
PROYECTO: BARRA DE GIMNASIO INTELIGENTE

Inteligencia Ambiental



Juan Bautista Muñoz Ruiz, Ismael Expósito Jiménez

jbmr0001@red.ujaen.es, iej00003@red.ujaen.es

Spotter-Inteligente	2
1- Antecedentes	2
2- Estado del Arte	3
3- Objetivo:	4
3- Descripción	5
3.1- Barra	6
3.2- Estructura móvil y Servomotor	6
4- Esquemas	9
5- Presupuesto y Materiales	10
6- Planificación	11
7- Metodología	11
7.1- Componentes finales utilizados	11
7.2- Diseño	15
7.3- Programación del motor en el LEGO Brick	17
7.4- Programación en Arduino	21
7.5- Construcción	23
8- Resultados	26
9- Conclusiones	27
10- Referencias	27

Spotter-Inteligente

1- Antecedentes

Cuando hablamos de Inteligencia Ambiental nos referimos a la integración de sistemas inteligentes junto con el usuario con el objetivo de incrementar el bienestar. Ya sea facilitando una tarea, incrementando la seguridad, automatizando un proceso o dotando de inteligencia a objetos cotidianos.

Estos sistemas toman decisiones tras recibir la entrada de datos de unos perceptores exteriores del ambiente del usuario mediante sensores, cámaras, otros dispositivos o interfaces.

En este proyecto nos centraremos en los sistemas de seguridad inteligentes.

A día de hoy son muchos los sistemas inteligentes que tenemos a nuestro alcance. Algunos de los principales ejemplos son:

- Sistemas de Detección de Incendios.
- Sistemas de Detección de Fugas de Gas.
- Sistemas de Detección de Intrusión en el Domicilio.
- Sistemas de Vídeo-monitorización.
- Sistemas de Detección de Movimiento.

Para este proyecto nos diferenciaremos de la domótica del hogar para centrarnos en la seguridad en el gimnasio.

2- Estado del Arte

El deporte es uno de los pilares fundamentales de una vida saludable. Actualmente, debido a la vida sedentaria derivada de la gran mayoría de oficios con escasa demanda física, el sector fitness es un campo en auge. Por lo que es de principal interés en desarrollar dispositivos inteligentes para este ámbito.

En este campo encontramos gran variedad de dispositivos en la monitorización de la actividad deportiva como: contadores de pasos, monitorizadores de distancia recorrida vía GPS, monitorizadores de pulso. Algunos ejemplos del mercado:

- [Monitorización Fitness de Ejercicios](#): Se trata de un dispositivo que cuenta con una pantalla integrada y dos "brazos" que terminan en un cable elástico. A estos brazos se le conectan los diferentes elementos compatibles para cada ejercicio. Entre sus funciones se encuentran: Creación de rutinas, seguimiento del ejercicio y personalización de las rutinas.
- [Monitorizador de Ejercicios de Barra y Cuerda para el Hogar](#): Es prácticamente similar al anterior. Principales diferencias:
 - Peso digital ajustable.
 - Soporta mayor número de ejercicios: barra, canoa, etc...
 - Altura de los brazos ajustable.

- [Contador de Pasos](#): Este dispositivo transforma el movimiento del cuerpo en pasos para calcular la distancia recorrida y las calorías consumidas.
- [Pesa Rusa Inteligente](#): Dispone de conectividad para monitorizar el número de repeticiones realizadas.
- [Rastreador GPS para el Cálculo de Rendimiento en Fútbol](#): Se trata de un chaleco GPS junto con un acelerómetro que se conecta vía Bluetooth al móvil. Sirve para medir: distancia recorrida, velocidad máxima, esprines, mapas de calor de movimiento, momentos del partido y comparar tus datos con los de los futbolistas profesionales.

Todos los dispositivos están centrados en el control de los ejercicios del usuario, creación de rutinas y seguimiento del progreso. Sin embargo, ninguno se centra en la seguridad del ejercicio. Por lo que, para este proyecto, nos centraremos en el desarrollo de un sistema de seguridad para una ejercicio de barra de gimnasio.

3- Objetivo:



El objetivo de este proyecto es incrementar la seguridad durante la realización de un ejercicio de barra. Son bastante numerosos, y de alta gravedad, los accidentes producidos en esta rutina. Se necesita de una persona para ayudar en la realización del ejercicio y quitar la barra en el caso de ser necesario. Pero muchas veces esto no es así, ya sea porque no haya personas que puedan ayudar en el ejercicio, el levantador crea que no va a

necesitar ayuda, u otros motivos. Resolver esta problemática es una de las asignaturas pendientes en el mundo del Fitness actual.

