

Muchas veces los humanos no podemos acceder a lugares de alto riesgo, como lo son edificios derrumbados, zonas de terreno accidentado, o zonas con algún tipo de restricción de acceso como lo puede ser zonas de alta radiación, o algún tipo de restricción sanitaria.

Para ello es necesario contar con equipo especializado para su exploración o tomar algún tipo de muestras que puedan almacenar, sin contaminar las evidencias y posteriormente se puedan analizar.

ALCANCES

El proyecto se desarrollará hasta una implementación física incluyendo las características mencionadas en la descripción del proyecto y en los objetivos. Además del prototipo físico se desarrollará una aplicación por medio de Processing con la cual se podrá manejar a distancia el robot por medio de la computadora, así como ver la cámara que estará incluida dentro del prototipo. A través de este proyecto, se busca brindar una alternativa en la

exploración de lugares peligrosos, para los seres humanos con el fin de prevenir situaciones de riesgo en las personas.

LIMITACIONES

1. Tiempo de construcción del prototipo que comprende 4 meses, a partir del mes de Octubre del presente año, a Enero del próximo año.
2. Debido a la contingencia sanitaria actual, se presenta como una dificultad, el realizar reuniones constantes para el desarrollo del proyecto.
3. La implementación de las pruebas puede que se realice únicamente en terrenos irregulares y no a muy altas temperaturas.

OBJETIVOS

Objetivo general

Elaborar un robot explorador creado con el fin de reconocer y explorar un lugar o terreno no apto para el ingreso de personas (condiciones físicas peligrosas) siendo capaces de moverse controlado por personas a control remoto.

Objetivos específicos

- Movimientos controlables desde la computadora por medio de Wi-Fi para mandos en todas las direcciones: adelante, atrás, izquierda y derecha.
- Cámara que permitirá la visualización de la vista del ambiente del robot desde el ordenador, por lo cual se podrá guiar mejor sus movimientos, medirá las distancias que hay con objetos cercanos y en las zonas laterales.
- Permitirá sentir la temperatura y humedad del ambiente, gases como el monóxido de carbono.
- Brazos móviles.
- Compartimiento para almacenar objetos pequeños como muestras.

HIPÓTESIS

El problema específico, base del desarrollo del presente proyecto, es la falta de un robot explorador para ingresar en áreas o terrenos, donde pueda haber potencial peligro para los seres humanos o sea de difícil acceso para ellos, sin embargo con ello, también surgen otras interrogantes cómo: ¿Podrá el robot explorador trabajar correctamente a altas temperaturas?, ¿Estará capacitado para recorrer caminos irregulares?, ¿Será posible la visualización de los ambientes por donde circule el robot?

Primeramente se deberán realizar las pruebas pertinentes para constatar cuál será la temperatura máxima a la que podrá ser sometido de tal manera que proporcione los resultados satisfactorios que se esperan. También se pretende dotar al robot de un sistema que le permita movilizarse en este tipo de caminos. Además se pretende que a través de una cámara inalámbrica le permita visualizar el área.

JUSTIFICACIÓN DEL PROYECTO

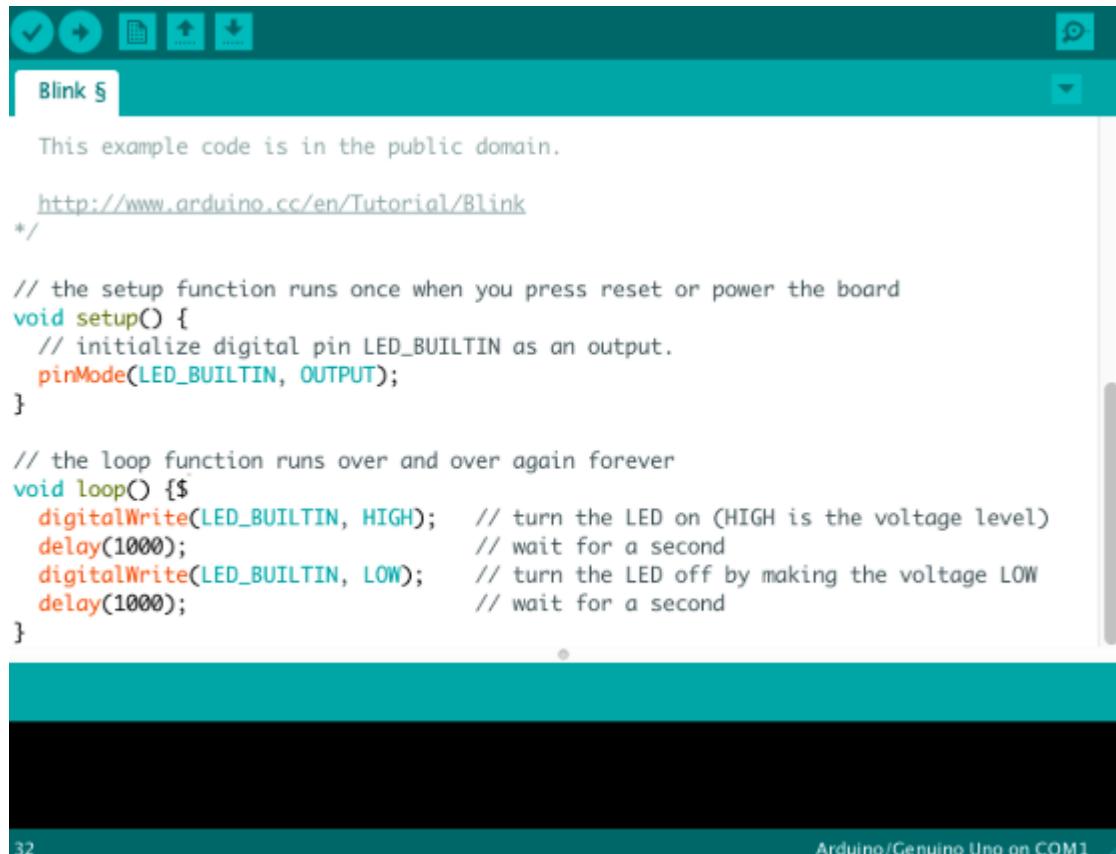
Este proyecto se basa en la necesidad de diseñar e implementar un robot explorador que pueda ingresar en áreas con potencial peligro para las personas o difícil acceso o

estructuras inestables con el fin de explorar este tipo de terrenos y tomar muestras de estos, alimentándose si en un espacio adecuado dentro del prototipo. Además que sirven como ayuda prioritaria dentro del desarrollo de conocimientos, ya que la robótica encierra muchos tópicos y diferentes campos de aplicación.

MARCO TEÓRICO

Software

Arduino IDE

A screenshot of the Arduino IDE interface. The title bar says "Blink §". The code editor contains the "Blink" sketch. The code is as follows:

```
This example code is in the public domain.  
http://www.arduino.cc/en/Tutorial/Blink  
*/  
  
// the setup function runs once when you press reset or power the board  
void setup() {  
  // initialize digital pin LED_BUILTIN as an output.  
  pinMode(LED_BUILTIN, OUTPUT);  
}  
  
// the loop function runs over and over again forever  
void loop() {  
  digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)  
  delay(1000); // wait for a second  
  digitalWrite(LED_BUILTIN, LOW); // turn the LED off by making the voltage LOW  
  delay(1000); // wait for a second  
}
```

The status bar at the bottom shows "32" on the left and "Arduino/Genuino Uno on COM1" on the right.

Imagen 1. Arduino IDE ([Wikipedia, n.d.](#))

El IDE de Arduino es un Entorno de Desarrollo Integrado que puedes instalar en tu computadora. Utilizaremos este software para crear, editar y cargar tus sketch o códigos de tu proyecto a tu placa Arduino. La descarga es de aproximadamente 150 MB, es compatible con los sistemas operativos Windows, Mac y Linux. ([Wikipedia, n.d.](#))

Processing



Imagen 2. Processing (Inicio Processing Qué es Processing, 2016)
Processing es el conjunto de un lenguaje de programación, fundamentalmente orientado a objetos, basado en Java, por una parte, y el IDE correspondiente para desarrollar aplicaciones en dicho lenguaje. Del mismo modo que Arduino tiene su propio IDE, Processing también.

Las características principales de Processing son:

- Todo el entorno es gratuito y open source (cómo debe ser). Así que, de gastos, nada.
- Programas interactivos que permiten generar modelos en 2D, en 3D e, incluso, en documentos PDF.
- Integración con OpenGL para la aceleración 2D y 3D.
- Disponible para Linux, Windows y Mac OS.
- Más de cien librerías para ampliar las funcionalidades que ofrece el núcleo.
- Gran cantidad de documentación y libros.
- Nos permite realizar la conexión con Arduino mediante el puerto serial. (*Inicio Processing Qué es Processing, 2016*)

SolidWorks



Imagen 3. SolidWorks (Wikipedia, 2020)

SolidWorks es un software CAD (diseño asistido por computadora) para modelado mecánico en 2D y 3D, desarrollado en la actualidad por SolidWorks Corp.

El programa permite modelar piezas y conjuntos y extraer de ellos tanto planos técnicos como otro tipo de información necesaria para la producción. Es un programa que funciona con base en las nuevas técnicas de modelado con sistemas CAD. El proceso consiste en traspasar la idea mental del diseñador al sistema CAD, "construyendo virtualmente" la pieza o conjunto. Posteriormente todas las extracciones (planos y ficheros de intercambio) se realizan de manera bastante automatizada. ([Wikipedia, 2020](#)).

Ultimaker Cura

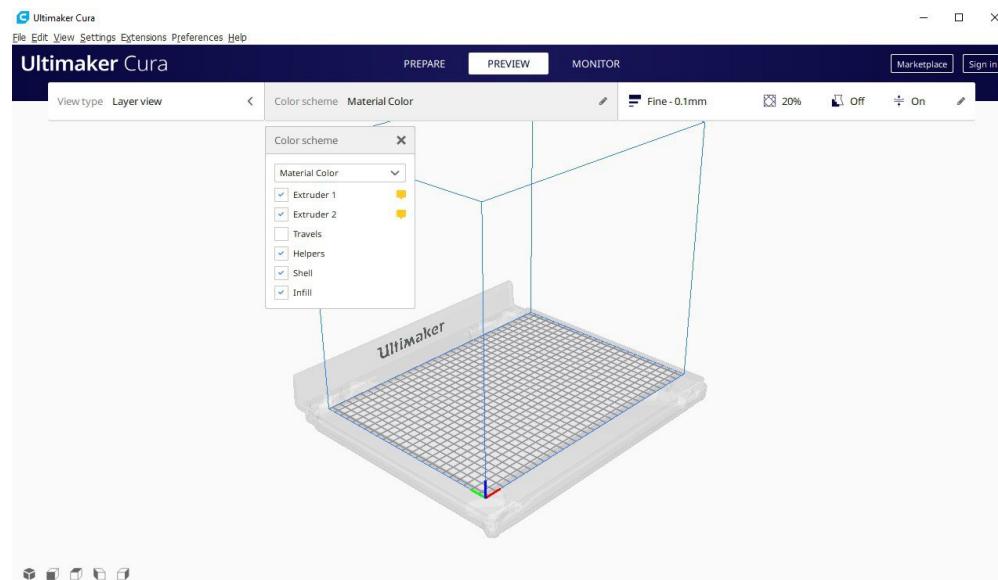
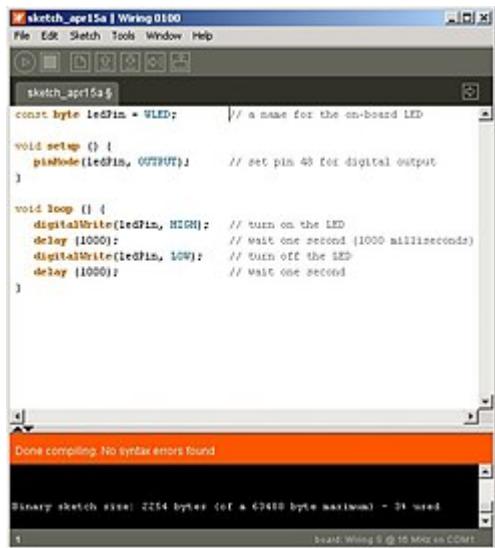


Imagen 4. Ultimaker Cura(Wikipedia, 2020)

Es una aplicación diseñada para impresoras 3D, en la que se pueden modificar los parámetros de impresión y después transformarlos a código G. Fue creada por David Braan, que después de un tiempo trabajaría para Ultimaker, una empresa dedicada al diseño y fabricación de impresoras 3D. ([Wikipedia, 2020](#)).

Lenguajes de programación

Wiring (Arduino).



```
sketch_apr15a | Wiring 0100
File Edit Sketch Tools Window Help
sketch_apr15a
const byte ledPin = 13; // a name for the on-board LED

void setup () {
  pinMode(ledPin, OUTPUT); // set pin 13 for digital output
}

void loop () {
  digitalWrite(ledPin, HIGH); // turn on the LED
  delay (1000); // wait one second (1000 milliseconds)
  digitalWrite(ledPin, LOW); // turn off the LED
  delay (1000); // wait one second
}
```

Done compiling. 0 syntax errors found.

Binary sketch size: 2254 bytes (of a 64480 byte maximum) - 3% used

build: Wiring 0 @ 16 MHz on CH3

Imagen 5. Wiring([Wikipedia, 2020](#))

Wiring es una plataforma de prototipado electrónico de fuente abierta compuesta de un lenguaje de programación, un entorno de desarrollo integrado (IDE), y un microcontrolador. Wiring está basado en Processing. Wiring permite escribir software para controlar dispositivos conectados a la tarjeta electrónica para crear toda clase de objetos interactivos, espacios o experiencias físicas que sienten y responden al mundo físico.

El IDE de Wiring viene con una librería de C/C++ llamada "Wiring", la cual hace operaciones comunes de input/output mucho más fáciles. Los programas de Wiring están escritos en C/C++, pese a que sus usuarios solo necesiten definir dos funciones para hacer un programa ejecutable:

- `setup()` – una función ejecutada solo una vez en el inicio de un programa la cual puede ser usada para definir los ajustes iniciales de un entorno.
- `loop()` – una función llamada repetidamente hasta que la tarjeta es apagada.

([Wikipedia, 2020](#))

Lenguaje C/C++

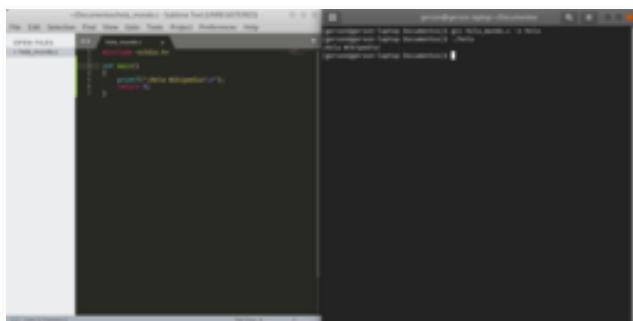
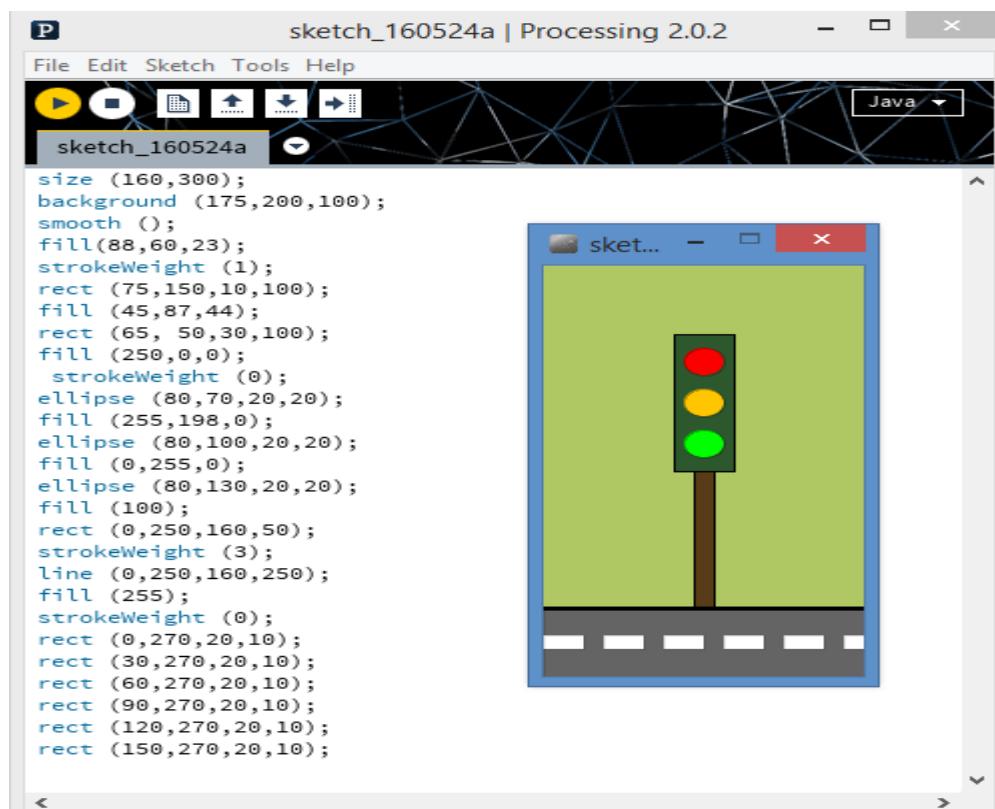


Imagen 6. Código simple en C compilado sobre GCC ([Wikipedia, 2020](#))

C es un lenguaje de programación de propósito general originalmente desarrollado por Dennis Ritchie entre 1969 y 1972 en los Laboratorios Bell. Se trata de un lenguaje de tipos de datos estáticos, débilmente tipificado, de medio nivel, ya que dispone de las estructuras típicas de los lenguajes de alto nivel pero, a su vez, dispone de construcciones del lenguaje que permiten un control a muy bajo nivel. Los compiladores suelen ofrecer extensiones al lenguaje que posibilitan mezclar código en ensamblador con código C o acceder directamente a memoria o dispositivos periféricos.

Se emplea en plataformas de robótica como Arduino. Con C se aprende a hacer cosas como depurar código, o a gestionar la memoria.

Processing



```
size (160,300);
background (175,200,100);
smooth ();
fill(88,60,23);
strokeWeight (1);
rect (75,150,10,100);
fill (45,87,44);
rect (65, 50,30,100);
fill (250,0,0);
strokeWeight (0);
ellipse (80,70,20,20);
fill (255,198,0);
ellipse (80,100,20,20);
fill (0,255,0);
ellipse (80,130,20,20);
fill (100);
rect (0,250,160,50);
strokeWeight (3);
line (0,250,160,250);
fill (255);
strokeWeight (0);
rect (0,270,20,10);
rect (30,270,20,10);
rect (60,270,20,10);
rect (90,270,20,10);
rect (120,270,20,10);
rect (150,270,20,10);
```

Imagen 7. Código en Processing para la elaboración de un semáforo (Gómez, 2016)

Processing es un lenguaje de programación y entorno de desarrollo integrado de código abierto basado en Java, de fácil utilización, y que sirve como medio para la enseñanza y producción de proyectos multimedia e interactivos de diseño digital.

Uno de los objetivos declarados de Processing es el de actuar como herramienta para que artistas, diseñadores visuales y miembros de otras comunidades ajenas al lenguaje de la programación, aprendieran las bases de la misma a través de una muestra gráfica instantánea y visual de la información. El lenguaje de Processing se basa en Java, aunque hace uso de una sintaxis simplificada y de un modelo de programación de gráficos. ([Wikipedia, 2020](#))

Módulos y componentes

Arduino Mega 2560



Imagen 8. Arduino Mega 2560 ([Demeyer, 2018](#))

Cuenta con una velocidad de sincronización de 16 MHz, así como 256 KB de flash, 8 KB de SRAM y 4 KB de EEPROM. Debido a su robusto procesador, el Mega cuenta con 54 pines de E/S digitales (15 de los cuales pueden ofrecer una salida de modulación por ancho de pulsos [PWM]) y 16 pines analógicos. ([Demeyer, 2018](#))

Bateria 9.6v 1600-2000mAh genérica



Imagen 9. Bateria MELASTRA (Aliexpress, 2020)

Es un conjunto de baterías de 1.5v agrupadas en serie y paralelo (Configuracion mixta para obtener un voltaje de salida de 9.6v y una carga de entre 1600 y 2000 mAh dependiendo del modelo y marca)

Regulador Voltaje LM2596

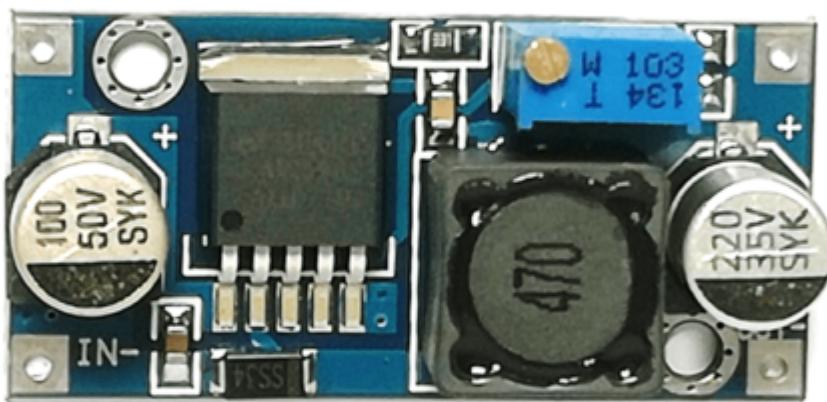


Imagen 10. Regulador Voltaje (LM2596, n.d.)

Dispositivo electrónico que tiene la capacidad de regular o disminuir el voltaje de entrada de un circuito a partir de una fuente de alimentación con un voltaje mayor. Este módulo tiene un trimpot multivuelta (potenciómetro) para ajustar el voltaje de salida. Dado que el trimpot tiene 25 vueltas de ajuste, puede ajustar fácilmente la salida del módulo exactamente al voltaje que necesites. (**CDMX Electrónica, 2020**)

Motores DC 12V



Imagen 11. Motores DC 12V (electrocrea, 2020)

El motor de corriente continua (denominado también motor de corriente directa, motor CC o motor DC) es una máquina que convierte la energía eléctrica en mecánica, provocando un movimiento rotatorio, gracias a la acción que se genera del campo magnético. (**electrocrea, 2020**)

Puente H L298n

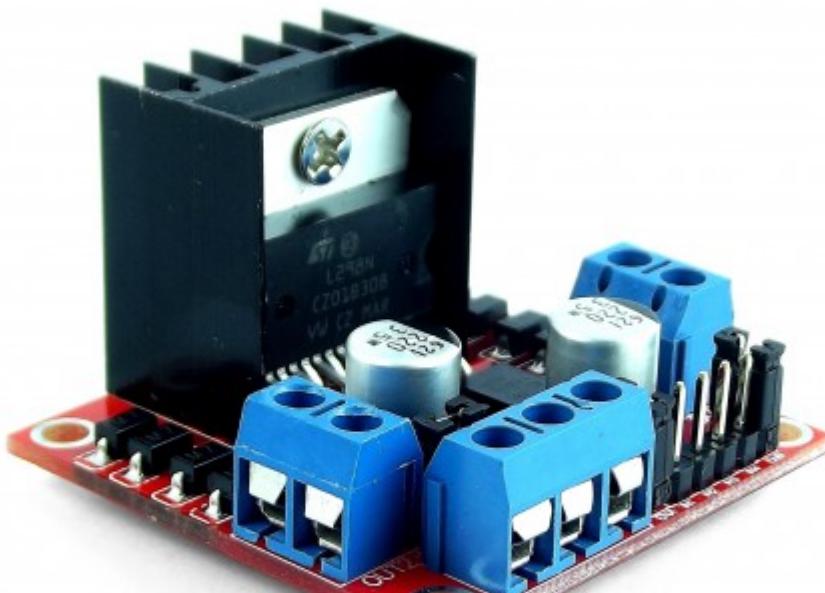


Imagen 12. Puente H L298n ([naylampmechatronics, 2020](#))

El driver puente H L298N es el módulo más utilizado para manejar motores DC de hasta 2 amperios. El chip L298N internamente posee dos puentes H completos que permiten controlar 2 motores DC o un motor paso a paso bipolar/unipolar.

El módulo permite controlar el sentido y velocidad de giro de motores mediante señales TTL que se pueden obtener de microcontroladores y tarjetas de desarrollo como Arduino, Raspberry Pi o Launchpads de Texas Instruments. El control del sentido de giro se realiza mediante dos pines para cada motor, la velocidad de giro se puede regular haciendo uso de modulación por ancho de pulso (PWM por sus siglas en inglés).

Tiene integrado un regulador de voltaje LM7805 de 5V encargado de alimentar la parte lógica del L298N, el uso de este regulador se hace a través de un Jumper y se puede usar para alimentar la etapa de control. ([naylampmechatronics, 2020](#))

Cámara WIFI ML8-004



Imagen 13. Cámara WIFI ([Mercado Libre, 2020](#))

Cámara 2mp 1080p Wifi Inalámbrica, Visión Nocturna, detección de movimiento, alerta, alarma.

Cámara de seguridad, le brinda en tiempo real lo que sucede dentro del entorno, puede

enviar a través de la aplicación de su celular voz a la cámara, gracias a que cuenta con una bocina dentro de ella que le permite interactuar, además de contar con sensor de movimiento y alarma que le enviara una alerta a su dispositivo móvil para que vea lo que sucede en tiempo real. (**Mercado Libre, 2020**)

Motor A Pasos 5v 28byj-48



Imagen 14. Motor A Pasos (sandorobotics, 2020)

El 28BYJ-48 es un pequeño motor paso a paso unipolar de bajo precio. Las características eléctricas del 28BYJ-48 son modestas (trabaja a 5V), pero incorpora un reductor integrado que lo convierte en un componente mucho más útil e interesante al trabajar con un paso de 5.625 grados (64 pasos por vuelta). (**sandorobotics, 2020**)

Driver ULN2003

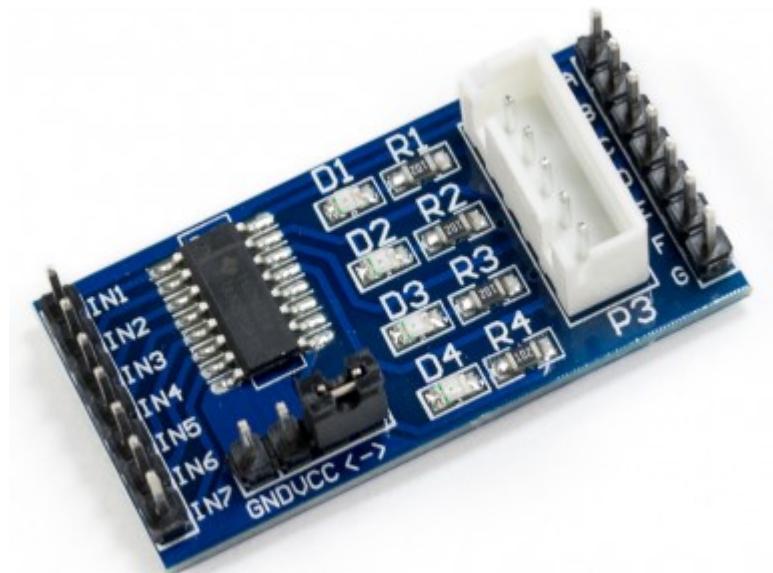
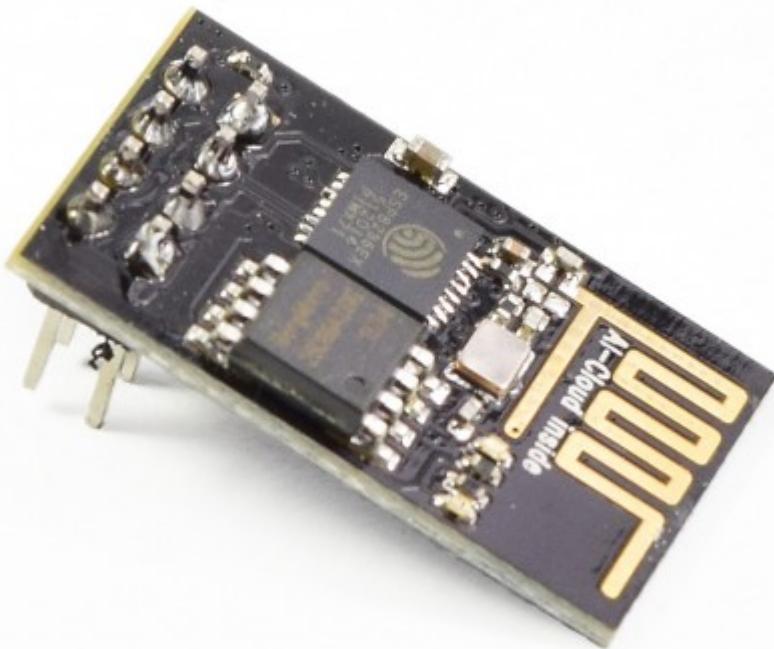


Imagen 15. Drive ULN2003 (naylampmechatronics, 2020)

El módulo ULN2003 es un driver especialmente diseñado para manejar el motor a pasos 28BYJ-48 (unipolar). Adicionalmente puede ser usado para manejar Relays, motores DC o cualquier carga DC de bajo consumo de corriente. Internamente posee un arreglo de 7 transistores NPN Darlington con diodos de protección para

cargas inductivas. Cada transistor o canal es capaz de manejar una carga de hasta 500mA, se pueden utilizar canales en paralelo y así aumentar la capacidad de corriente. (**naylampmechatronics, 2020**)

Módulo ESP-01 ESP8266 WiFi-Serial



*Imagen 16. Módulo ESP-01 (**naylampmechatronics, 2020**)*

Está basado en el SoC (System on Chip) ESP8266, un chip altamente integrado, diseñado para las necesidades de un mundo conectado. Integra un potente procesador con arquitectura de 32 bits y conectividad WiFi. Ofrece una completa y autocontenido solución WiFi Networking, permitiéndole trabajar como host de aplicaciones o reducir la carga de WiFi Networking de otro procesador. A nivel de conectividad el módulo puede trabajar en 2 modos: como estación WiFi (WiFi Station) o como Punto de Acceso (Access Point). (**naylampmechatronics, 2020**)

Sensor DHT11

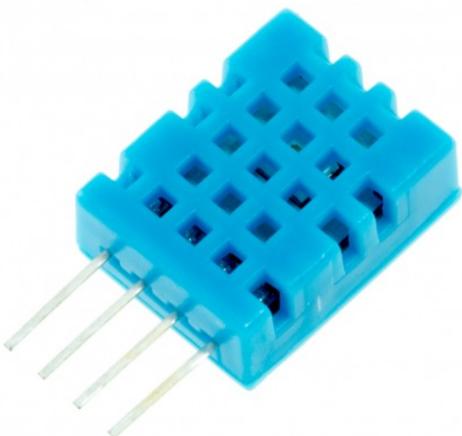


Imagen 17. Sensor DHT11 (naylampmechatronics, 2020)

El DHT11 es un sensor digital de temperatura y humedad relativa de bajo costo y fácil uso. Integra un sensor capacitivo de humedad y un termistor para medir el aire circundante, y muestra los datos mediante una señal digital en el pin de datos (no posee salida analógica). Utilizado en aplicaciones académicas relacionadas al control automático de temperatura, aire acondicionado, monitoreo ambiental en agricultura y más. (naylampmechatronics, 2020)

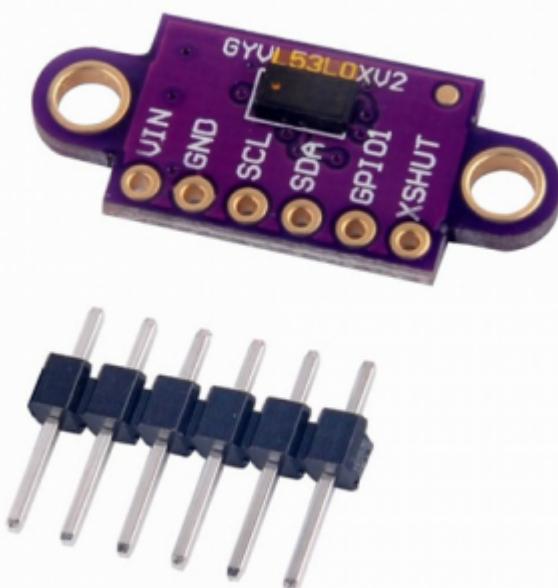
Sensor MQ-9



Imagen 18. Sensor MQ-9 (geekfactory, 2020)

El módulo MQ-9 Sensor de Gas es un dispositivo semiconductor que posee alta sensibilidad para detectar una concentración de Monóxido de Carbono y gas combustible; en cuanto el sensor detecta alguna de estas señales emitirá una señal analógica variable dependiendo la concentración de gas detectada. (geekfactory, 2020)

Módulo VL53L0x



www.electronicaplugandplay.com

Imagen 19. Módulo VL53L0x (electronicaplugandplay, 2020)

Es un sensor de distancia de nueva generación capaz de detectar el "tiempo de vuelo" y ofrece mediciones exactas sin importar la superficie reflectante, con un rango de alcance de hasta 2 metros. Este módulo posee un emisor láser que periódicamente emite un haz de

luz, la cual es reflejada al encontrar algún obstáculo, el tiempo transcurrido entre la emisión y la detección del haz de luz, es medido por el sensor y de esta manera establece la distancia a la que se encuentra el objeto. Este dispositivo tiene una salida digital a través de un puerto I2C. ([electronicaplugandplay, 2020](#))

Sensor Ultrasonico Hc-sr04



Imagen 20. Sensor Ultrasónico ([naylampmechatronics, 2020](#))

El sensor HC-SR04 posee dos transductores: un emisor y un receptor piezoeléctricos, además de la electrónica necesaria para su operación. El funcionamiento del sensor es el siguiente: el emisor piezoeléctrico emite 8 pulsos de ultrasonido(40KHz) luego de recibir la orden en el pin TRIG, las ondas de sonido viajan en el aire y rebota al encontrar un objeto, el sonido de rebote es detectado por el receptor piezoeléctrico, luego el pin ECHO cambia a Alto (5V) por un tiempo igual al que demoró la onda desde que fue emitida hasta que fue detectada, el tiempo del pulso ECO es medido por el microcontrolador y así se puede calcular la distancia al objeto. El funcionamiento del sensor no se ve afectado por la luz solar o material de color negro (aunque los materiales blandos acústicamente como tela o lana pueden llegar a ser difíciles de detectar). ([naylampmechatronics, 2020](#))

METODOLOGÍA EXPERIMENTAL

El método experimental, también conocido como científico-experimental, se caracteriza porque permite que el investigador manipula y controla las variables de una investigación tanto como pueda, con la intención de estudiar las relaciones que existen entre estas con las bases del método científico.

Se trata de un proceso que se utiliza para investigar fenómenos, adquirir nuevos conocimientos o corregir e integrar conocimientos previos. Se utiliza en la investigación científica y se basa en la observación sistemática, la toma de mediciones, la experimentación, la formulación de pruebas y la modificación de hipótesis. ([Rodríguez, 2020](#))

Se pretende aplicar esta metodología con el objetivo de reducir tiempo del proyecto para que sea alcanzable y facilitar la comunicación y colaboración de los integrantes.

El proyecto se divide en tres fases:

1. Diseño y fabricación
2. Control
3. Pruebas

Lo principal a realizar es modelar el comportamiento cinemático estructural del robot, para de ahí, partir a diseñar y construir los sistemas que conforman al robot. De aquí se procede a realizar el diseño del control.

Posteriormente se procederá a la fabricación y ensamblaje, para proceder con las pruebas, es decir, el comportamiento real del prototipo.

CRONOGRAMA

	Octubre	Noviembre	Diciembre	Enero
Definición del problema				
Antecedentes del proyecto				
Presupuesto de material a usar				
diseño del prototipo				
Recolección de material				
Construcción del prototipo				
Pintado del prototipo				
Programación				
Pruebas				
Presentación				

PRESUPUESTO

Productos ya en posesión: \$precio -> \$final

DESCRIPCIÓN	CANTIDAD	VALOR
Arduino Mega 2560	1	\$276 -> \$0
Bateria 9.6v 1600-2000mAh	1	\$300 -> \$0
Regulador Voltaje con LM2596	1	\$179
Motores DC 12V	2	\$345 -> \$0
Puente H L298n	1	\$50
Cámara WIFI ML8-004	1	\$315
Motor A Pasos 5v 28byj-48	6	\$150->(1)\$125
Driver Uln2003	6	\$150->(1)\$125
Modulo Wifi Serial Esp8266	1	\$50
Sensor DHT11	1	\$38
Sensor Mq-9	1	\$50
Módulo VI53l0x	1	\$149
Sensor Ultrasonico Hc-sr04	4	\$128->(2)\$64
1KG Filamento PLA	1	\$499
Jumpers Dupont H-h, M-m, H-m	120	\$93
Rollo soldadura de estaño	1	\$178->\$0
Cautin	1	\$150->\$0
Impresora 3D	1	\$3360->\$0
SolidWorks Student Edition	1	\$2090->\$0
Tornillería M3	1	\$80
	TOTAL	\$1817

ESTRATEGIAS DE COMPORTAMIENTO DEL PROTOTIPO

Modo de comportamiento

- El prototipo se controlará de manera remota por una computadora a través de una conexión Wi-Fi privada utilizando un servidor TCP hosteado por el módulo ESP8266.

- El prototipo enviará la imagen de la cámara a la computadora por medio de Wi-Fi utilizando la misma red ya creada anteriormente.
- Se moverá gracias a dos motores 12V DC por medio de un diseño similar al tractor oruga anclado al chasis principal.
- El prototipo contará con un sistema de inclinación vertical así como un giro horizontal en la cámara para poder ajustar la visión, este sistema se compondrá de dos motores a pasos
- El sensor de distancia láser, estará alineado con la vista de la cámara de tal manera que se pueda medir la distancia de los objetos hasta 2 metros de distancia.
- Si se quiere tomar un objeto con el brazo, el sensor determinará si el objeto está a la distancia adecuada para tomar el objeto, de no ser así, se le indicará al operario que avance o retroceda el robot.
- El sensor de temperatura trabajará de manera permanente para mostrarnos la temperatura ambiente en tiempo real en donde se encuentra el prototipo y esta temperatura se mostrará en la computadora para mantener el prototipo en una temperatura segura para mantener su funcionalidad adecuada y evitar su deterioro.
- El sensor de gas, nos mostrará la cantidad de monóxido de carbono y gas combustible que se encuentre en el ambiente para así mostrarlo en la computadora donde se controla para poder determinar si es posible que una persona pueda entrar sin intoxicarse en ese ambiente.
- Las señales recibidas por los sensores así como los botones para el control de la vista y movimiento del robot se graficarán en una interfaz de control diseñada en processing que enviará peticiones al servidor del módulo wifi en función de la acción que se desee hacer o la lectura del sensor que se desea obtener.
- El módulo Wi-Fi actuará como intermediario entre la interfaz de control y el microcontrolador arduino por medio de una conexión serial.
- El movimiento del brazo será posible gracias a los motores a pasos, se contará con una interfaz adjunta a la interfaz de control del robot donde podremos ver la posición del brazo y podremos mover cada articulación.

Errores

- Si el robot se encuentra en una posición en la que sea imposible moverse, enviará un mensaje al usuario que lo controla para notificarlo de la situación.
- Al desconectarse involuntariamente un componente, en la computadora del usuario controlador se notificará lo sucedido.
- Cuando el nivel de batería sea excesivamente bajo (si no se atiende la alerta de batería baja) se enviará un mensaje al operario con las últimas lecturas de los sensores seguido de esto entrará en un modo de recuperación donde emitirá un sonido para permitir su localización, en este modo no es posible controlar el brazo ni realizar las mediciones de los sensores, pues se debe ahorrar la mayor cantidad posible de energía.

Obtención de mediciones

- Los resultados que se reportarán serán los niveles de monóxido de carbono y gas combustible del ambiente en el que se encuentre, así como también reportes de temperatura para monitorear y asegurar el buen funcionamiento del prototipo.

- Se enviarán reportes de las fallas que este tenga, como en el caso de los errores, al desconectarse algún componente o cuando el prototipo se encuentre en un terreno de difícil acceso.
- Cuando el nivel de batería sea demasiado bajo se enviará una alerta al operario.

COMPONENTES ADQUIRIDOS

A continuación se muestra evidencia del 100% de los componentes y material adquiridos a ser utilizados en el desarrollo del prototipo.

Material



Imagen 22. [Autoría propia,2020]



Imagen 26. Motor a pasos [Autoría propia,2020]



Imagen 23. Cámara Wifi [Autoría propia,2020]

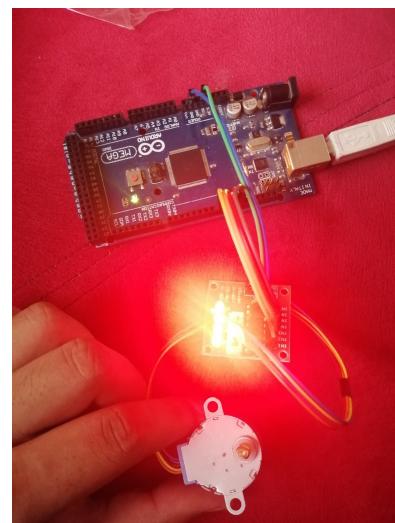


Imagen 27. Motor a pasos Prueba [Autoría propia,2020]

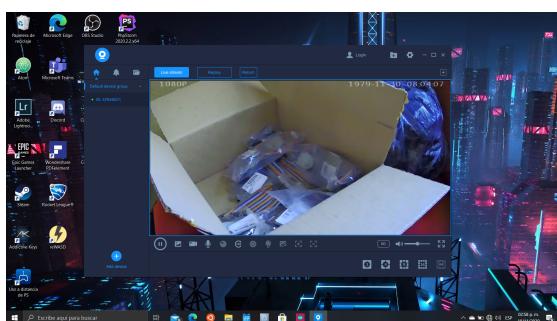


Imagen 24. Cámara Wifi Prueba [Autoría propia,2020]

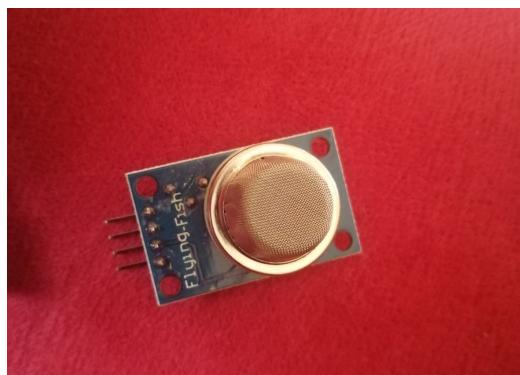


Imagen 28. **Sensor de gas** [Autoría propia,2020]



Imagen 31. **Módulo WiFi Serial Esp8266** [Autoría propia,2020]



Imagen 29. **Regulador Voltaje** [Autoría propia,2020]

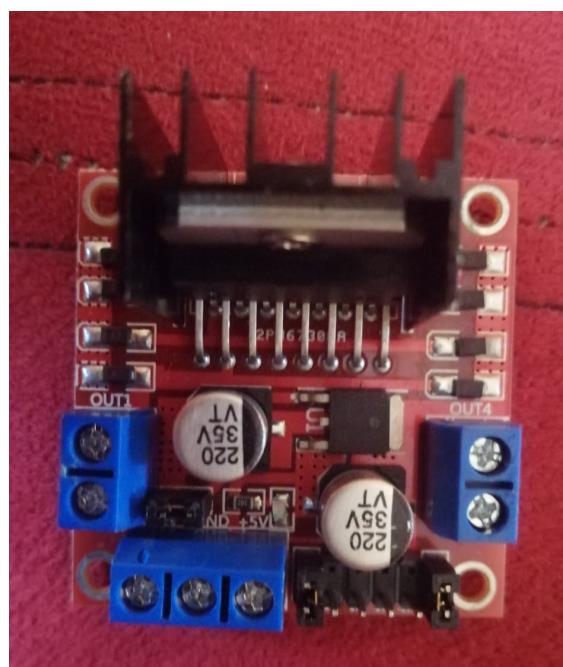


Imagen 32. **Puente H L298n** [Autoría propia,2020]



Imagen 30. **Módulo VL53L0x (sensor distancia)** [Autoría propia,2020]



Imagen 33. **Sensor Ultrasonico Hc-sr04** [Autoría propia,2020]



Imagen 34. Arduino Mega 2560 [Autoría propia,2020]



Imagen 37. Filamento PLA para la impresora 3D [Autoría propia,2020]

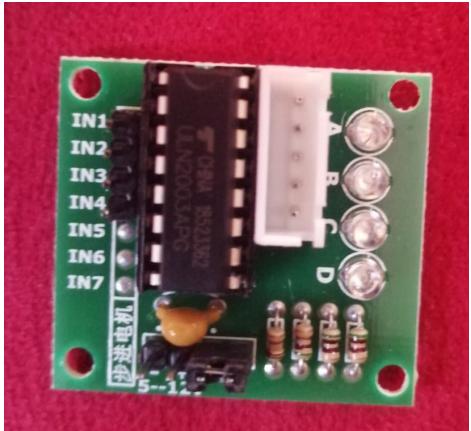


Imagen 35. Driver Uln2003 [Autoría propia,2020]



Imagen 38. Impresora 3D [Autoría propia,2020]



Imagen 36. Motor 12V [Autoría propia,2020]

ANÁLISIS Y PRUEBAS DEL SOFTWARE

Se realizó una investigación sobre algunos códigos de ejemplo y librerías como posibilidades para emplearlos en el desarrollo del prototipo.