Para ello construiremos, desarrollaremos e implementaremos un dispositivo inteligente. Un sistema similar al que podemos ver en el siguiente [vídeo](#).

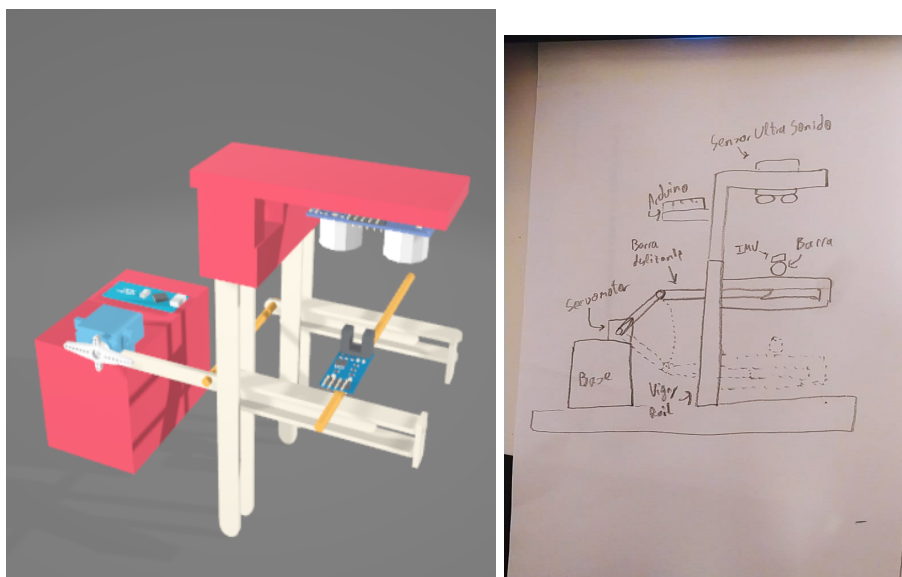
Este prototipo contará con un Spotter-Inteligente junto a una Barra-Inteligente. La función del Spotter es seguir a la barra y estar “atento” para que no se caiga. La Barra-Inteligente detecta cuando se está cayendo y de ser así el Spotter se detendrá para sostener el peso y evitar que caiga.

La principal diferencia con respecto a otros dispositivos en el mercado es la condición de pionero. Es decir, no hay casi ningún producto en el mercado. Y más aún, en el campo de la seguridad de la realización del ejercicio.

Como consecuencia. la utilidad de un producto así es máxima. Ya que, además de no existir nada parecido en el mercado, ayuda a las personas y reduce el número de accidentes producidos en este tipo de ejercicios.

3- Descripción

Se tratará de un prototipo a escala del sistema real para evitar excesivos costes. Por lo que para el material de esta estructura se usarán piezas de Lego.



3.1- Barra

En cuanto a la barra de gimnasio se utilizará un cilindro de cartón. Para detectar en todo momento su posición y movimiento tendrá un sensor IMU acoplado.

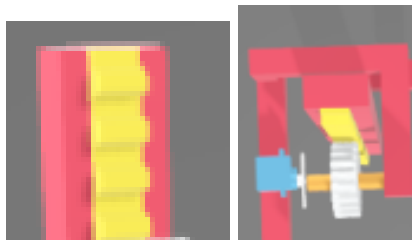
Sabremos si la barra está cayendo cuando tenga una aceleración instantánea cercana a la aceleración de la gravedad. Este sensor estará conectado a nuestra placa arduino. El programa tomará las decisiones de qué debe hacer el motor.