Librerías

Stepper.h library

Esta librería le permite controlar motores paso a paso unipolares o bipolares.

```
#include <Stepper.h>
```

FUNCIONES

```
Stepper(steps, pin1, pin2);  
Stepper(steps, pin1, pin2, pin3, pin4);
```

Descripción:

Esta función crea una nueva instancia de la clase Stepper que representa un motor paso a paso, conectado a la placa Arduino. Utilícelo en la parte superior de su programa, por encima de setup () y loop (). El número de parámetros depende de cómo haya conectado el cable a su motor - ya sea utilizando dos o cuatro pines de la placa Arduino.

Parámetros:

steps: el número de pasos en una revolución de su motor. Si su motor da el número de grados por paso, dividir ese número por 360 para obtener el número de pasos (por ejemplo 360 / 3.6 da 100 pasos). (int)

pin 1, pin 2: dos pines que están conectados al motor (int)

pin3, pin4: opcional los últimos dos pines unidos al motor, si está conectado a cuatro pines(int)

Retornos:

Una nueva instancia de la clase Stepper

Ejemplo:

```
Stepper myStepper = Stepper(100, 5, 6);
```

LiquidCrystal

Se contempla la posible utilización de esta librería para mostrar alguna información importante en un display LCD si fuera necesario.

La librería LiquidCrystal nos permite crear un objeto que representa al display LCD y que contiene todas las operaciones “de bajo nivel” para que a nosotros nos resulte fácil la programación de este dispositivo.

Es el constructor de la clase LyquidCrystal. Permite crear un objeto de esta clase, que se usará para gestionar el display LCD. Como argumentos recibe una serie de números que se refieren a pines concretos de la placa Arduino, conectados a diferentes pines del display.

```
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
```

Los dos primeros números (el 12 y el 11) se refieren a los pines conectados a los puntos RS y E del display. Los cuatro últimos números se refieren a los pines de D4 a D7 del bus de datos del display. En general, esta configuración es la más simple y típica. No obstante, este constructor admite otras listas de argumentos, como podemos ver a continuación:

```
LiquidCrystal(lcd, RS, E, D4, D5, D6, D7);
LiquidCrystal(lcd, RW, E, D4, D5, D6, D7);
LiquidCrystal(lcd, RS, E, D0, D1, D2, D3, D4, D5, D6, D7);
LiquidCrystal(lcd, RW, E, D0, D1, D2, D3, D4, D5, D6, D7);
```

De lo que se trata, como podemos deducir de las líneas de arriba, es que podemos especificar los pines de Arduino que se conectan a distintos terminales del display. En el ejemplo del artículo 12 sólo necesitábamos los pines de la primera configuración del constructor. En otros montajes, o con otros modelos de displays, necesitaremos otras configuraciones.

EL MÉTODO begin()

Como hemos comentado durante la descripción del primer sketch, este método es necesario para inicializar el display. Recibe dos argumentos: el primero es la anchura en caracteres y el segundo la altura (número de filas) del display. También posiciona el cursor en el primer carácter de la primera fila.

EL MÉTODO clear()

Limpia el display y posiciona el cursor en el primer carácter de la primera fila.

EL MÉTODO home()

Sitúa el cursor en el primer carácter de la primera fila, sin borrar el display.

EL MÉTODO setCursor()

Posiciona el cursor en una ubicación específica del display. Recibe dos argumentos. El primero se refiere al carácter de la fila y el segundo a la fila. Ten en cuenta que tanto los caracteres como las filas se empiezan a contar desde 0, no desde 1.

EL MÉTODO write()

Escribe una cadena (bien sea un objeto de tipo String o una matriz de caracteres) en el display. Como argumento recibe la cadena a mostrar.

EL MÉTODO print()

Actúa de un modo similar a write(), pero con la posibilidad de enviar directamente números enteros al display, en distintas bases de numeración.

LOS MÉTODOS cursor() y noCursor()

Estos métodos no reciben argumentos. El primero se emplea para que el cursor sea visible en el display como un signo de subrayado. El segundo método hace el cursor invisible (es el estado por defecto).

LOS MÉTODOS blink() y noBlink()

Cuando el cursor está visible por haber usado el método cursor(), podemos hacer que aparezca parpadeante usando el método blink(). Si usamos el método noBlink() el cursor no parpadeará.

LOS MÉTODOS display() y noDisplay()

Si usamos el método noDisplay() el display se apagará. Podemos reactivarlo con el método display(). Estos métodos no reciben argumentos.

LOS MÉTODOS autoscroll() y noAutoscroll()

Cuando se completa el espacio del display destinado a mostrar texto o datos, lo que “sobra” no se verá (aparece truncado). Si activamos el método autoscroll(), el contenido más antiguo se desplazará, desapareciendo por el principio de la línea, para dejar sitio al final de la misma para los nuevos contenidos. Con el método noAutoscroll() restablecemos el comportamiento por defecto.

LOS MÉTODOS scrollDisplayLeft() y scrollDisplayRight()

Cuando usamos estos métodos, el contenido se desplaza un carácter a la izquierda o a la derecha. Además, según el que empleemos, si activamos el método autoscroll() los contenidos se desplazarán en el sentido indicado.

LOS MÉTODOS leftToRight() y rightToLeft()

Establecen el sentido de la escritura, por si el mensaje se envía en idiomas, como el árabe, que se escriben “al revés” que el nuestro.

Librería ESP8266 WiFi

La librería WiFi para ESP8266 ha sido desarrollada basándose en el SDK de ESP8266, usando nombres convencionales y la filosofía de funcionalidades generales de la librería WiFi de Arduino. Con el tiempo, la riqueza de las funciones WiFi del SDK de ESP8266 pasadas a ESP8266/Arduino superan a la librería WiFi de Arduino y se hizo evidente que tenemos que proporcionar documentación por separado sobre lo que es nuevo y extra.

Para conectar el módulo ESP al WiFi (como conectar un teléfono móvil a un punto caliente), solo necesita un par de líneas de código:

```
#include <ESP8266WiFi.h>

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

```

Serial.println();

WiFi.begin("nombre-red", "contraseña-red");

Serial.print("Conectando");
while (WiFi.status() != WL_CONNECTED)
{
    delay(500);
    Serial.print(".");
}
Serial.println();

Serial.print("Conectado, dirección IP: ");
Serial.println(WiFi.localIP());

void loop() {}

```

En la línea WiFi.begin("nombre-red", "contraseña-red") reemplace nombre-red y contraseña-red con el nombre y contraseña a la red WiFi que quiere conectarse. Entonces suba el sketch al módulo ESP y abra el Monitor Serie. Deberías ver algo como:

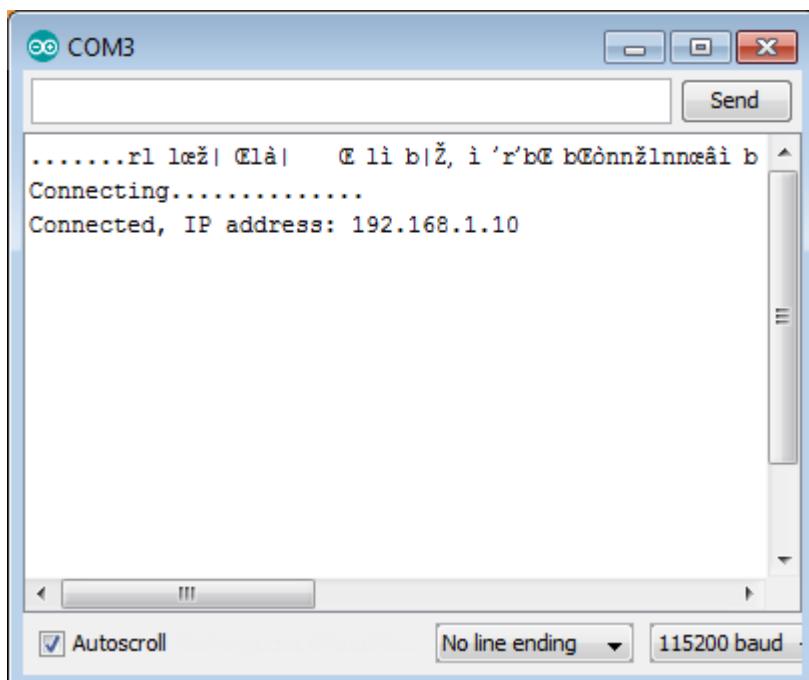


Imagen 39. (Arduino Core, 2020)

Librería DHT11

Esta librería nos permitirá trabajar con los sensores de temperatura a utilizar en nuestro prototipo.

Los sensores DHT11 y DHT22 son capaces de realizar mediciones simultáneas de humedad relativa y temperatura, entregándonos su lectura de forma digital. Podemos encontrar aplicaciones para el DHT11 o el DHT22 en el control de invernaderos, monitoreo de centros de datos, climatización de casas y edificios, etc.

```
#include <DHT.h>
// Definimos el pin digital donde se conecta el sensor
#define DHTPIN 2
// Dependiendo del tipo de sensor
#define DHTTYPE DHT11
// Inicializamos el sensor DHT11
DHT dht(DHTPIN, DHTTYPE);
```

Función setup

```
void setup() {
    // Inicializamos comunicación serie
    Serial.begin(9600);
    // Comenzamos el sensor DHT
    dht.begin();
}
```

Función Loop

```
void loop() {
    // Esperamos 5 segundos entre medidas
    delay(5000);
    // Leemos la humedad relativa
    float h = dht.readHumidity();
    // Leemos la temperatura en grados centígrados (por defecto)
    float t = dht.readTemperature();
    // Leemos la temperatura en grados Fahrenheit
    float f = dht.readTemperature(true);
    // Comprobamos si ha habido algún error en la lectura
    if (isnan(h) || isnan(t) || isnan(f)) {
        Serial.println("Error obteniendo los datos del sensor DHT11");
        return;
    }
    // Calcular el índice de calor en Fahrenheit
    float hif = dht.computeHeatIndex(f, h);
    // Calcular el índice de calor en grados centígrados
    float hic = dht.computeHeatIndex(t, h, false);
    Serial.print("Humedad: ");
    Serial.print(h);
    Serial.print(" %\t");
    Serial.print("Temperatura: ");
    Serial.print(t);
    Serial.print(" *C ");
    Serial.print(f);
    Serial.print(" *F\t");
    Serial.print("Índice de calor: ");
    Serial.print(hic);
    Serial.print(" *C ");
    Serial.print(hif);
    Serial.println(" *F");
```

}

Biblioteca del VL53L0X

API VL53L0X de ST y esta biblioteca

La mayor parte de la funcionalidad de esta biblioteca se basa en la API VL53L0X proporcionada por ST (STSW-IMG005), y algunos de los comentarios explicativos en el código se citan o parafrasean del código fuente de la API, el manual del usuario de la API (UM2039) y el Hoja de datos de VL53L0X. Para obtener más explicaciones sobre el código de la biblioteca y cómo se derivó de la API, consulte los comentarios en VL53L0X.cpp.

Esta biblioteca está destinada a proporcionar una manera más rápida y sencilla de comenzar a usar el VL53L0X con un controlador compatible con Arduino, en contraste con la personalización y compilación de la API de ST para Arduino. La biblioteca tiene una interfaz más optimizada, así como un menor espacio de almacenamiento y memoria. Sin embargo, no implementa algunas de las funciones más avanzadas disponibles en la API (por ejemplo, calibrar el sensor para que funcione bien debajo de un cubreobjetos) y tiene una verificación de errores menos sólida. Para aplicaciones avanzadas, especialmente cuando el almacenamiento y la memoria son un problema menor, considere usar la API VL53L0X directamente.

VL53L0X ()

Constructor.

setBus void (bus TwoWire *)

Configura este objeto para utilizar el bus I²C especificado. bus debe ser un puntero a un objeto TwoWire; el bus predeterminado es Wire, que suele ser el primer o único bus I²C en un Arduino. Si su Arduino tiene más de un bus I²C y tiene el VL53L0X conectado al segundo bus, que normalmente se llama Wire1, puede llamar a sensor.setBus (& Wire1);.

TwoWire * getBus ()

Devuelve un puntero al bus I²C que utiliza este objeto.

setAddress vacío (uint8_t new_addr)

Cambia la dirección del dispositivo esclavo I²C del VL53L0X al valor dado (7 bits).

uint8_t getAddress ()

Devuelve la dirección I²C que utiliza este objeto.

bool init (bool io_2v8 = verdadero)

Inicializa y configura el sensor. Si el argumento opcional io_2v8 es verdadero (el valor predeterminado si no se especifica), el sensor está configurado para el modo 2V8 (E / S de 2,8 V); si es falso, el sensor se deja en modo 1V8. El valor de retorno es un booleano que indica si la inicialización se completó correctamente.

```
void writeReg (uint8_t reg, uint8_t valor)
```

Escribe un registro de sensor de 8 bits con el valor dado.

Las constantes de dirección de registro se definen mediante el tipo de enumeración regAddr en VL53L0X.h.

Ejemplo de uso: sensor.writeReg (VL53L0X :: SYSRANGE_START, 0x01);

```
void writeReg16Bit (uint8_t reg, uint16_t valor)
```

Escribe un registro de sensor de 16 bits con el valor dado.

```
void writeReg32Bit (uint8_t reg, uint32_t valor)
```

Escribe un registro de sensor de 32 bits con el valor dado.

```
uint8_t readReg (uint8_t reg)
```

Lee un registro de sensor de 8 bits y devuelve el valor leído.

```
uint16_t readReg16Bit (uint8_t reg)
```

Lee un registro de sensor de 16 bits y devuelve el valor leído.

```
uint32_t readReg32Bit (uint8_t reg)
```

Lee un registro de sensor de 32 bits y devuelve el valor leído.

```
void writeMulti (uint8_t reg, uint8_t const * src, uint8_t count)
```

Escribe un número arbitrario de bytes desde la matriz dada al sensor, comenzando en el registro dado.

```
void readMulti (uint8_t reg, uint8_t * dst, uint8_t count)
```

Lee un número arbitrario de bytes del sensor, comenzando en el registro dado, en la matriz dada.

```
bool setSignalRateLimit (límite flotante_Mcps)
```

Establece el límite de velocidad de la señal de retorno al valor dado en unidades de MCPS (mega recuentos por segundo). Esta es la amplitud mínima de la señal reflejada desde el objetivo y recibida por el sensor necesaria para que informe una lectura válida. Establecer un límite inferior aumenta el rango potencial del sensor, pero también aumenta la probabilidad de obtener una lectura inexacta debido a reflejos de objetos distintos del objetivo previsto. Este límite se inicializa a 0,25 MCPS de forma predeterminada. El valor de retorno es un booleano que indica si el límite solicitado era válido.

```
flotar getSignalRateLimit ()
```

Devuelve el límite de velocidad de la señal de retorno actual en MCPS.

```
bool setMeasurementTimingBudget (uint32_t presupuesto_us)
```

Establece el presupuesto de tiempo de medición al valor dado en microsegundos. Este es el tiempo permitido para una medición de rango; un presupuesto de tiempo más largo permite mediciones más precisas. El presupuesto predeterminado es de aproximadamente 33000

microsegundos o 33 ms; el mínimo es 20 ms. El valor de retorno es un booleano que indica si el presupuesto solicitado era válido.

`uint32_t getMeasurementTimingBudget ()`

Devuelve el presupuesto de tiempo de medición actual en microsegundos.

`bool setVcselPulsePeriod (tipo vcselPeriodType, uint8_t period_pclks)` Establece el período de pulso VCSEL (láser de emisión de superficie de cavidad vertical) para el tipo de período dado (VL53L0X :: VcselPeriodPreRange o VL53L0X :: VcselPeriod) en PCLKFinals valor. Los períodos más largos aumentan el rango potencial del sensor. Los valores válidos son (solo números pares):

Pre: 12 a 18 (initializado a 14 por defecto)

Final: 8 a 14 (initializado a 10 por defecto)

El valor de retorno es un booleano que indica si el período solicitado fue válido.

`uint8_t getVcselPulsePeriod (tipo vcselPeriodType)`

Devuelve el período de pulso VCSEL actual para el tipo de período dado.

`void startContinuous (uint32_t period_ms = 0)`

Inicia mediciones de rango continuas. Si el argumento opcional period_ms es 0 (el valor predeterminado si no se especifica), se usa el modo continuo adosado (el sensor toma medidas con la mayor frecuencia posible); si es distinto de cero, se usa el modo temporizado continuo, con el período de inter-medición especificado en milisegundos que determina la frecuencia con la que el sensor toma una medición.

`void stopContinuous ()`

Detiene el modo continuo.

`uint16_t readRangeContinuousMillimeters ()`

Devuelve una lectura de rango en milímetros cuando el modo continuo está activo.

`uint16_t readRangeSingleMillimeters ()`

Realiza una medición de rango de un solo disparo y devuelve la lectura en milímetros.

`void setTimeout (tiempo de espera de uint16_t)`

Establece un período de tiempo de espera en milisegundos después del cual las operaciones de lectura se cancelarán si el sensor no está listo. Un valor de 0 desactiva el tiempo de espera.

`uint16_t getTimeout ()`

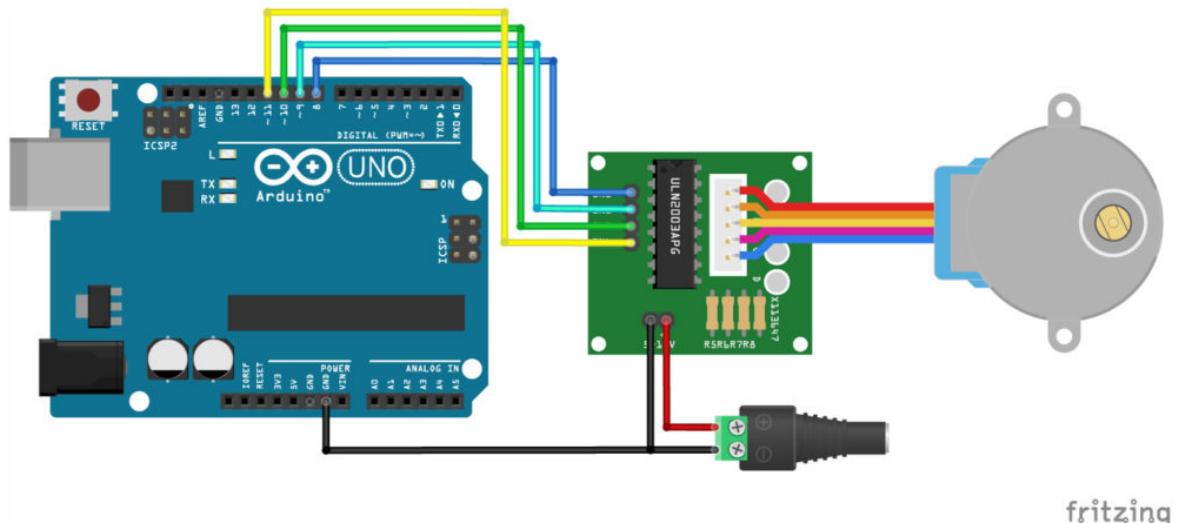
Devuelve la configuración del período de tiempo de espera actual.

`bool timeoutOccurred ()`

Indica si se ha producido un tiempo de espera de lectura desde la última llamada a timeoutOccurred () .

Ejemplos de códigos

Manejo de motor a pasos 28BYJ-48 con driver UNL2003



fritzing

Imagen 40. Manejo de motor a pasos 28BYJ-48 (Patagoniatec, 2020).