3.2- Estructura móvil y Servomotor

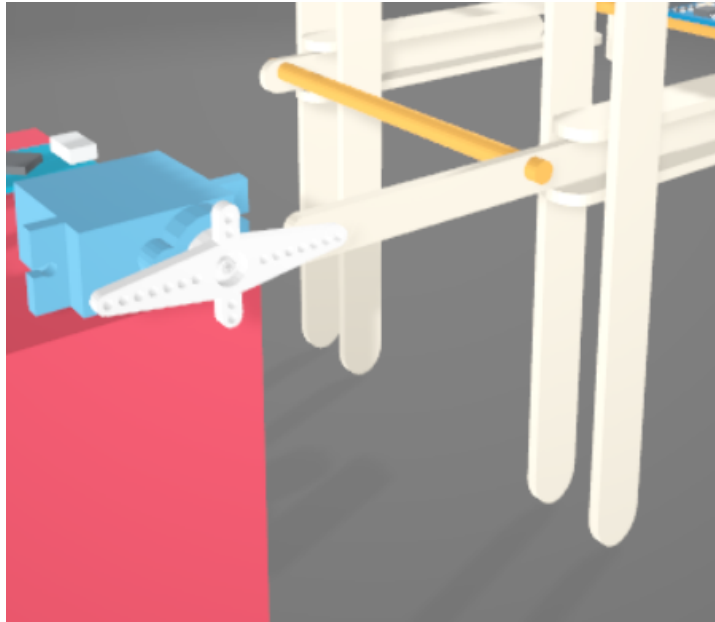
La estructura de seguridad consta con las siguientes partes:

- **Vía vertical y servomotor:**
 - El Servomotor se desplazará hacia arriba o hacia abajo por una vía o canal vertical mediante una rueda acoplada. La rueda y la vía contarán con recubrimiento dentado o de goma.



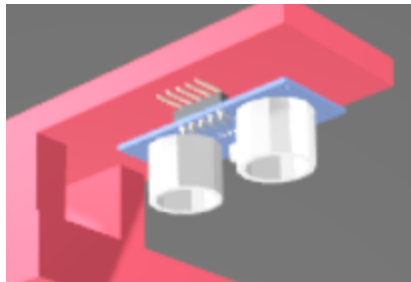
- En el caso de no poder implementar este mecanismo, se tendrá una viga o rail vertical por el que se desliza el brazo horizontal. Este brazo es empujado

por otro brazo que rota y está unido al aspa del servomotor



- **Sensor de UltraSonido:**

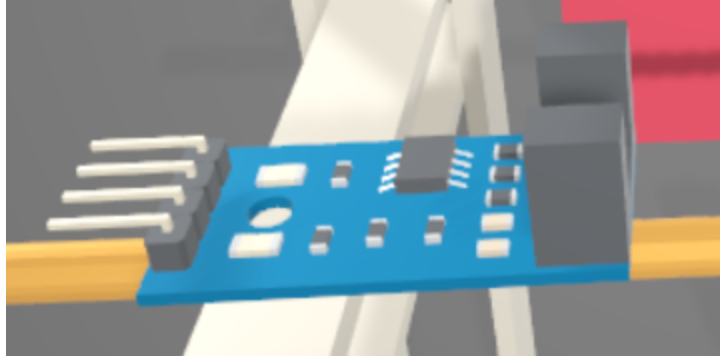
- Este estará sobre una base fija en la parte superior alineado con la dirección de la barra, en una posición que esta no debería alcanzar. Desde ahí recoge la posición de la barra y le envía los datos al arduino para que este tome las decisiones.



-

- **IMU:**

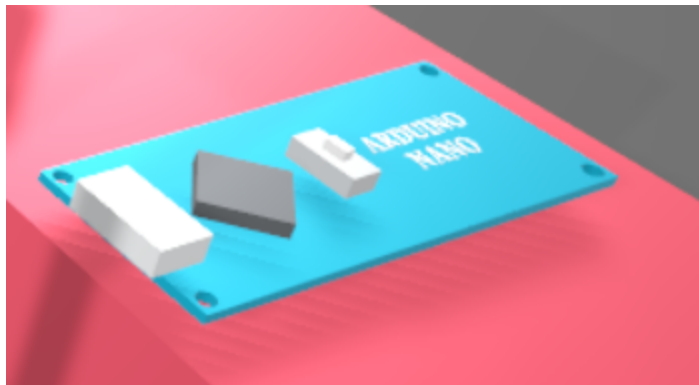
- Este sensor estará en la barra que se esté levantando, toma la medida de la aceleración que hay en la barra y de esta forma el Arduino puede saber de si la barra está bajando de forma controlada o si está cayendo



○

- **Arduino:**

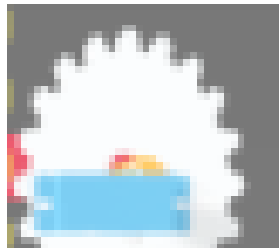
- Este tomará los datos del sensor ultrasónico y del IMU, si detecta que la barra se está moviendo de forma controlada. Es decir, no está cayendo, manda al servomotor que gire en el ángulo adecuado que se corresponde a la posición de la barra. En el caso de que detecte que la barra está cayendo pasa al modo recoger barra y manda la señal al servomotor para que el brazo ascienda y eleve la barra.



○

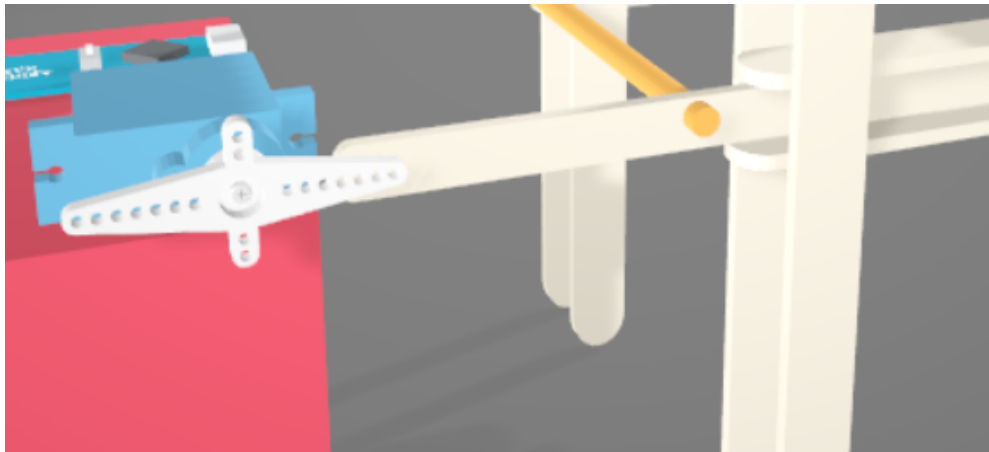
- **Servomotor con rueda:**

- El servomotor estará unido a una estructura que abrace a la vía para evitar que caiga la rueda.

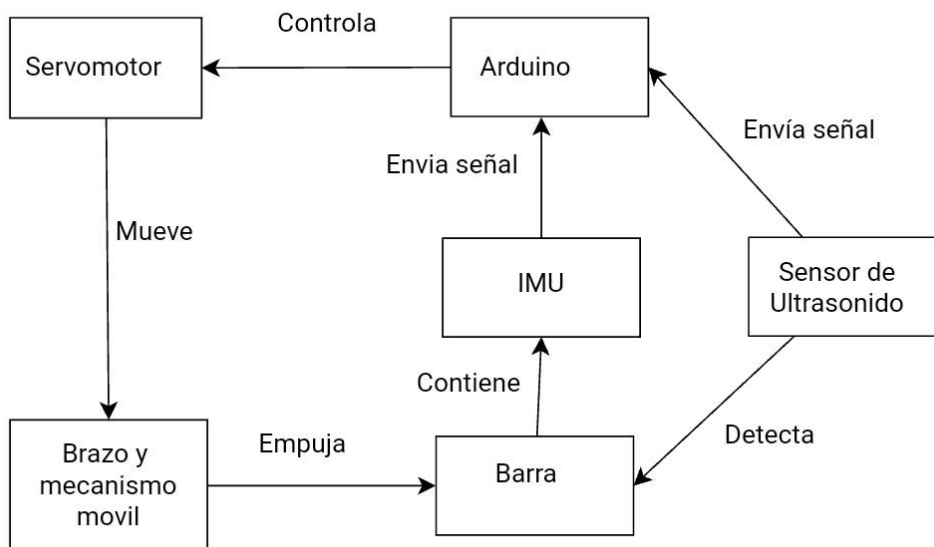


- **Servomotor con brazo:**

- En el caso de tener un servomotor con un brazo que extienda el recorrido que cubre las aspas del servomotor, este estará en una posición elevada que se ubique en la misma altura del punto medio del recorrido de la barra.