```
#include <Stepper.h> //Agrega la librería Stepper
// Se definen cuantos pasos son por revolución
const int stepsPerRevolution = 2048;

// Pin 8 al IN1 en el ULN2003
// Pin 9 al IN2 en el ULN2003
// Pin 10 al IN3 en el ULN2003
// Pin 11 al IN4 en el ULN2003
// Se crea un objeto Stepper 'myStepper', tomar en cuenta el orden:
Stepper myStepper = Stepper(stepsPerRevolution, 8, 10, 9, 11);
void setup() {
    // Poner la velocidad a 5rpm:
    myStepper.setSpeed(5);

    // Se comienza la comunicación serial a 9600 baudios:
    Serial.begin(9600);
}

void loop() {
    // Gira una revolución en sentido del reloj
    Serial.println("clockwise");
    myStepper.step(stepsPerRevolution);
    delay(1000);

    // Gira una revolución al sentido contrario del reloj
    Serial.println("counterclockwise");
```

```
myStepper.step(-stepsPerRevolution);  
delay(1000);}
```

Prueba



Imagen 41. Manejo de motor a pasos 28BYJ-48 (Autoría Propia, 2020).

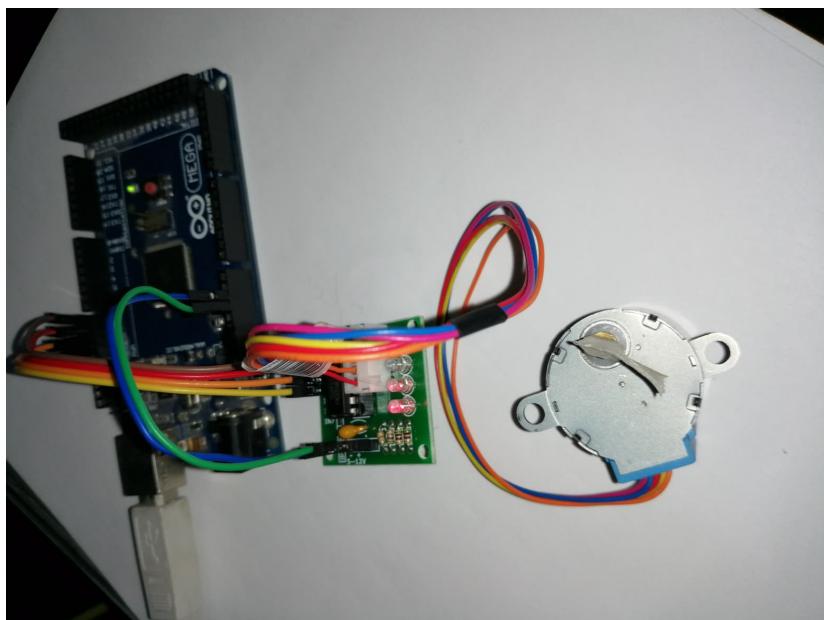


Imagen 42. Manejo de motor a pasos 28BYJ-48 (Autoría Propia, 2020).

Medición con sensor ultrasónico

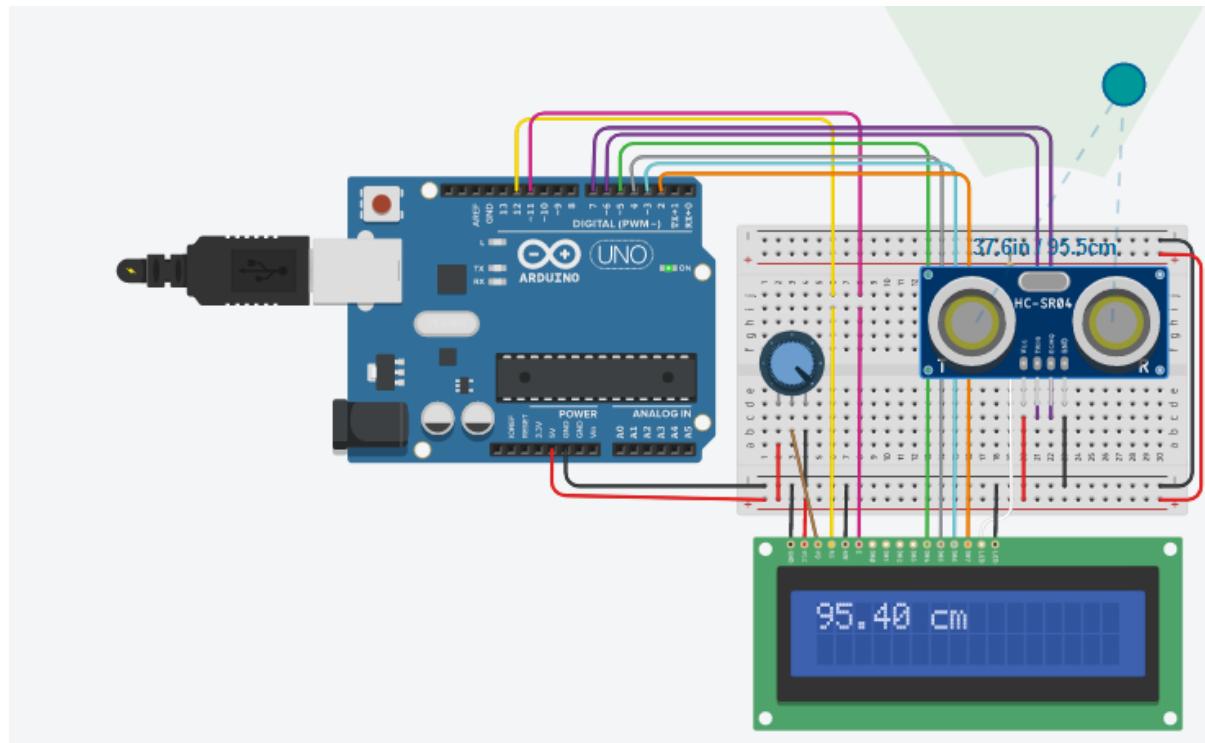


Imagen 43. Medición con sensor ultrasónico. (Autoría propia).

```
//Librería LiquidCrystal
#include <LiquidCrystal.h>

// variables
int trig=6;
int echo=7;
float tiempo,distancia;

// Inicializa la librería con los números de los pines de la interfaz.
LiquidCrystal lcd(12,11, 5, 4, 3, 2);

void setup()
{
    // Especificar el número de columnas y filas de la pantalla LCD
    lcd.begin(16, 2);
    pinMode(trig,OUTPUT);
    pinMode(echo,INPUT);
    /*Esta instrucción le indica al Arduino que inicie comunicación con la
    computadora
    con una velocidad de comunicación serial de 9600 bits por segundo */
    Serial.begin(9600);
}
```

```

void loop() {

    digitalWrite(trig,LOW);
    delayMicroseconds(2);
    //pulso de 10 microsegundos
    digitalWrite(trig,HIGH);
    delayMicroseconds(10);
    digitalWrite(trig,LOW);
    tiempo=pulseIn(echo,HIGH); //mide el ancho del echo en us
    //distancia = 340m/s x tiempo/2 porque el tiempo es ida y vuelta
    distancia=tiempo/58.2; //Distancia en cm
    Serial.println(distancia);

    lcd.setCursor(0,0);
    lcd.print(distancia);
    lcd.print(" cm");
    delay(500);
    lcd.clear();
}

}

```

Prueba

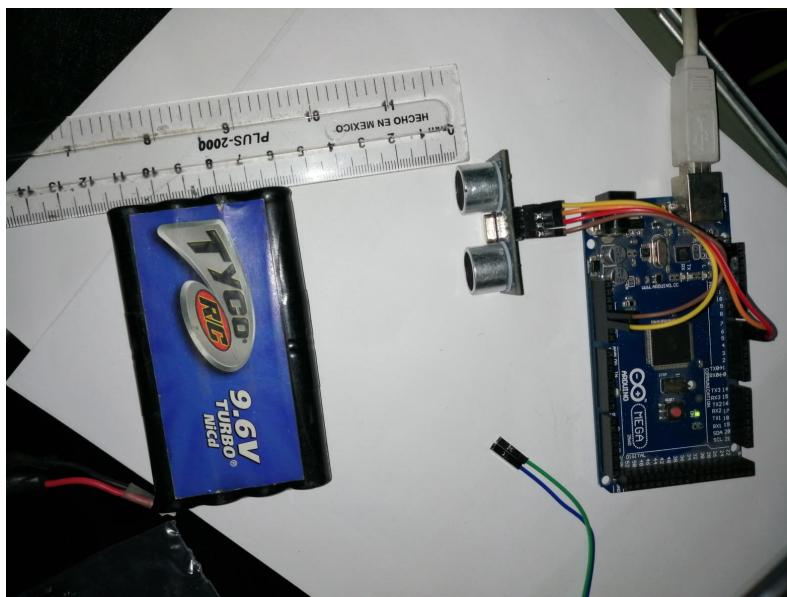


Imagen 44. Prueba sensor ultrasónico. (Autoría propia).



Imagen 45. Medición sensor ultrasónico. (Autoría propia).

Creación de servidor TCP en ESP8266

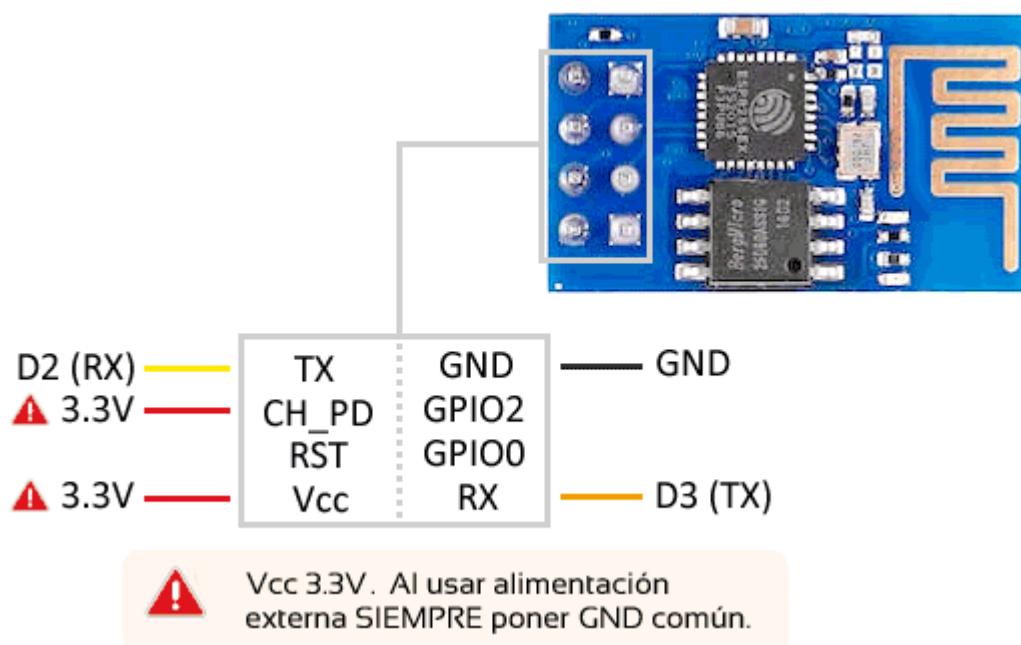


Imagen 46. Diagrama del circuito (Llamas, 2017)

```
#include "ESP8266.h"
#include <SoftwareSerial.h>

const char* SSID = "myssid";
const char* PASSWORD = "mypassword";

SoftwareSerial softSerial(2, 3); // RX, TX
ESP8266 wifi(softSerial);

void setup(void)
```

```

{
    pinMode(LED_BUILTIN, OUTPUT);

    Serial.begin(9600);
    Serial.print("setup begin\r\n");

    wifi.restart();
    delay(500);
    if (wifi.setOprToStationSoftAP()) {
        Serial.print("to station + softap ok\r\n");
    }
    else {
        Serial.print("to station + softap err\r\n");
    }

    if (wifi.joinAP(SSID, PASSWORD)) {
        Serial.print("Join AP success\r\n");
        Serial.print("IP: ");
        Serial.println(wifi.getLocalIP().c_str());
    }
    else {
        Serial.print("Join AP failure\r\n");
    }

    if (wifi.enableMUX()) {
        Serial.print("multiple ok\r\n");
    }
    else {
        Serial.print("multiple err\r\n");
    }

    if (wifi.startTCPServer(80)) {
        Serial.print("start tcp server ok\r\n");
    }
    else {
        Serial.print("start tcp server err\r\n");
    }

    if (wifi.setTCPServerTimeout(20)) {
        Serial.print("set tcp server timout 20 seconds\r\n");
    }
    else {
        Serial.print("set tcp server timout err\r\n");
    }

    Serial.println("setup end\r\n");
}

```

```

#define wifiWrite(A) wifi.send(mux_id, (uint8_t*) A, sizeof(A) - 1);
void loop(void)
{
    uint8_t buffer[128] = { 0 };
    uint8_t mux_id;

    uint32_t len = wifi.recv(&mux_id, buffer, sizeof(buffer), 100);
    if (len > 0) {
        Serial.print("Received from: ");
        Serial.print(mux_id);
        Serial.print("\r\n");

        wifiWrite("HTTP/1.1 200 OK\r\nContent-Type: /html\r\nConnection:
close\r\n\r\n");

        wifiWrite("<html>\n<head>\n<title>Luis Llamas</title>\n</head>\n<body>");
        wifiWrite("<h2>Salidas digitales</h2>");
        wifiWrite("<button onClick=location.href='./?data=0'>ON</button>");
        wifiWrite("<button onClick=location.href='./?data=1'>OFF</button>");
        wifiWrite("</body></html>");

        Serial.println("Send finish");

        for (uint32_t i = 0; i < len; i++) {
            char c = (char)buffer[i];
            if (c == '?') {
                if ((char)buffer[i + 6] == '1')
                {
                    digitalWrite(LED_BUILTIN, HIGH);
                    Serial.println("LED ON");
                }
                else
                {
                    digitalWrite(LED_BUILTIN, LOW);
                    Serial.println("LED OFF");
                }
            }

            break;
        }
    }
}

```

Prueba

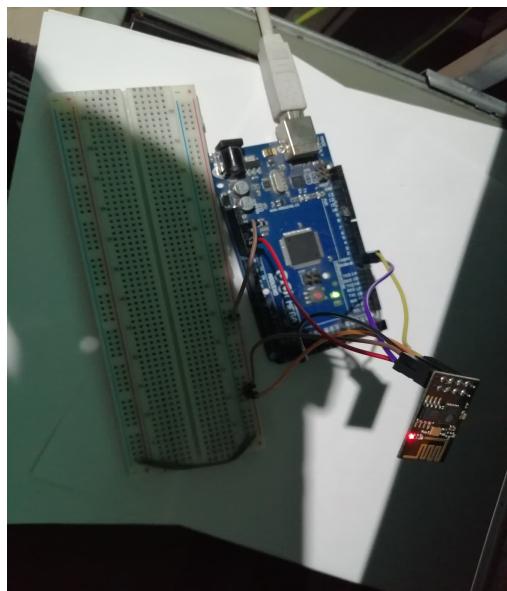


Imagen 47. Prueba sensor wifi. (Autoría propia).



Imagen 48. Red wifi creada por el sensor. (Autoría propia).

Conexión como cliente a servidor TCP desde processing

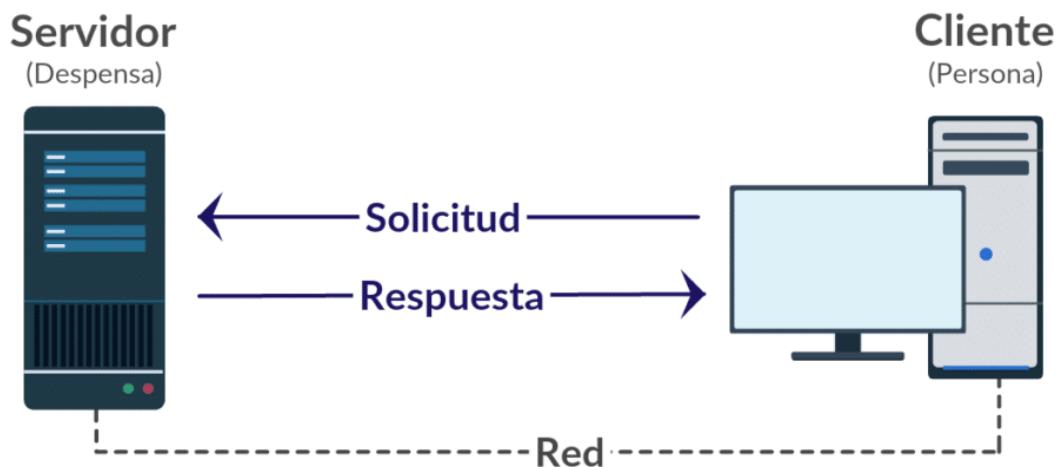


Imagen 49. (*siaguanta, 2020*)

```
import processing.net.*;  
  
Client myClient;  
int dataIn;  
  
void setup() {  
    size(200, 200);  
    // Connect to the local machine at port 5204.  
    // This example will not run if you haven't  
    // previously started a server on this port.  
    myClient = new Client(this, "127.0.0.1", 5204);  
}  
  
void draw() {  
    if (myClient.available() > 0) {  
        dataIn = myClient.read();  
    }  
    background(dataIn);  
}
```

Medición de temperatura y humedad con DHT11

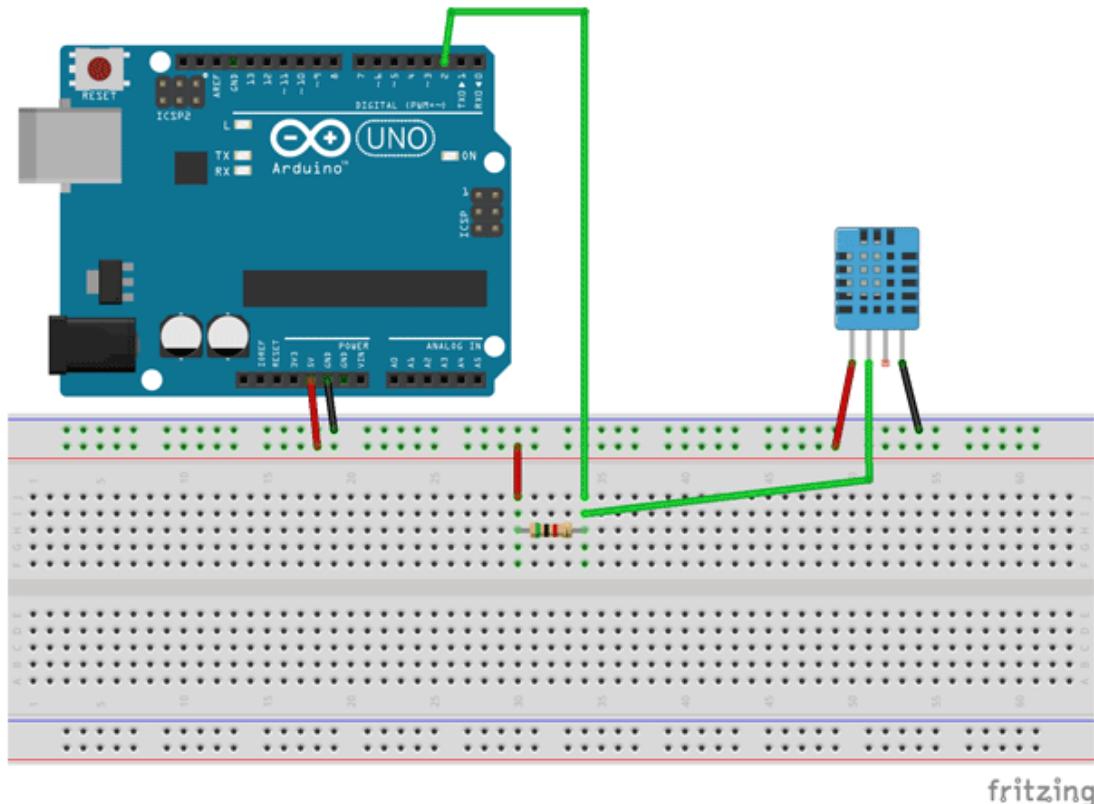


Imagen 50. Diagrama del circuito medición de temperatura y humedad (Valle Hernández, 2018)

```
// Incluimos librería
#include <DHT.h>

// Definimos el pin digital donde se conecta el sensor
#define DHTPIN 2
// Dependiendo del tipo de sensor
#define DHTTYPE DHT11

// Inicializamos el sensor DHT11
DHT dht(DHTPIN, DHTTYPE);

void setup() {
    // Inicializamos comunicación serie
    Serial.begin(9600);

    // Comenzamos el sensor DHT
    dht.begin();
}

void loop() {
```

```

    // Esperamos 5 segundos entre medidas
delay(5000);

    // Leemos la humedad relativa
float h = dht.readHumidity();
    // Leemos la temperatura en grados centígrados (por defecto)
float t = dht.readTemperature();
    // Leemos la temperatura en grados Fahrenheit
float f = dht.readTemperature(true);

    // Comprobamos si ha habido algún error en la lectura
if (isnan(h) || isnan(t) || isnan(f)) {
    Serial.println("Error obteniendo los datos del sensor DHT11");
    return;
}

    // Calcular el índice de calor en Fahrenheit
float hif = dht.computeHeatIndex(f, h);
    // Calcular el índice de calor en grados centígrados
float hic = dht.computeHeatIndex(t, h, false);

Serial.print("Humedad: ");
Serial.print(h);
Serial.print(" %\t");
Serial.print("Temperatura: ");
Serial.print(t);
Serial.print(" *C ");
Serial.print(f);
Serial.print(" *F\t");
Serial.print("Índice de calor: ");
Serial.print(hic);
Serial.print(" *C ");
Serial.print(hif);
Serial.println(" *F");

}

```

Medición del nivel de monóxido de carbono y gases inflamables

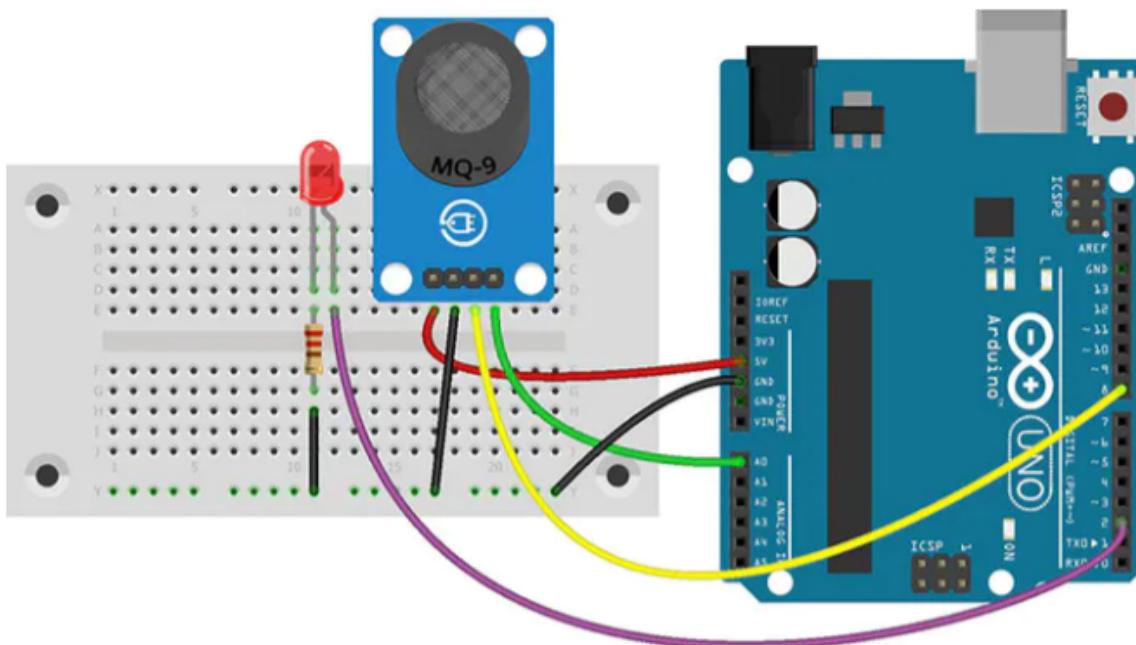


Imagen 51. Diagrama del circuito medición de nivel de monóxido de carbono y gases inflamables (project hub, 2020)