4- Esquemas




5- Presupuesto y Materiales

Elemento	Enlace	Precio
Sensor IMU	https://amzn.eu/d/1sMFnie	11,49 €
Arduino	https://www.amazon.es/AR	43,06 €
Motor Servo	https://amzn.eu/d/fbhbLUe	8,32 €
Piezas de Lego	https://www.amazon.es/Stri	12,48 €
Cilindro de cartón	https://www.amazon.es/Alb	3,75 €
Sensor ultrasónico	https://www.amazon.es/AZ	5,99 €
Protoboard	https://createc3d.com/es/m	1,50 €
Rueda para motor servo con recubrimiento de goma o recubrimiento dentado	https://electronperdido.c	2,25 €

Pack de cableado	https://www.amazon	4,99 €
Cable de alimentación para el arduino	https://www.concash	1,79 €

6- Planificación

 Juan Bautista Muñoz Ruiz

 Ismael Expósito Jiménez

Planificación del proyecto

TAREAS	MARZO			ABRIL			MAYO		
Creación del Anteproyecto									
Adquisición de materiales									
Creación de la estructura									
Integración y programación del servomotor									
Creación e Integración de la barra con el sensor IMU									
Programación y comunicación del Imu con el arduino									
Desarrollo del prototipo									
Finalización del prototipo									
Presentación del proyecto									

Fecha estimada para la presentación del proyecto: 8-21 de Mayo de 2023

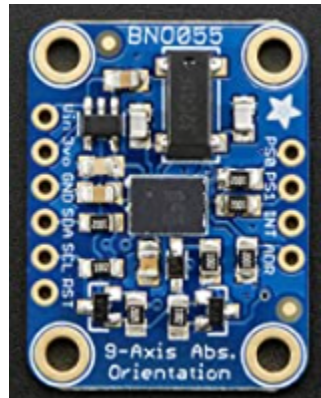
La planificación del proyecto ha sido establecida en el siguiente [diagrama de Gantt](#).

7- Metodología

7.1- Componentes finales utilizados

Tras varias pruebas y decisiones de diseño estos fueron los principales componentes de nuestro sistema:

- **Sensor IMU:** Unidad de medición inercial para detectar cuando la barra cae.



- **Sensor Ultrasónico:** Utilizado para medir la distancia a la que se encuentra la barra. La principal causa de esta elección es el hecho de que el sensor de infrarrojos no fuera capaz de detectar la distancia a la que se encontraba la barra. Simplemente detectaba si había un obstáculo o no, por lo que tuvimos que cambiarlo.



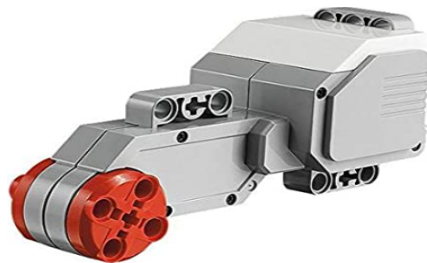
- **Arduino MKR1000:** Se trata del mismo Arduino utilizado en las sesiones de prácticas. Se encargará de recibir la información de los dos sensores anteriores así como enviar instrucciones al broker.



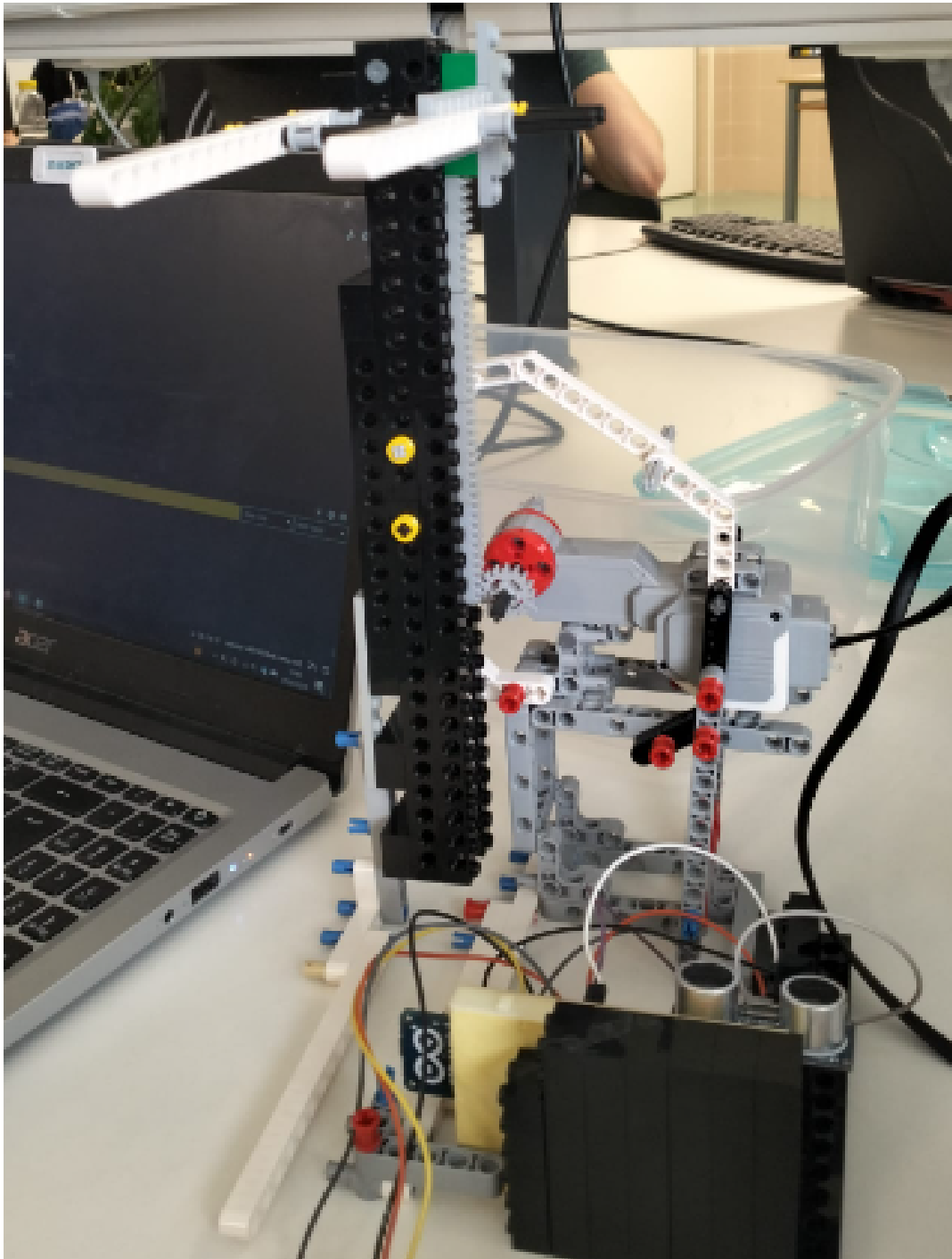
- **Brick Lego EV3:** De cara a simplificar la integración del motor con la estructura, así como la programación del motor, se optó por usar la centralita de Lego EV3 para su control.



- **Motor para Lego EV3:** Se trata del motor compatible con la centralita Lego EV3 anteriormente descrita. Este motor es controlable mediante MicroPython.



- **Estructura de LEGO:** Estructura construida con dos tipos de piezas de Lego.



Consiste en un mecanismo piñón-cremallera que sube o baja según la rotación de la rueda traccionadora. Cuenta con la protoboard con el Arduino y el sensor Ultrasónico mirando hacia arriba de cara a medir la distancia a la que se encuentra la barra.

- **“Barra de Gimnasio”:** A modo de simulación de la barra hemos utilizado una caja de papel de aluminio.