```
const int LED = 2;
const int DO = 8;
void setup() {
  Serial.begin(9600);
  pinMode(LED, OUTPUT);
  pinMode(DO, INPUT);
}
void loop() {
  int alarm = 0;
  float sensor_volt;
  float RS_gas;
  float ratio;

  float R0 = 0.91; //Asignar el valor de la calibración
  int sensorValue = analogRead(A0);
  sensor_volt = ((float)sensorValue / 1024) * 5.0;
  RS_gas = (5.0 - sensor_volt) / sensor_volt;
  ratio = RS_gas / R0; // ratio = RS/R0
  //-----
  Serial.print("sensor_volt = ");
  Serial.println(sensor_volt);
  Serial.print("RS_ratio = ");
  Serial.println(RS_gas);
  Serial.print("Rs/R0 = ");
  Serial.println(ratio);
  Serial.print("\n\n");
  alarm = digitalRead(DO);
```

```

if (alarm == 1) digitalWrite(LED, HIGH);
else if (alarm == 0) digitalWrite(LED, LOW);
delay(1000);
}

```

Medición de distancia con VL53L0X

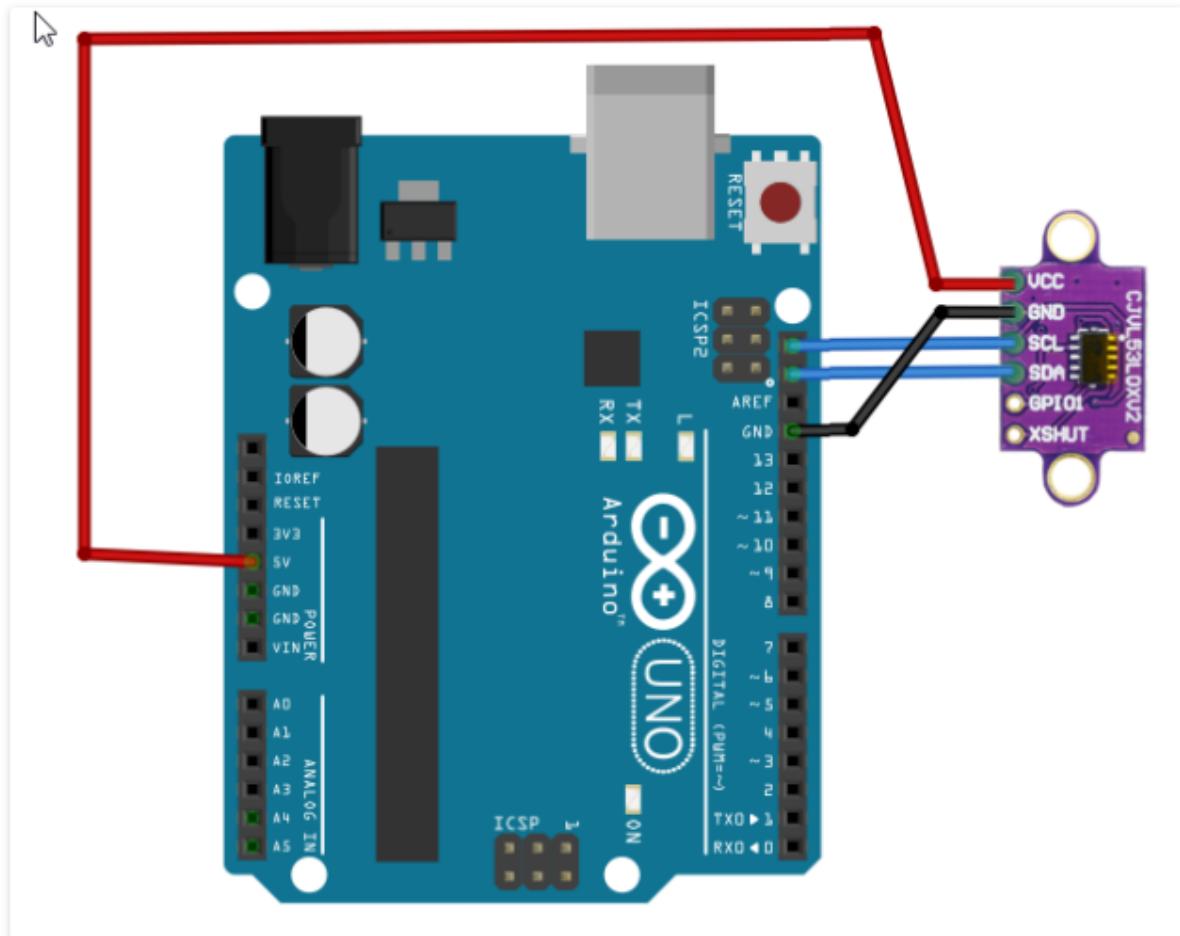


Imagen 52. Diagrama del circuito medición de distancia con VL52L0X (robots didácticos, 2020,)

```

#include <Wire.h>
#include <VL53L0X.h>
VL53L0X sensor;

int n_Samples = 5; // numero de muestras para promediar

#ifndef LONG_RANGE // Aumenta sensibilidad, +rango, -precision
#define HIGH_SPEED // Mayor velocidad, menor precision
#define HIGH_ACCURACY // Alta precision, menor velocidad

void setup() {

```

```

Serial.begin(9600);
Wire.begin();

sensor.init();
sensor.setTimeout(500);

##### Parametros Medida simple #####
#if defined LONG_RANGE
// Limite de la tasa de retorno (por defecto 0.25 MCPS)
sensor.setSignalRateLimit(0.1);
// Periodo de pulso laser (por defecto 14 y 10 PCLKs)
sensor.setVcselPulsePeriod(VL53L0X::VcselPeriodPreRange, 18);
sensor.setVcselPulsePeriod(VL53L0X::VcselPeriodFinalRange, 14);
#endif

#if defined HIGH_SPEED
// reduce tiempo estimado a 20 ms (por defecto 33 ms)
sensor.setMeasurementTimingBudget(20000);
#elif defined HIGH_ACCURACY
// incrementa tiempo estimado a 200 ms
sensor.setMeasurementTimingBudget(200000);
#endif
}

void loop() {

float DISTANCIA = getDISTANCIA (n_Samples);

if (sensor.timeoutOccurred()) {
Serial.println(" TIME OUT");
} else {
if (DISTANCIA< 2) Serial.println("Fuera de Rango (d < 2 cm)");
else if (DISTANCIA>220) Serial.println("Fuera de Rango (d > 2 m)");
else {
Serial.print(DISTANCIA, 1); // distancia en cm y 1 decimal
Serial.println(" cm");
}
}
delay (300);
}

float getDISTANCIA(int n) { // hacemos "n" mediciones

float SUMA_n = 0;
for (int i = 0; i < n; i++) {
SUMA_n += sensor.readRangeSingleMillimeters();
}
return( SUMA_n /n /10); // Promedio en centimetros
}

```

DISEÑO DEL PROTOTIPO

Boceto base

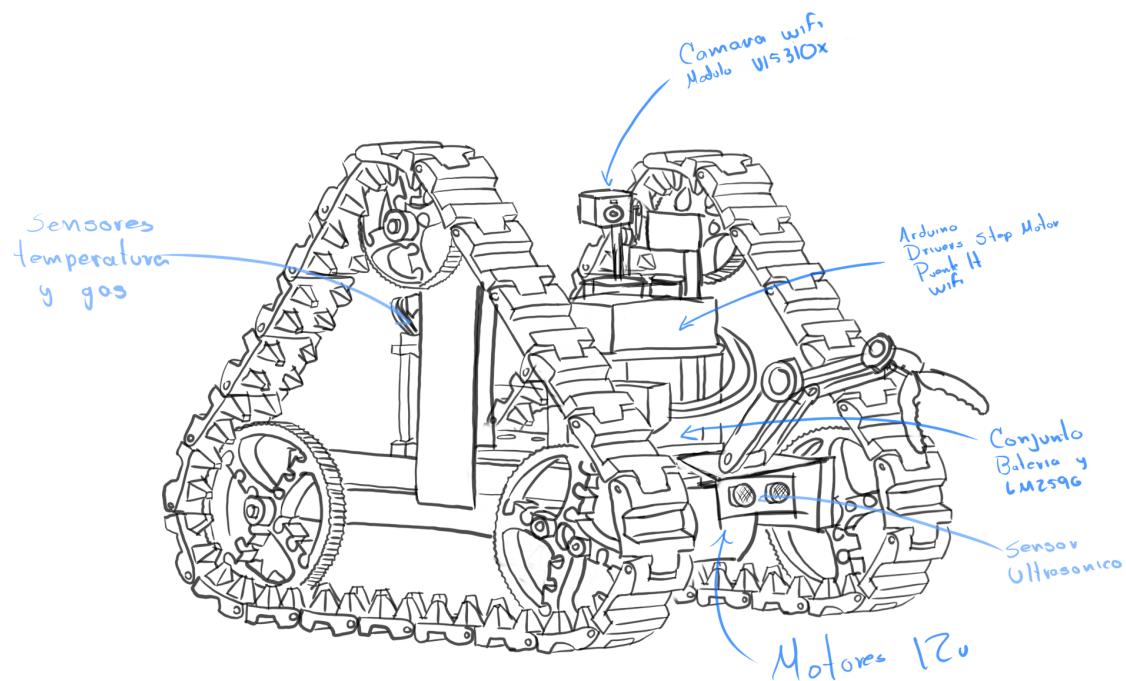


Imagen 53. Boceto base (Autoría propia, 2020)

Diseño del chasis del robot

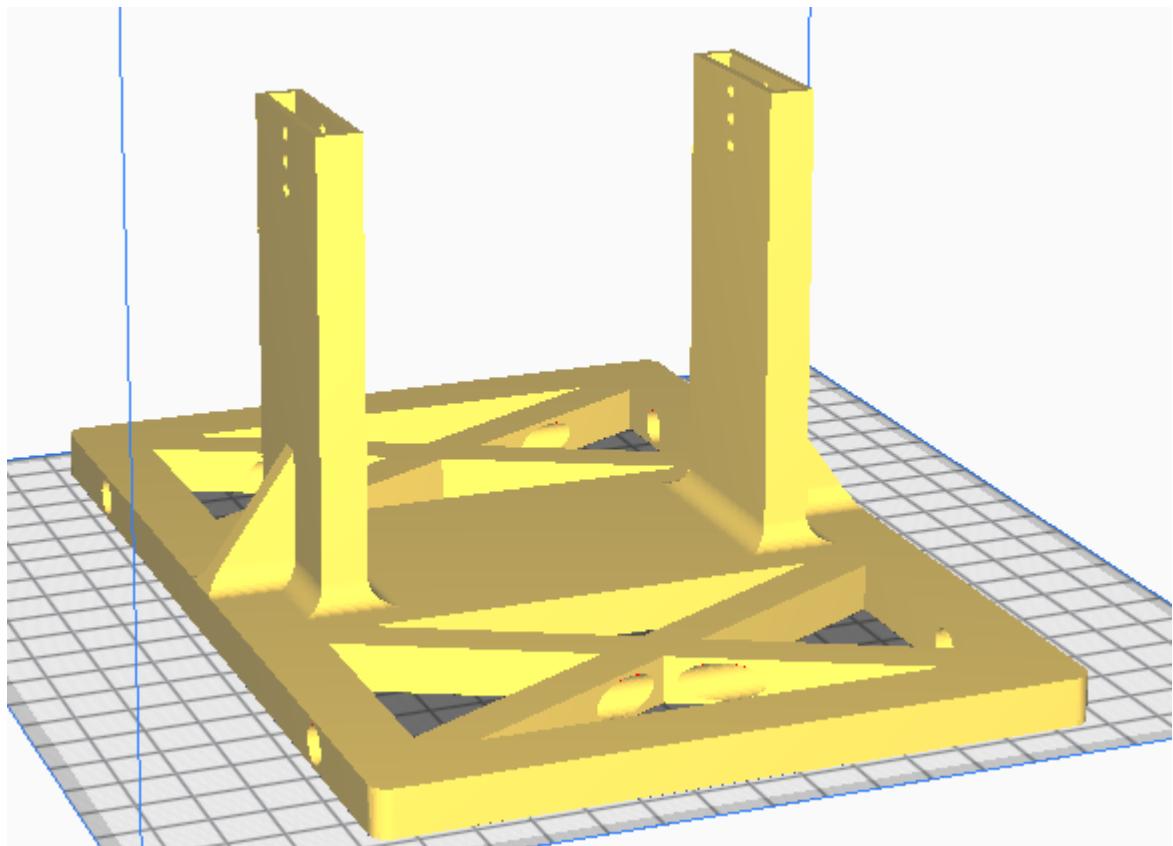


Imagen 54. Diseño del chasis (Autoría propia, 2020)

Diseño de los engranajes de las orugas

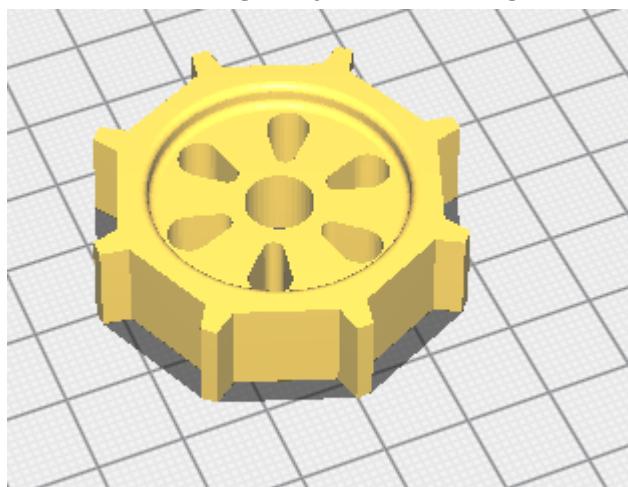


Imagen 54. Diseño de los engranajes de las orugas [Autoría propia, 2020].

Este es un diseño general de los engranajes inferiores de la oruga, se hicieron ajustes en el diseño central para lograr una optimización de 10gr por pieza, sin sacrificar la integridad de la misma.

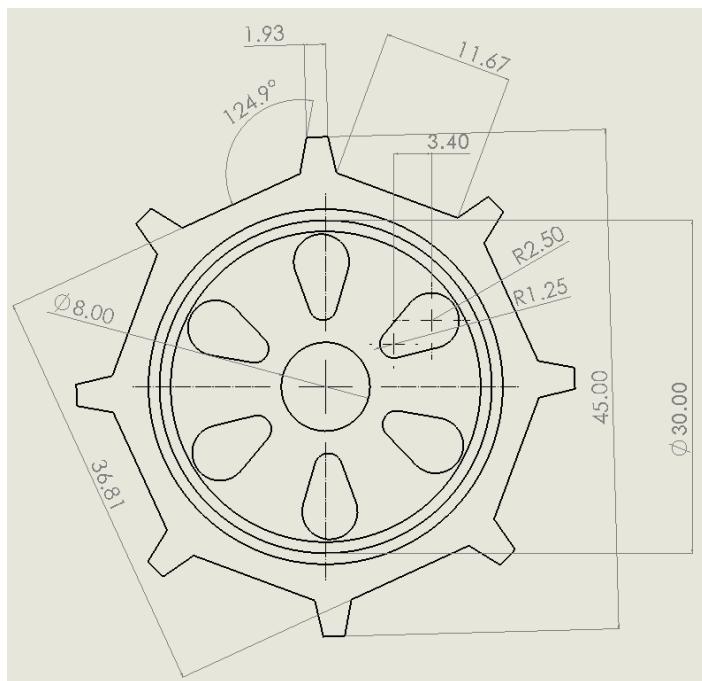


Imagen 55. Plano del engranaje inferior [Autoría propia, 2020].

Diseño de los eslabones de la oruga

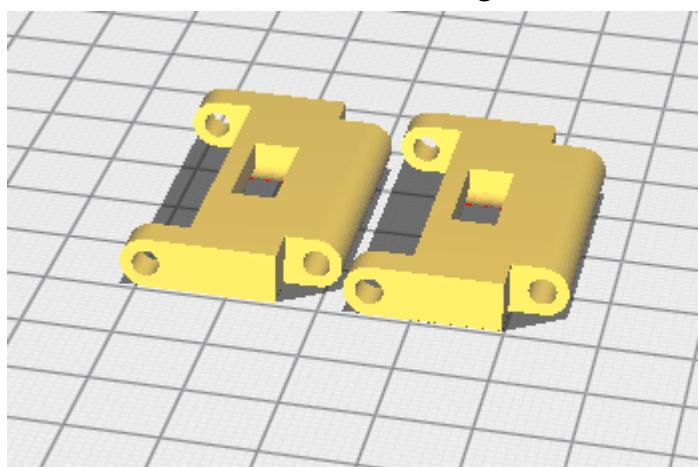


Imagen 56. Diseño de los eslabones de la oruga [Autoría propia, 2020].

El diseño de estos eslabones es un diseño genérico para los cuales se diseñaron los engranajes inferiores que se presentaron anteriormente, le realizó un cálculo de paso para evitar saltos en los dientes y así evitar un desgaste prematuro o falla de transmisión.

Diseño base de motores

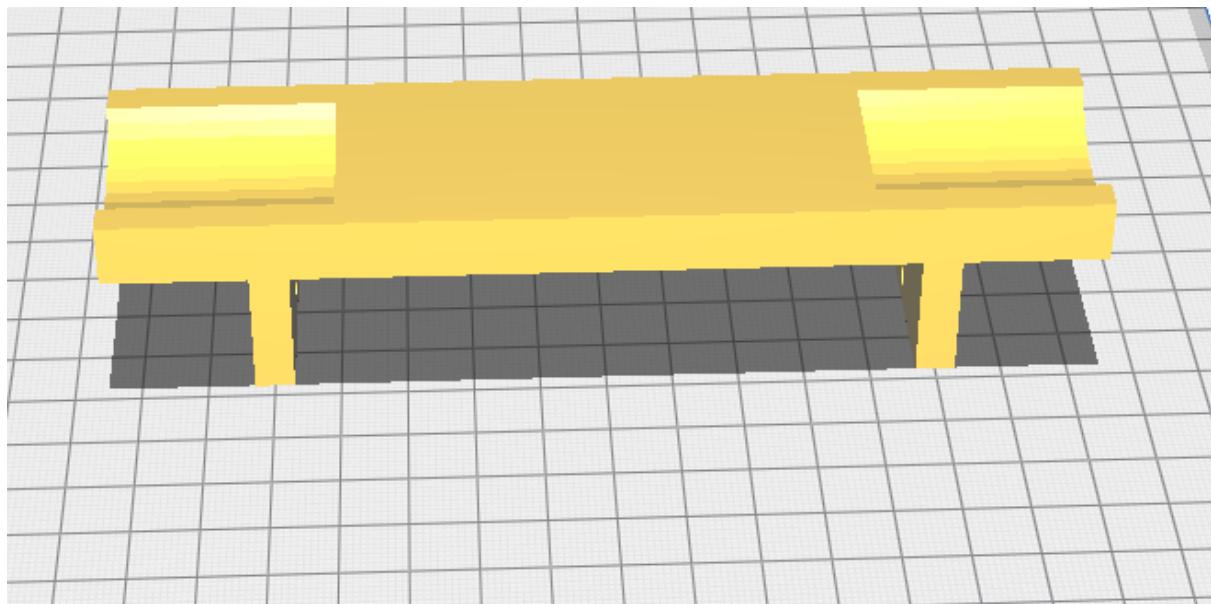


Imagen 57. *Diseño base de motores [Autoría propia, 2021]*.

Diseño brida para motor

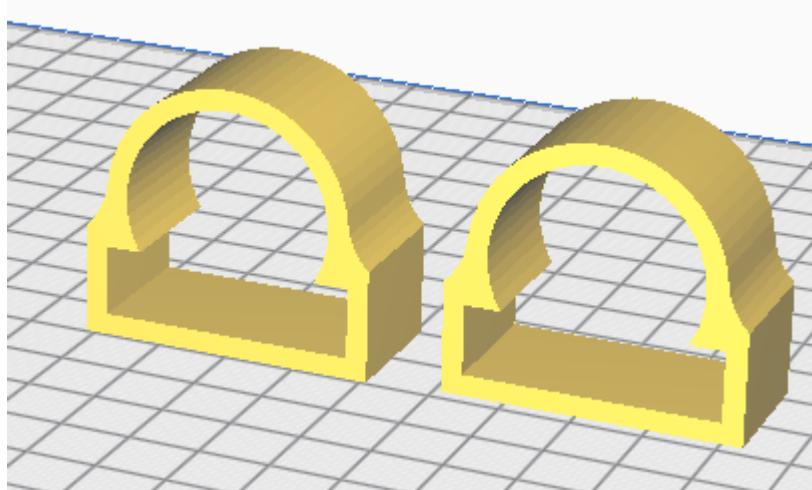


Imagen 58. *Diseño brida para motor [Autoría propia, 2021]*.

Diseño final

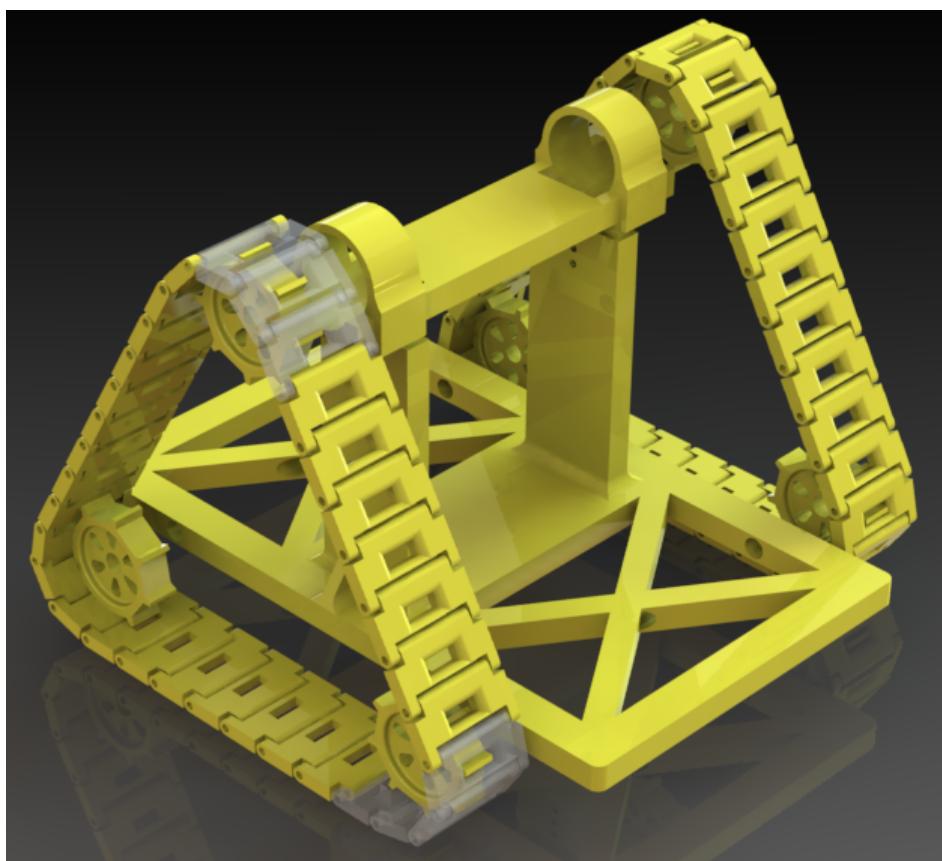


Imagen 59. Diseño final [Autoría propia,2021].

INTERFAZ DE USUARIO

Se muestra la interfaz funcional mostrando los valores obtenidos de los sensores que se encuentran en la habitación así como la cámara para ver lo que está visualizando el robot.



Imagen 60. Interfaz de usuario [Autoría propia,2021].

Código fuente

Arduino