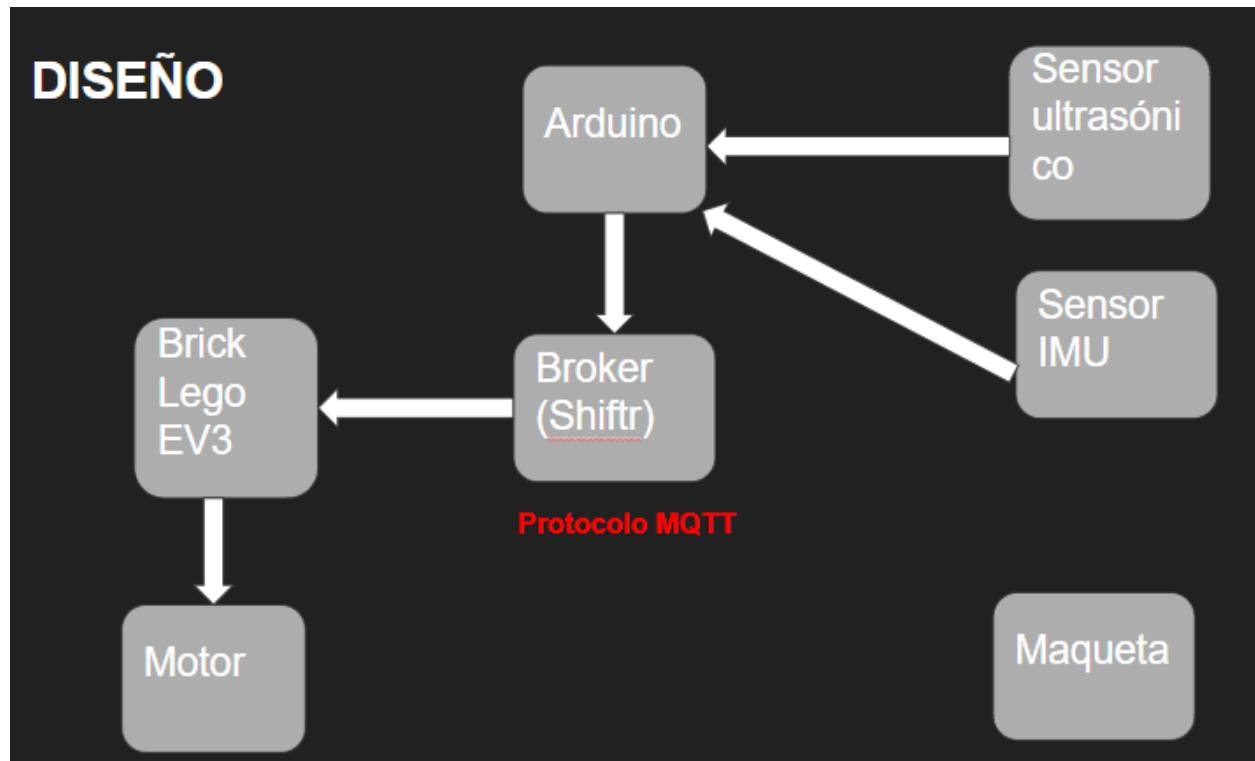
- **Broker:** Se ha usado el broker Shiftr.io para el envío y recepción de mensajes entre el Arduino y el Lego Brick con el motor. Esta comunicación será establecida a través del protocolo MQTT.



7.2- Diseño

En cuanto a la estructura general final de nuestro sistema encontramos un broker en el que mediante protocolo MQTT el arduino publica las órdenes en una serie de topics a los que el Lego Brick está suscrito.

- El Arduino enviará una orden u otra según los inputs que recibe del sensor IMU y el sensor ultrasónico. Todo esto será detallado posteriormente.
- El motor ejecutará una rotación del motor u otra según la orden que reciba en el broker.