```
#include <SoftwareSerial.h>

String instruction="";
String objective="";
String response;
boolean done=LOW;
boolean last=LOW;
boolean second=LOW;
int c=1,s=2;
int t;
SoftwareSerial espSerial(5, 6);

void setup() {
    //MOTOR VERTICAL
    pinMode(53,OUTPUT);
    pinMode(51,OUTPUT);
    pinMode(49,OUTPUT);
    pinMode(47,OUTPUT);
    //MOTOR HORIZONTAL
    pinMode(52,OUTPUT);
    pinMode(50,OUTPUT);
    pinMode(48,OUTPUT);
    pinMode(46,OUTPUT);
    // Open serial communications and wait for port to open:
    Serial.begin(115200);
```

```
espSerial.begin(115200);

}

void loop() { // run over and over

boolean sended=HIGH;

while (Serial.available()) {

    sended=LOW;

    char r=Serial.read();

    Serial.write(r);

    response.concat(r);

    delay(1);

}

if(!sended){

    if(response.equals("HR")){

        for(int i=0;i<=10;i++){

            nonestep(2);

        }

    }

    if(response.equals("HL")){

        for(int i=0;i<=10;i++){

            onestep(2);

        }

    }

    if(response.equals("VU")){

        for(int i=0;i<=10;i++){


```

```
nonestep(1);

}

}

if(response.equals("VD")){
    for(int i=0;i<=10;i++){
        onestep(1);
    }
}

response="";
}

if(t==5000){

t=0;

String temp=String(random(1500, 4000)/100);

String humedad=String(random(500, 1500)/100);

String gas=String(random(500, 1500)/100);

String laser=String(random(0, 20000)/100);

String ultra=String(random(0, 3000)/100);

String bateria=String(random(0, 10000)/100);

String msg="SENS$"+temp+"$"+humedad+"$"+gas+"$"+laser+"$"+ultra+"$"+bateria;

espSerial.println(msg);

c++;

}else{
    t++;
}

delay(1);
```

```
}

void writee(int a,int b,int c,int d, int M){

    if(M==1){

        digitalWrite(53,a);

        digitalWrite(51,b);

        digitalWrite(49,c);

        digitalWrite(47,d);

    }else if(M==2){

        digitalWrite(52,a);

        digitalWrite(50,b);

        digitalWrite(48,c);

        digitalWrite(46,d);

    }

}

void onestep(int M){

    writee(1,0,0,0,M);

    delay(s);

    writee(1,1,0,0,M);

    delay(s);

    writee(0,1,0,0,M);

    delay(s);

    writee(0,1,1,0,M);

    delay(s);

    writee(0,0,1,0,M);

    delay(s);
```

```
writee(0,0,1,1,M);
```

```
delay(s);
```

```
writee(0,0,0,1,M);
```

```
delay(s);
```

```
writee(1,0,0,1,M);
```

```
delay(s);
```

```
writee(0,0,0,0,M);
```

```
}
```

```
void nonestep(int M){
```

```
writee(0,0,0,1,M);
```

```
delay(s);
```

```
writee(0,0,1,1,M);
```

```
delay(s);
```

```
writee(0,0,1,0,M);
```

```
delay(s);
```

```
writee(0,1,1,0,M);
```

```
delay(s);
```

```
writee(0,1,0,0,M);
```

```
delay(s);
```

```
writee(1,1,0,0,M);
```

```
delay(s);
```

```
writee(1,0,0,0,M);
```

```
delay(s);
```

```
writee(1,0,0,1,M);
```

```
delay(s);
```

```
writee(0,0,0,0,M);
}
```

ESP

```
#include <SoftwareSerial.h>

#include <ESP8266WiFi.h>

#include <WiFiUdp.h>

const char* ssid = "INFINITUM799D_2.4";

const char* password = "CB530045BC";

String response="";

IPAddress local_IP(192, 168, 1, 103);

// Set your Gateway IP address

IPAddress gateway(192, 168, 1, 254);

IPAddress subnet(255, 255, 255, 0);

IPAddress primaryDNS(8, 8, 8, 8); //optional

IPAddress secondaryDNS(8, 8, 4, 4); //optional

WiFiUDP Udp;

unsigned int localUdpPort = 4210; // local port to listen on

char incomingPacket[255]; // buffer for incoming packets

String str;

void setup(){

pinMode(D1,OUTPUT);

Serial.begin(115200);

Serial1.begin(115200);

delay(2000);

// Configures static IP address
```

```
Serial.println("Test de salida");

/*if (!WiFi.config(local_IP, gateway, subnet, primaryDNS, secondaryDNS)) {

    Serial.println("STA Failed to configure");

}*/



digitalWrite(D1,HIGH);

delay(1000);

digitalWrite(D1,LOW);

Serial.printf("Connecting to %s ", ssid);

WiFi.begin(ssid, password);

while (WiFi.status() != WL_CONNECTED)

{

    delay(500);

    Serial.print(".");

}

Serial.println(" connected");

Udp.begin(localUdpPort);

Serial.printf("Now listening at IP %s, UDP port %d\n", WiFi.localIP().toString().c_str(), localUdpPort);

response="IP&"+WiFi.localIP();

int str_len = response.length() + 1;

char char_array[str_len];

response.toCharArray(char_array, str_len);

Udp.beginPacket("192.168.1.99", 4210);

Udp.write(char_array);

Udp.endPacket();

response="";
```

```
digitalWrite(D1,HIGH);

delay(1000);

digitalWrite(D1,LOW);

}

void loop()
{

boolean sended=LOW;

while (Serial.available()) {

sended=HIGH;

char r=Serial.read();

Serial.write(r);

response.concat(r);

delay(10);

digitalWrite(D1,HIGH);

}

Serial.println(sended);

if(sended){

int str_len = response.length() + 1;

char char_array[str_len];

response.toCharArray(char_array, str_len);

Udp.beginPacket("192.168.1.145", 4210);

Udp.write(char_array);
```

```

    Udp.endPacket();

    response="";
    digitalWrite(D1,LOW);

}

int packetSize = Udp.parsePacket();

if (packetSize)

{
    int len = Udp.read(incomingPacket, 255);

    if (len > 0)

    {

        incomingPacket[len] = 0;

    }

    Serial1.printf(incomingPacket);

}

delay(1000);

}

```

Processing

```

import processing.video.*;
import hypermedia.net.*;

int medx=350,medy=350;

String msg;

UDP udp;

Boolean con=false;

int life=0;

String temp="0",humedad="0",gas="0",laser="0",ultra="0",bateria="0";

```

```
PImage img,img2,img3,img4,vcam,fondo;

Capture video;

int clk = 1;

String message; // the message to send

String ip      = "192.168.1.170"; // the remote IP address

int port       = 4210;           // the destination port

void setup() {

    size(displayWidth, displayHeight);

    println(displayWidth);

    println(displayHeight);

    String[] cameras = Capture.list();

    video=new Capture(this,1280,720,cameras[0]);

    video.start();

    udp = new UDP( this, 4210 );

    udp.listen( true );

    //message = "0";

    //udp.send( message, ip, port );

    img = loadImage("imagenes/termo.png");

    img2 = loadImage("imagenes/battery.png");

    img3 = loadImage("imagenes/rule.png");

    img4 = loadImage("imagenes/arrow.png");

    vcam=loadImage("imagenes/marco.png");

    fondo=loadImage("imagenes/fondo.jpg");

}

void draw() {
```

```
background(0, 0, 0);

image(fondo,0,0);

if (video.available() == true) {

    video.read();

}

if(!icon){

    textSize(50);

    fill(255, 0, 0);

    text("No conectado", 500, 350);

}else{

    int cx=displayWidth/2,cy=260;

    stroke(255);

    strokeWeight(5);

    imageMode(CENTER);

    image(video,cx, cy,896,504);

    image(vcam,cx,cy,1060,1010);

    imageMode(CORNER);

    textSize(15);

    fill(255,255,255);

    text("Temperatura: "+temp+"°",medx+50,medy+680);

    text("Humedad: "+humedad+"%",medx+300,medy+680);

    text("Nivel CO: "+gas+"%", medx+550,medy+680);

//text("Distancia Camara: "+laser+"cm",medx+50,medy+280);

    text("Dist front: "+ultra+"cm",medx+800,medy+680);

    text("Bateria: "+bateria+"%",medx+1050,medy+680);
```

```
image(img, medx+50, medy+360, 100,300);

fill(255,0,0);

float yi=-2.2*(Integer.valueOf(temp));

noStroke();

rect(medx+89, medy+599, 19, yi);

stroke(255);

fill(25,0,0,0);

arc(medx+355, medy+630, 200, 200, radians(-180), 0, PIE);

fill(51, 204, 204);

arc(medx+355, medy+630, 200, 200, radians(-180),
radians(1.8*Integer.valueOf(humedad)-180), PIE);

fill(25,0,0,0);

arc(medx+355, medy+630, 150, 150, radians(-180), 0, PIE);

fill(51,204,204);

textSize(30);

text(humedad+"%",medx+335,medy+620);

fill(140,242,30);

noStroke();

yi=-1.4*(Integer.valueOf(bateria));

rect(medx+1050,medy+ 599, 70, yi);

image(img2, medx+985, medy+400, 200,250);

//3er

stroke(255);

fill(25,0,0,0);

arc(medx+600,medy+ 630, 200, 200, radians(-180), 0, PIE);

fill(255, 153, 0);
```

```

arc(medx+600, medy+630, 200, 200, radians(-180),
radians(1.8*Integer.valueOf(gas)-180), PIE);

fill(25,0,0,0);

arc(medx+600, medy+630, 150, 150, radians(-180), 0, PIE);

fill(255, 153, 0);

textSize(30);

text(gas+"%",medx+575,medy+620);

//



image(img3,medx+750,medy+350,200,300);

yi=-3*(Integer.valueOf(ultra))+595;

image(img4,medx+870,medy+yi,50,50);

fill(255);

textSize(20);

text("CONTROLES CAMARA",40,40);

}

if(life>=500){

con=false;

}else{

life++;

}

//text(life,50,50);

delay(1);

}

void exec(String inst) {

String message = inst;

// formats the message for Pd

```

```
// send the message  
udp.send( message, ip, port );  
  
}  
  
void receive( byte[] data){  
    data = subset(data, 0, data.length-2);  
    msg = new String( data );  
    println(msg);  
    String[] list = split(msg, '$');  
    if(list[0].equals("SENS")){  
        temp=list[1];  
        humedad=list[2];  
        gas=list[3];  
  
        laser=list[4];  
        ultra=list[5];  
        bateria=list[6];  
    }else if(list[0].equals("IP")){  
        ip=list[1];  
    }  
    con=true;  
    life=0;  
}
```

```
void keyPressed() {  
    if (key == CODED) {  
  
        if (keyCode == UP) {  
            exect("VU");  
        } else if (keyCode == DOWN) {  
            exect("VD");  
        }  
        else if (keyCode == LEFT) {  
            exect("HL");  
        }  
        else if (keyCode == RIGHT) {  
            exect("HR");  
        }  
    }  
}  
}
```

Pruebas de armado y operación

En esta sección presentamos evidencias de algunos componentes conectados y el comportamiento del prototipo.

Armado y prueba de ESP

Se realizó la conexión del componente ESP y vemos que se muestra la red para su uso, luego vemos su funcionamiento en conjunto con el Arduino.



Imagen 61. Prueba del ESP [Autoría propia, 2021].

The screenshot shows the Arduino IDE interface with the following details:

- Title Bar:** UDP_code | Processing 3.5.4
- Menu Bar:** Archivo, Editar, Sketch, Depuración, Herramientas, Ayuda
- Code Area:** The code is for an ESP32 using the Arduino framework. It includes functions for setting up UDP, processing events, drawing, and handling key presses. It also includes a UDP message sending function and a note about implementing a handler for UDP datagram reception.
- Serial Monitor:** A window titled "COM4" is open, showing the serial communication log. It displays multiple received UDP packets from IP 192.168.1.145, port 4210, containing the message "This is simple Message from Computer".
- Bottom Status Bar:** Shows the message "packets, please refer to the UDP Class documentation" from 192.168.1.170 on port 4210.
- Bottom Navigation:** Includes tabs for Consola and Errores, and a search bar.

Imagen 62. Prueba del código del ESP [Autoría propia, 2021].

Conexión de los componentes suministrados por la batería

Conexión completa de los componentes electrónicos a utilizar siendo alimentados por la batería

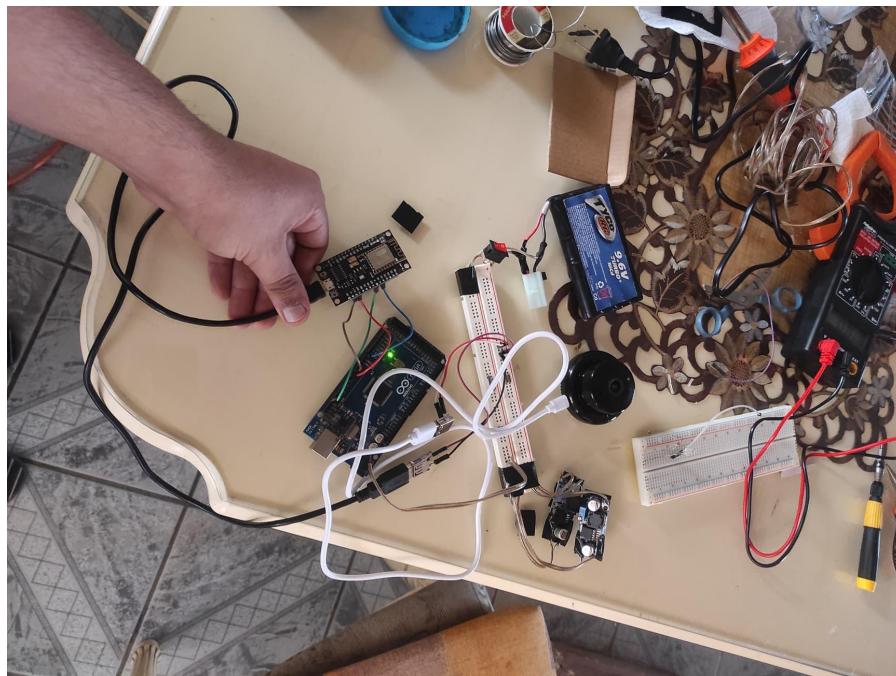


Imagen 63. Conexión de componentes [Autoría propia, 2021].

Interfaz funcional con los componentes

Se muestra la interfaz funcional mostrando los valores obtenidos de los sensores que se encuentran en la habitación.

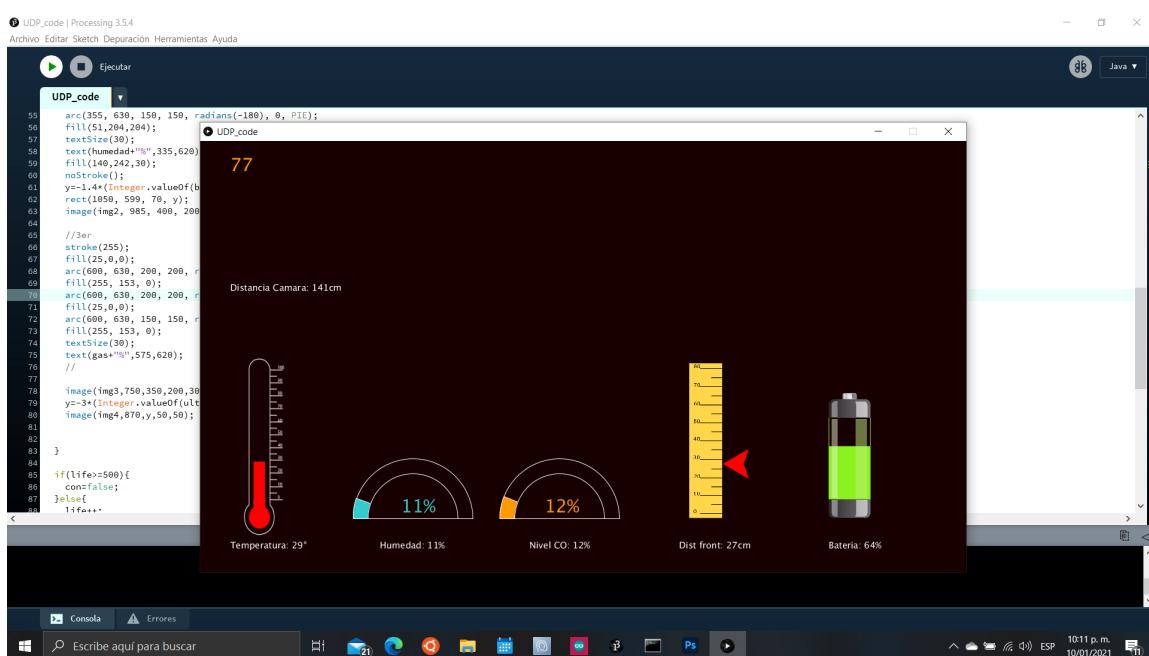


Imagen 64. Interfaz funcional [Autoría propia, 2021].

Armado motores a paso junto con la cámara

Transmisión de la cámara al arduino y del arduino al computador, donde ya se encuentra montada en los motores a paso.



Imagen 65. Armado de motores a paso [Autoría propia, 2021].

Impresión y armado del prototipo

Se realizó la impresión del diseño parte por parte y al final el armado con los motores.

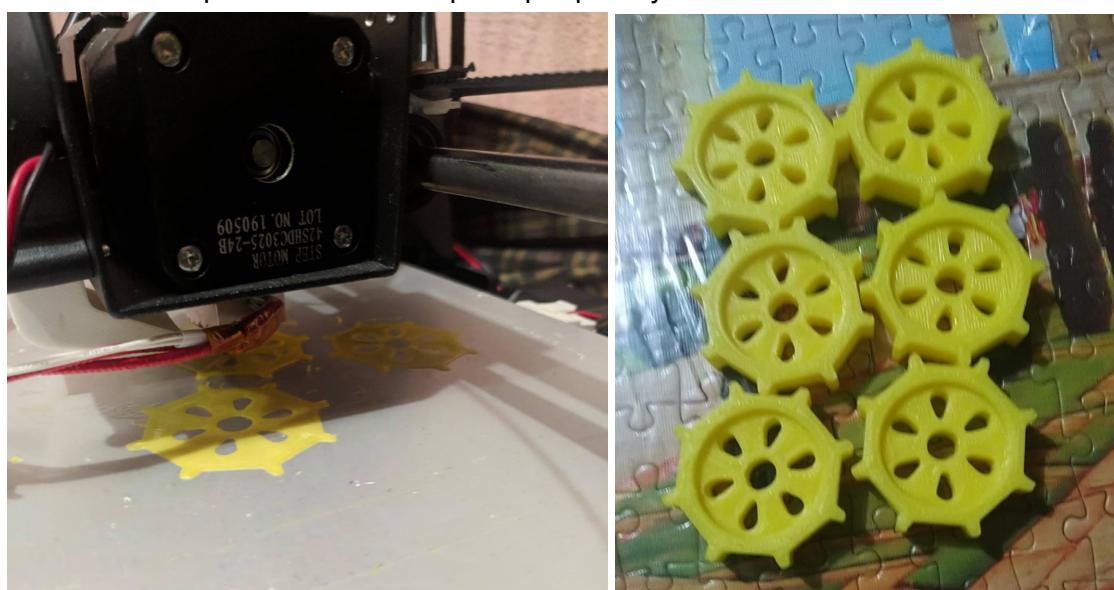


Imagen 66. Impresión y armado del prototipo [Autoría propia, 2021].

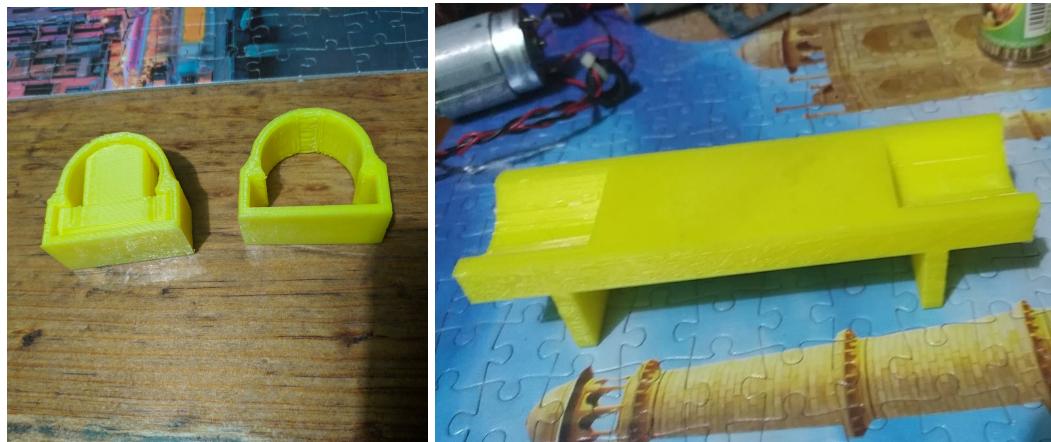


Imagen 67. Impresión y armado del prototipo [Autoría propia,2021].

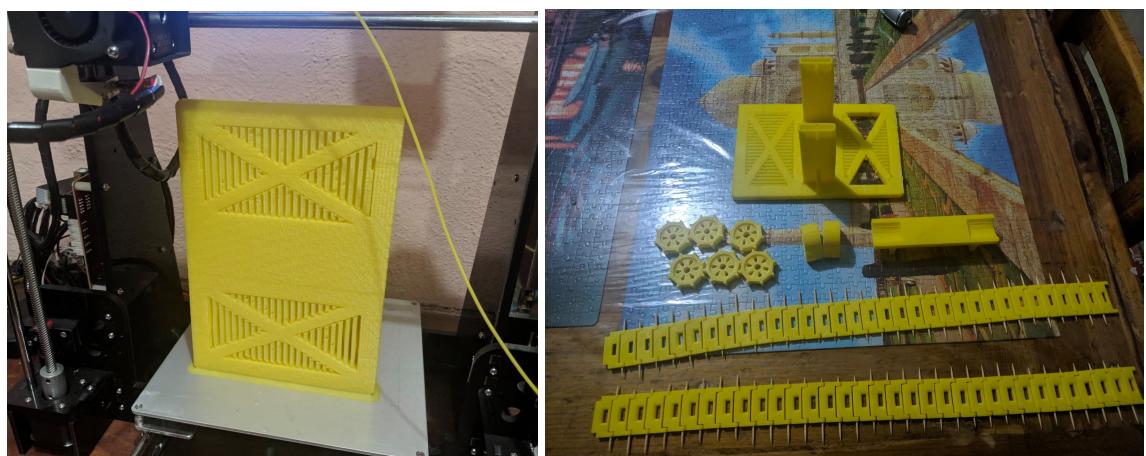


Imagen 68. Impresión y armado del prototipo [Autoría propia,2021].

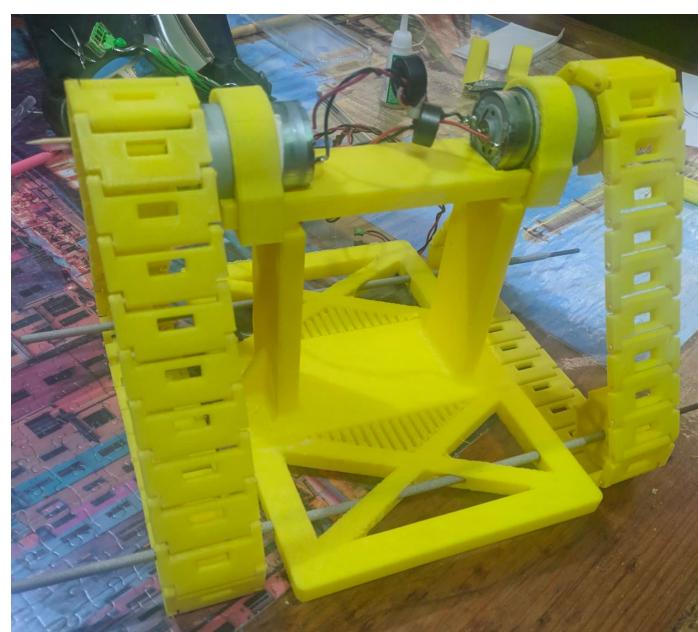


Imagen 69. Impresión y armado del prototipo [Autoría propia,2021].

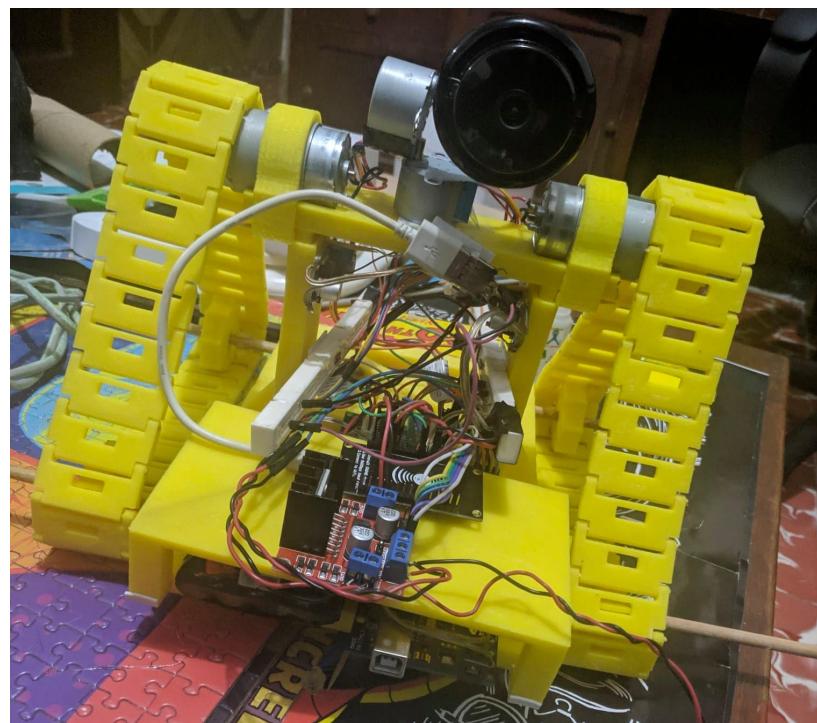


Imagen 70. Prototipo con componentes [Autoría propia,2021].

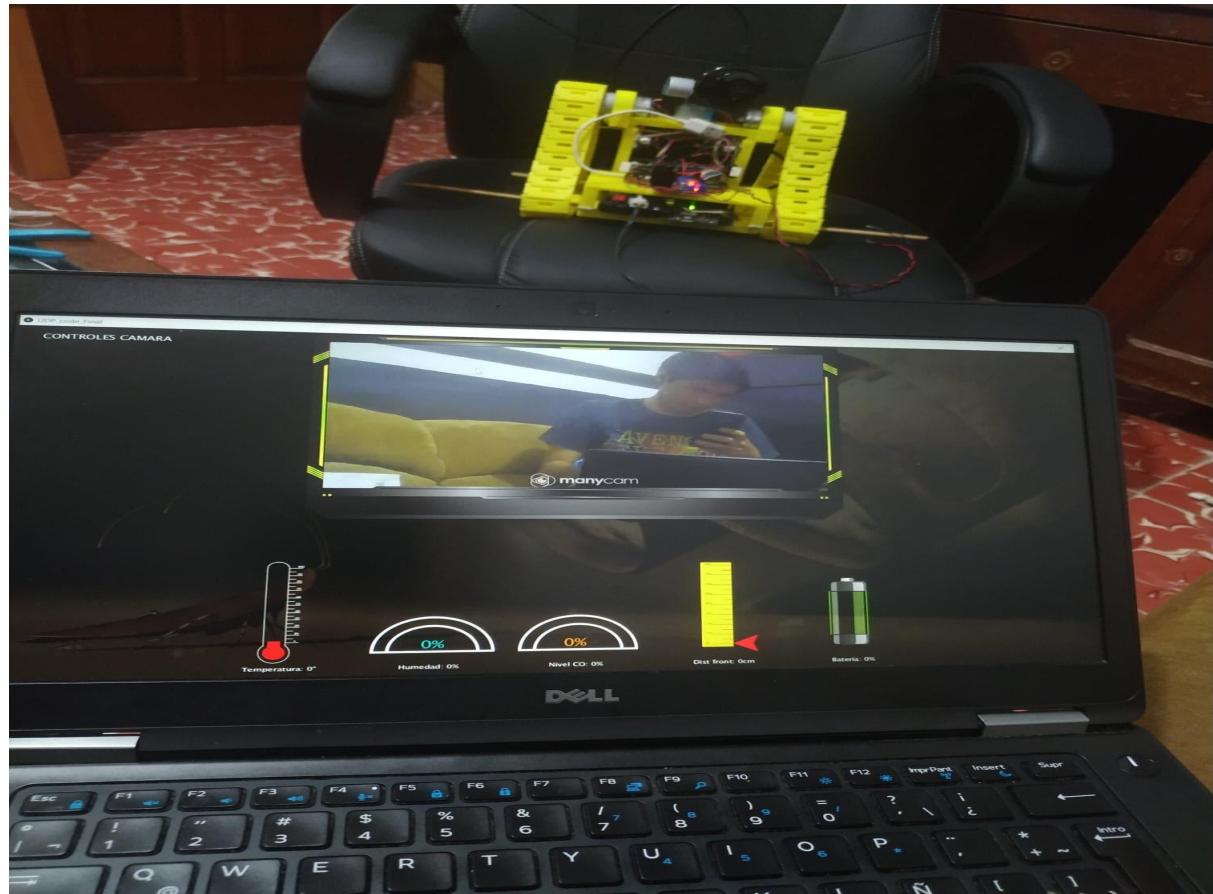


Imagen 71. Prototipo en funcionamiento [Autoría propia,2021].

PROSPECCIÓN COMERCIAL

Actualmente existen diversos robots exploradores, estos robots son utilizados mayormente para el desarrollo de proyectos o para enseñar a los alumnos a utilizar diversos componentes electrónicos, estos robots son en un grado menor como el que se muestra en las imágenes a continuación, estos tienen precios muy variados dependiendo el grado de complejidad que manejan.

Se puede observar que el salto entre un precio a otro es demasiado alto, pero esto igual va de la mano con las características que ofrecen, la siguiente imagen muestra un kit básico para un robot.

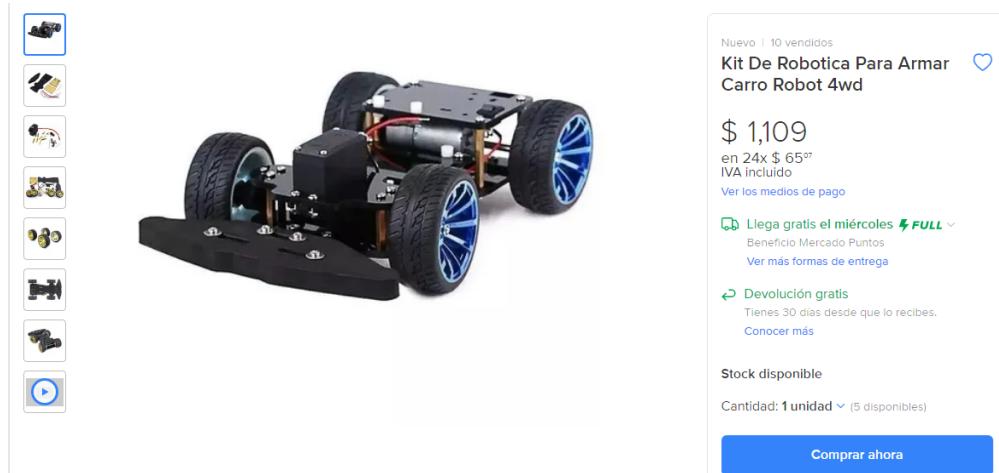


Imagen 72. Robot sencillo. [Mercado Libre,2021].

El siguiente robot es más parecido al que desarrollamos ya que incluye una cámara así como algunos sensores y de igual manera se controla inalámbricamente.

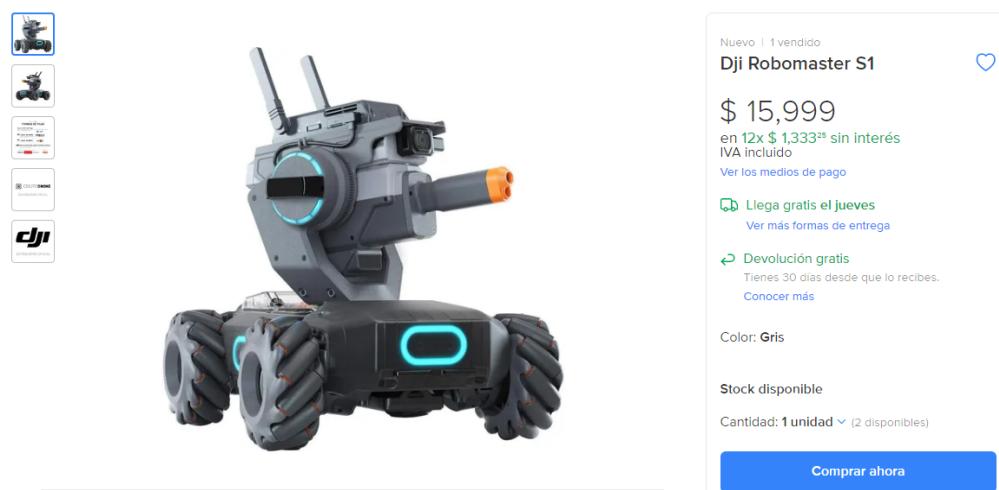


Imagen 73. Robot avanzado. [Mercado Libre,2021].

Por lo que podemos decir que comparando con lo anterior nuestro prototipo es más eficaz a lo que encontramos comercialmente o de acceso más fácil gracias a las ventajas que implementamos y el costo mucho menor a lo que tienen otros modelos.

En contraparte igual tenemos los robots exploradores avanzados como los siguientes:

- Valkyrie Robonaut R5 es un robot humanoide creado por la NASA para desarrollar tecnología que permita en un futuro cercano preparar asentamientos con los que colonizar el Planeta Marte.



Imagen 74. Valkyrie Robonaut R5. [NASA, 2021].

- Robot espacial Spirit. El 3 de Enero del 2004 el robot Spirit enviado por la Nasa llegó a Marte, realizando en el planeta rojo trabajos de investigación hasta que en marzo del 2010 dejó de enviar información.



Imagen 75. Robot espacial Spirit. [NASA, 2021].

- Rover Curiosity en el Planeta Marte. Se trata de un robot espacial que la NASA envió al planeta Marte el 25 de noviembre del 2011 y que aterrizó con éxito en 2012, concretamente en el cráter Gale aunque ha posteriormente ha realizado diversas investigaciones en el Monte Sharp. También se le conoce como Mars Science Laboratory o MSL, y tiene una forma similar a la del robot Spirit y Opportunity.

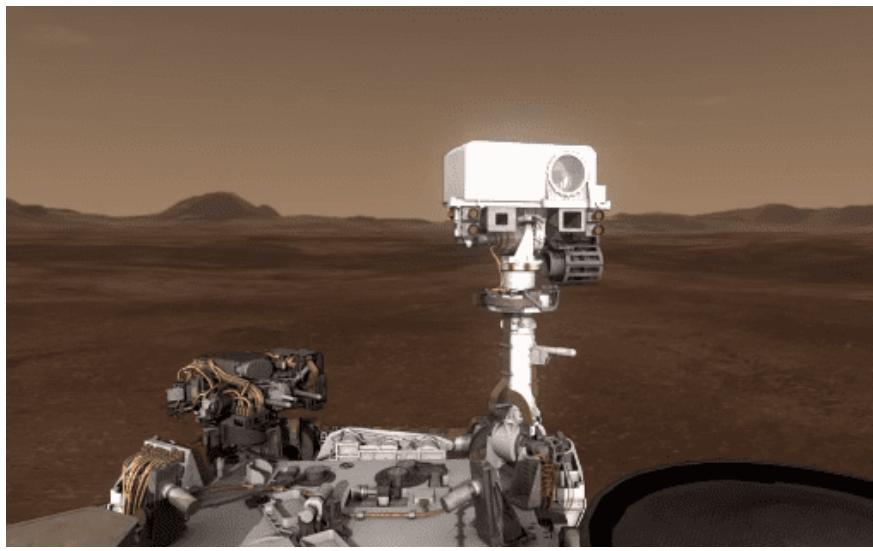


Imagen 76. Rover Curiosity. [NASA, 2021].

AJUSTES FINALES

En la etapa actual del prototipo los ajustes finales que se pueden realizar para un mejor funcionamiento son:

1. Tensar las bandas en diferentes niveles para encontrar el rendimiento óptimo.
2. Ajustar correctamente los motores para que no tengan movimiento alguno.
3. Ajustar los engranajes para que con el movimiento no salgan de su posición original.

TRABAJOS FUTUROS

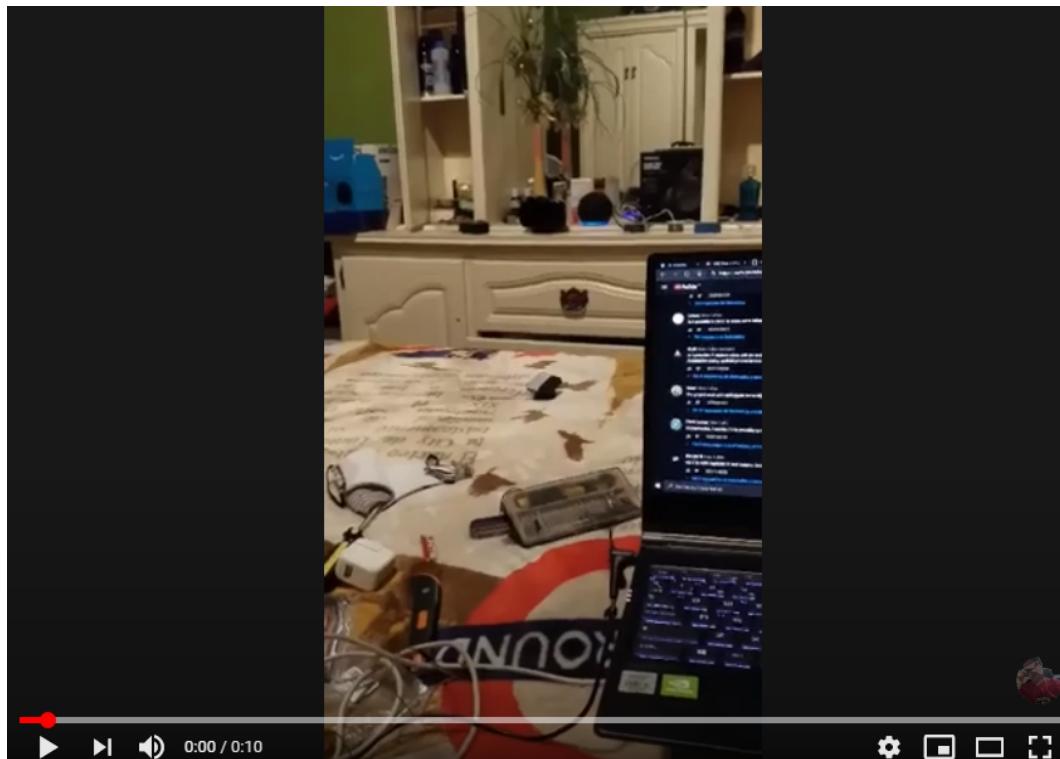
El prototipo actualmente es funcional con restricciones, para corregir ésto es necesario realizar algunas modificaciones como lo son:

1. Utilizar una fuente de alimentación única para los componentes, ya que actualmente tiene diferentes fuentes y por ello una puede tener suficiente batería pero la otra no y comienza a tener un comportamiento no esperado.
2. Mejor gestión de cables.
3. Rediseño del cuerpo para tener una mejor organización de los componentes.
4. Agregar sensores más eficientes y confiables.
5. Utilizar una cámara directamente conectada al controlador en lugar de una wifi.
6. Utilizar un mejor motor para el movimiento del prototipo ya que no cuentan con la suficiente potencia para moverlo constantemente debido a su peso.
7. Optimización del código utilizado para el funcionamiento.

VIDEOS DEL PROTOTIPO

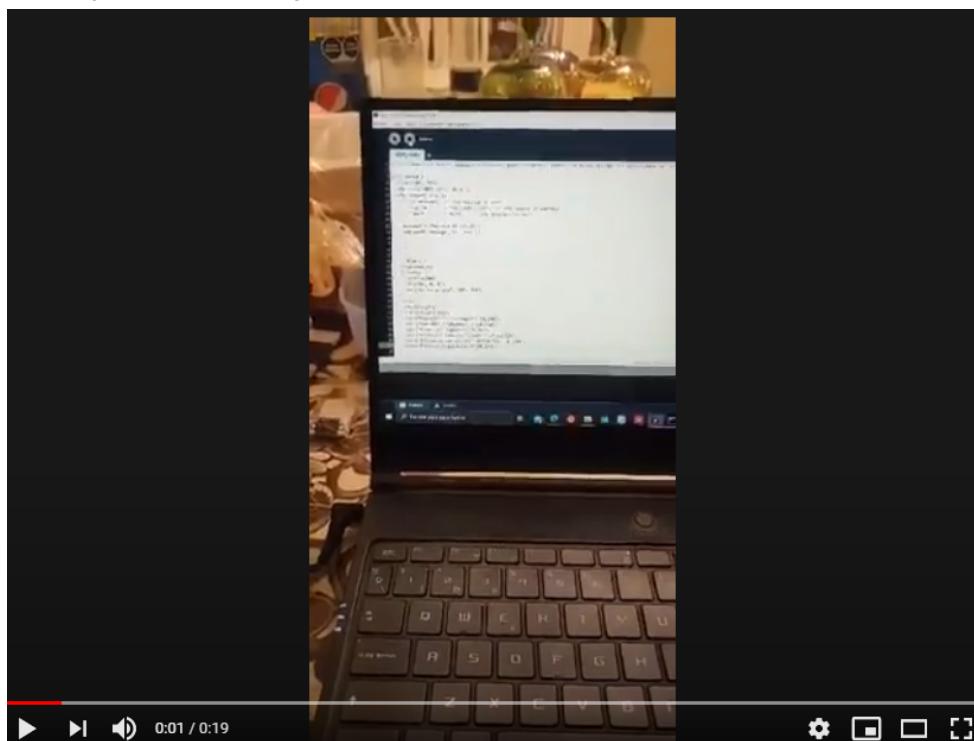
- Comunicación inalámbrica

<https://www.youtube.com/watch?v=0zUbFSScatA>



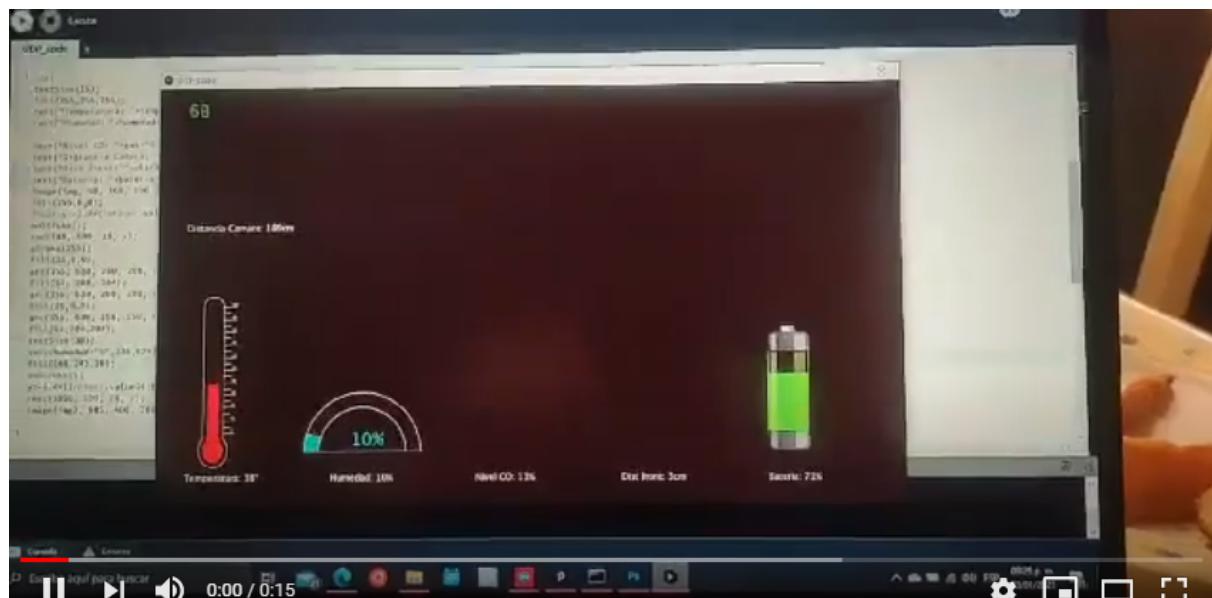
- Conexión de con los sensores

https://youtu.be/vLI-D_jQWMA



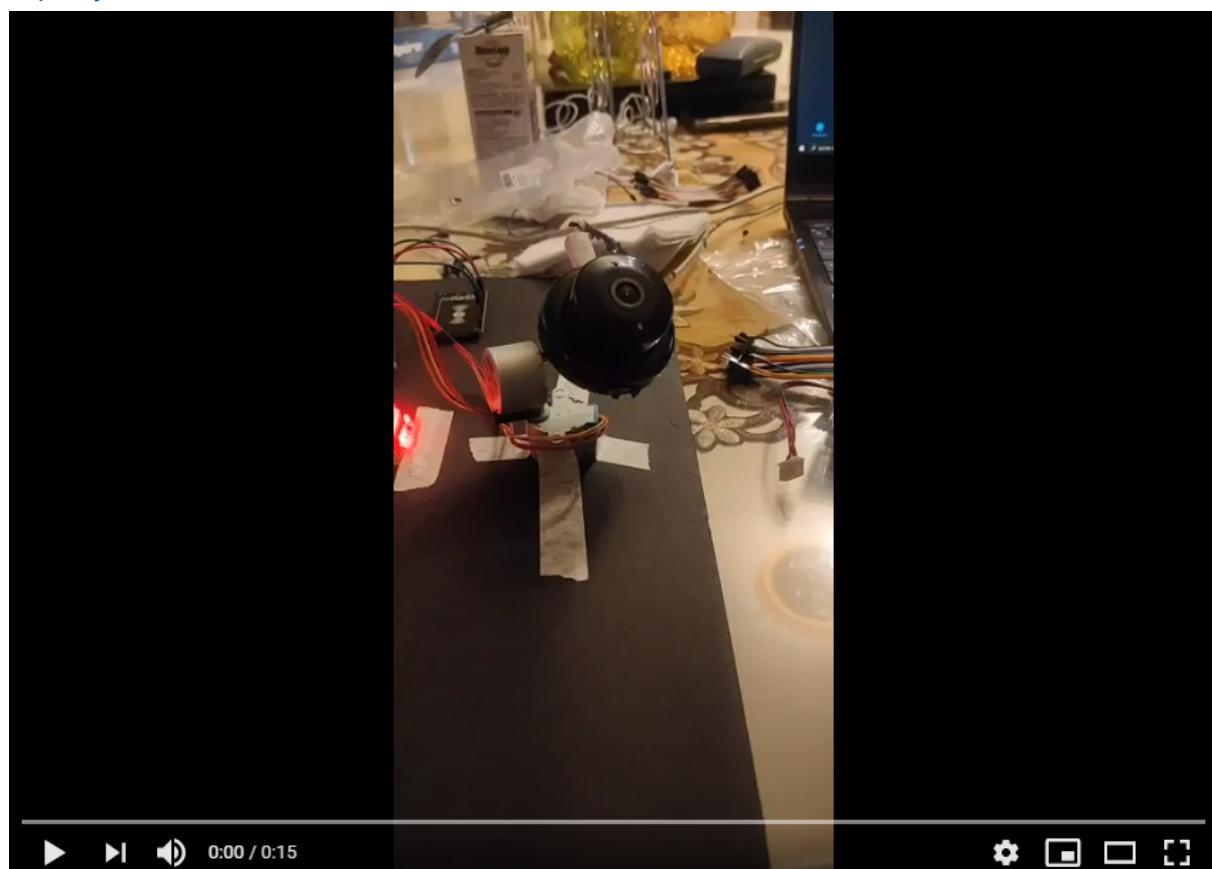
- Actualización interfaz principal

<https://youtu.be/ja6WpU1tbGw>



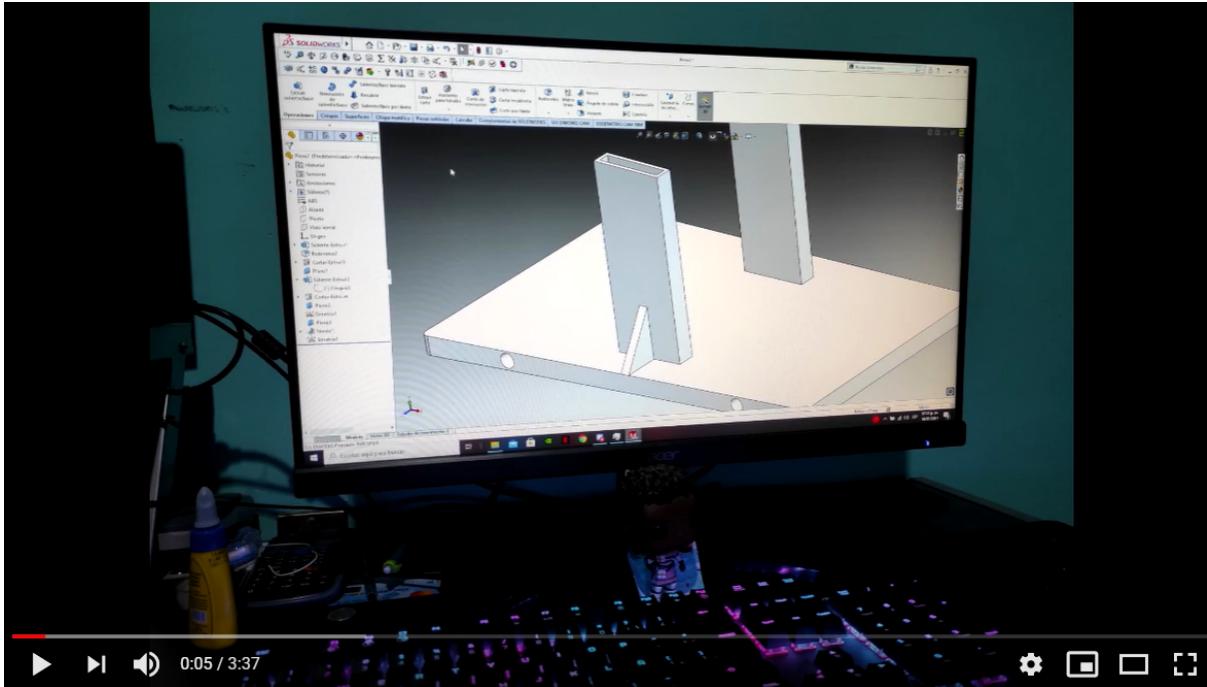
- Movimiento de cámara

<https://youtu.be/XuteL1OBEB0>



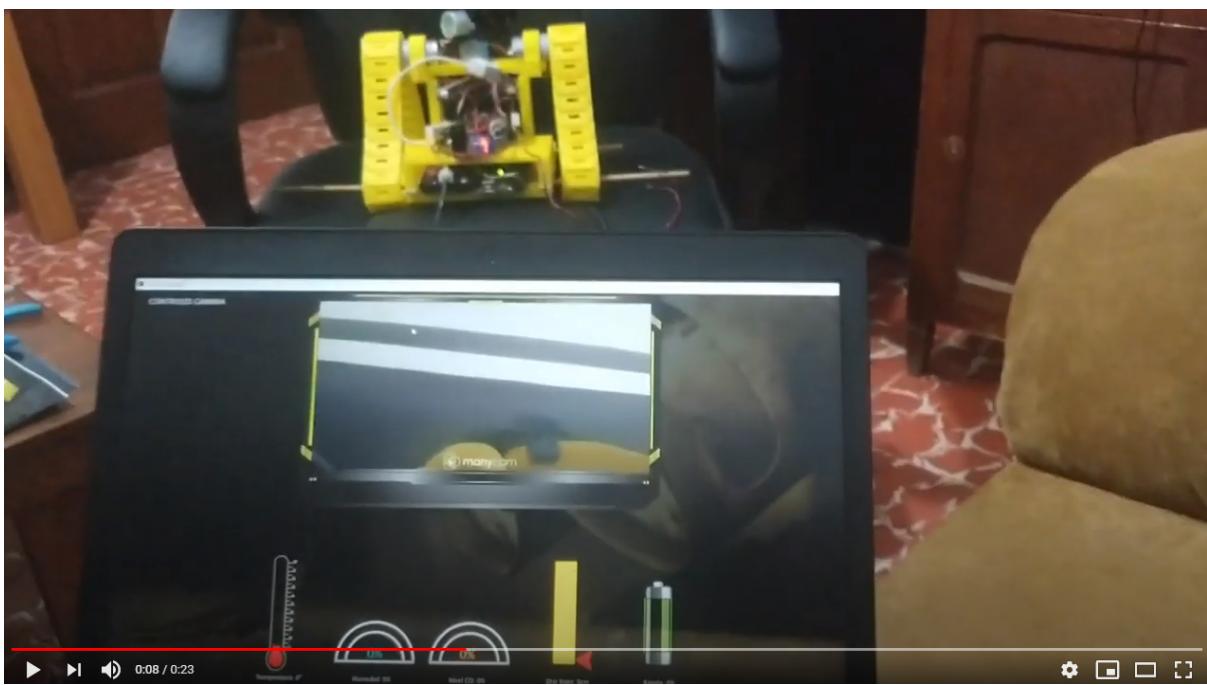
- Resumen del Avance, diseño hasta montado

<https://www.youtube.com/watch?v=fil6G4VHTWw>



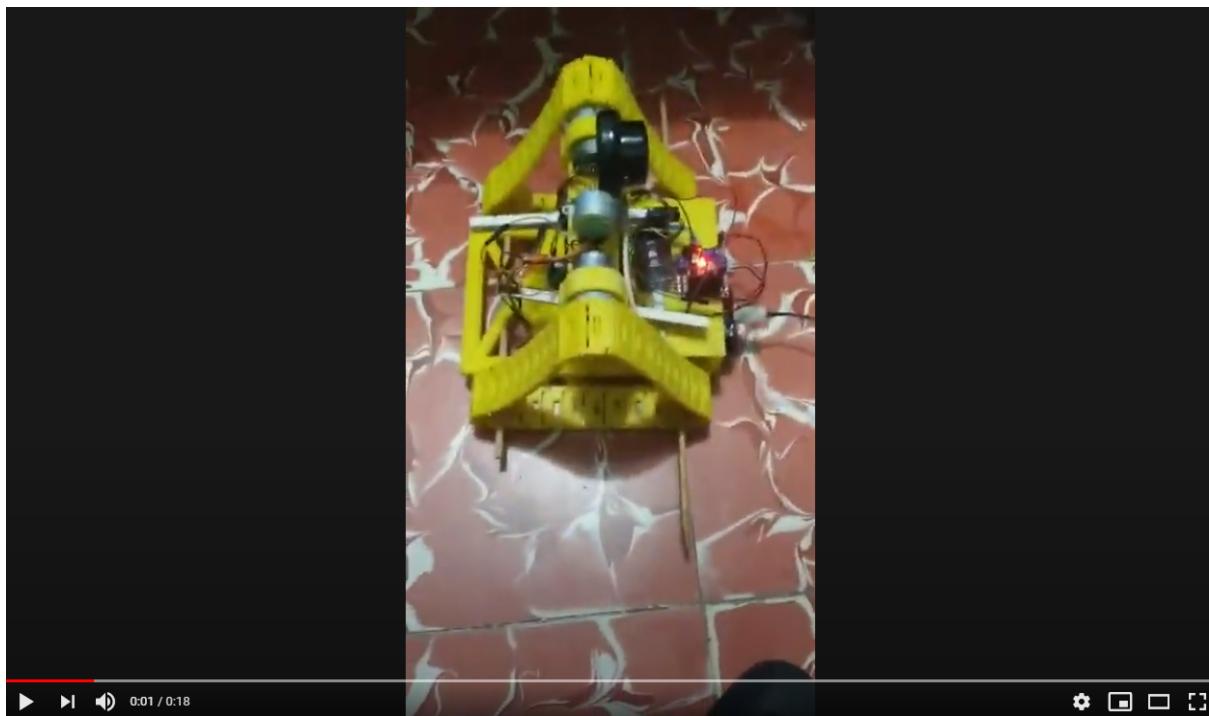
- Control de cámara inalámbrico

<https://www.youtube.com/watch?v=6d6pYFaB0Vw>



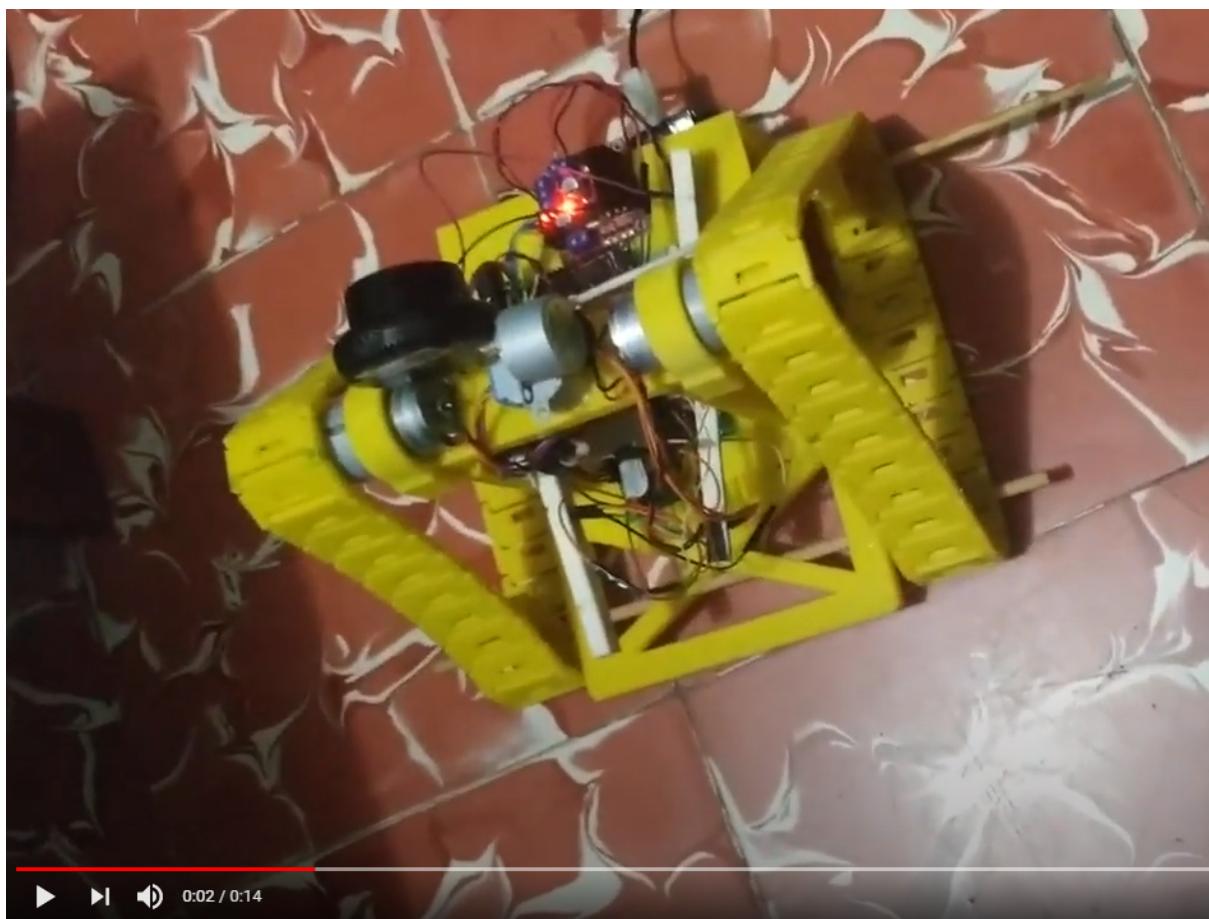
- Movimiento del prototipo 1

<https://www.youtube.com/watch?v=f01tzy5BLqY>



- Movimiento del prototipo 2

<https://www.youtube.com/watch?v=f0uWB7BzmXU>



BIBLIOGRAFÍA

Arduino. (2020, 10 23). *¿Que es Arduino?* Arduino. <https://arduino.cl/que-es-arduino/>

Arduino Core. (2020). *Librería ESP8266WiFi.*

<https://esp8266-arduino-spanish.readthedocs.io/es/latest/esp8266wifi/readme.html>

CDMX Electrónica. (2020, 10 23). *LM2596 Regulador Step Down 25W 3A.* LM2596

Regulador Step Down 25W 3A.

<https://cdmxelectronica.com producto/modulo-regulador-ajustable-lm2596-dc-dc-step-down-3a-1-25-30v/>

Demeyer, Z. (2018). *Información general acerca de la Mega 2560 de Arduino: la hermana mayor de la placa Uno.* Arrow Electronics.

<https://www.arrow.com/es-mx/research-and-events/articles/arduino-mega-2560-overview>

electrocrea. (2020, 10 23). *motor-dc-3-12v.* motor-dc-3-12v.

<https://electrocrea.com/products/motor-dc-3-12v-rectangular>

electronicaplugandplay. (2020, Octubre 23). *Módulo Sensor de Distancia Láser VL53L0X.*

Módulo Sensor de Distancia Láser VL53L0X.

<https://www.electronicaplugandplay.com/sensores-y-transductores/product/542-modulo-sensor-de-distancia-laser-vl53l0x>

geekfactory. (2020, Octubre 23). *MQ-9 Sensor de Gas.* MQ-9 Sensor de Gas.

<https://www.geekfactory.mx/tienda/sensores/mq-9-sensor-de-gas/#:~:text=El%20m%C3%B3dulo%20MQ-9%20Sensor,la%20concentraci%C3%B3n%20de%20gas%20detectada.>

Gómez, J. (2016, 05 24). *Dibujo libre.* dibujo libre.

<https://sites.google.com/site/artesvisualesitm123/processing>

Inicio Processing Qué es Processing Qué es Processing. (2016, 05 19). Eldesvandejose.

<https://eldesvandejose.com/2016/05/19/que-es-processing/>

Llamas, L. (2017). CONECTAR ARDUINO POR WIFI CON EL MÓDULO ESP8266 ESP01.

<https://www.luisllamas.es/arduino-wifi-esp8266-esp01/>

LM2596. (n.d.).

<https://cdmxelectronica.com/producto/modulo-regulador-ajustable-lm2596-dc-dc-step-down-3a-1-25-30v/>

LTH. (2020, 10 23). *Bateria LTH Ctz10s-bs*. Bateria LTH Ctz10s-bs.

<https://www.lth.com.mx/es-mx/productos/motobaterias/lth-motobaterias-agm/ctz10s-bs>

Mercado Libre. (2020, 10 23). *Cámara Con Visión Nocturna 2mp 1080p Wifi Inalámbric*.

Cámara Con Visión Nocturna 2mp 1080p Wifi Inalámbric.

https://articulo.mercadolibre.com.mx/MLM-775443727-camara-con-vision-nocturna-2mp-1080p-wifi-inalambric-_JM

naylampmechatronics. (2020, 10 23). *Driver Puente H L298N 2A*. Driver Puente H L298N

2A. <https://naylampmechatronics.com/drivers/11-driver-puente-h-l298n.html>

naylampmechatronics. (2020, Octubre 23). *Módulo Driver ULN2003*. Módulo Driver

ULN2003. <https://naylampmechatronics.com/drivers/366-modulo-driver-uln2003.html>

naylampmechatronics. (2020, Octubre 23). *Módulo ESP-01 ESP8266 WiFi-Serial*. Módulo

ESP-01 ESP8266 WiFi-Serial.

<https://naylampmechatronics.com/espressif-esp/48-modulo-esp-01-esp8266-wifi-serial.html>

naylampmechatronics. (2020, Octubre 23). *Sensor Ultrasonido HC-SR04*.

<https://naylampmechatronics.com/sensores-proximidad/10-sensor-ultrasonido-hc-sr04.html>

<https://naylampmechatronics.com/sensores-proximidad/10-sensor-ultrasonido-hc-sr04.html>

Patagoniatec. (2020). *MotorPaso A Paso 28BYJ-48 Con ULN2003*. Motor Paso A Paso 28BYJ-48 Con ULN2003. Retrieved 2020, from

<https://saber.patagoniatec.com/motor-paso-a-paso-28byj-48-uln2003-arduino/>

project hub. (2020). *How to Calibrate & Use MQ9 Gas Sensor w/ Arduino © GPL3+*.

<https://create.arduino.cc/projecthub/electropeak/how-to-calibrate-use-mq9-gas-sensor-w-arduino-e93cb1>

ROBOT EXPLORADOR. (2019, 01 01). *ROBOT EXPLORADOR*. ROBOT EXPLORADOR.

<https://www.robotexplorador.com>

Rodríguez, D. (2020, 10 22). *Método experimental: características, etapas, ejemplo*. lifeder.

<https://www.lifeder.com/metodo-cientifico-experimental/>

sandorobotics. (2020, 10 23). *Motor a Pasos 28BYJ-48 (5V)*. Motor a Pasos 28BYJ-48 (5V).

<https://sandorobotics.com/producto/hr0260/>

siaguanta. (2020). *Conexión como cliente a servidor TCP desde processing*.

<https://siaguanta.com/wp-content/uploads/2019/12/arg-1024x433.png>

Valle Hernández, L. (2018, 05 20). *Programar fácil*. Cómo utilizar el sensor DHT11 para medir la temperatura y humedad con Arduino.

<https://programarfácil.com/blog/arduino-blog/sensor-dht11-temperatura-humedad-arduino/>

Wikipedia. (n.d.). Arduino. <https://es.wikipedia.org/wiki/Arduino>

Wikipedia. (n.d.). *Arduino IDE*. https://es.wikipedia.org/wiki/Arduino_IDE

Wikipedia. (2020). *Wiring*. Wiring. Retrieved 2020, from <https://es.wikipedia.org/wiki/Wiring>

Wikipedia. (2020, Octubre 23). *SolidWorks*. SolidWorks.

<https://es.wikipedia.org/wiki/SolidWorks>