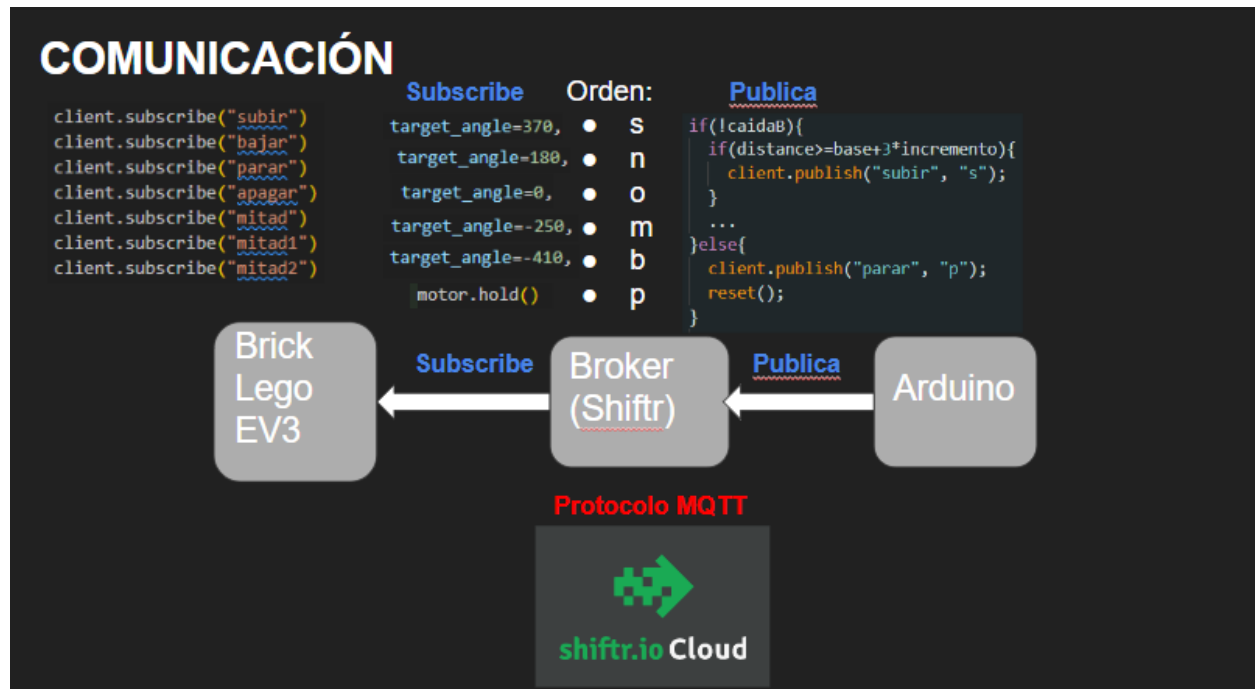


A continuación, detallaremos la estructura de topics de nuestro broker:

- Según la distancia a la que se encuentre la barra o si el sensor IMU detecta que cae la barra, el arduino enviará una orden a un canal u otro.
- De cara a la visualización en la plataforma Shiftr así como para evitar el acoplamiento de varias órdenes distintas en un mismo topic disponemos de 7 topics. El formato de la comunicación es el siguiente, donde según la orden recibida el motor realizará un rotación u otra:

Topic	Orden	Motor
subir	s	target_angle=370
mitad 1	n	target_angle=180
mitad	o	target_angle=0
mitad 2	m	target_angle=-250
bajar	b	target_angle=-410
parar	p	motor.hold()
apagar	a	Guarda ángulo actual en un txt

Este sería un esquema detallado de la comunicación entre el arduino y el Lego EV3:



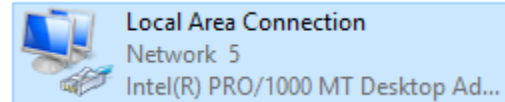
7.3- Programación del motor en el LEGO Brick

A continuación una serie de pasos seguidos de cara a la implementación de la programación del motor:

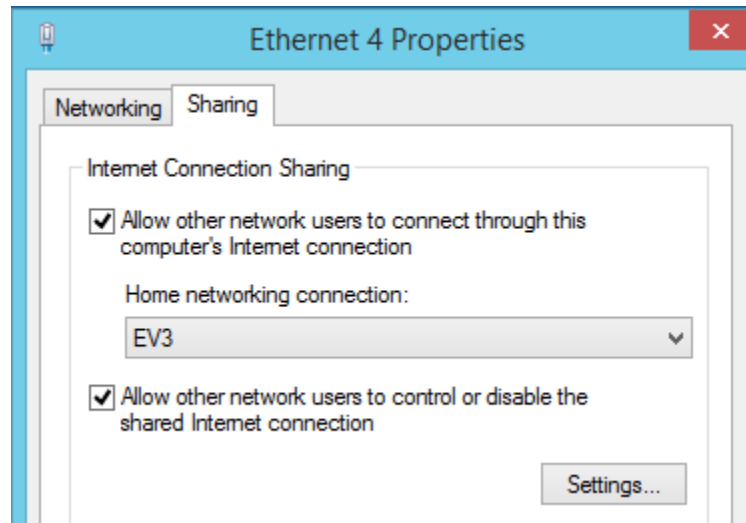
1. Descarga de la imagen ISO de MicroPython para Lego EV3, [enlace de la imagen](#).
2. Flasheo de una tarjeta SD con la ISO mediante el software [Rufus](#).
3. Introducción de la tarjeta en el Lego Brick: Una vez introducida, al encenderla ya tendremos el Lego Brick flasheado.
4. Conexión compartida de Internet mediante cable:
 - a. Conectar el cable al ordenador.
 - b. En dispositivos e impresoras seleccionar el Lego EV3.



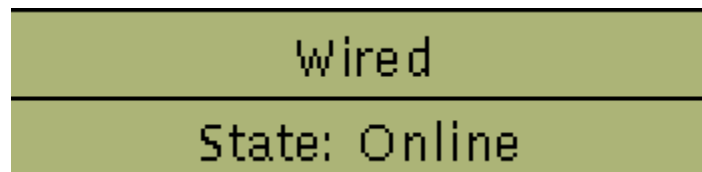
- c. Acceder a la conexión que queremos compartir:



d. Compartir conexión:

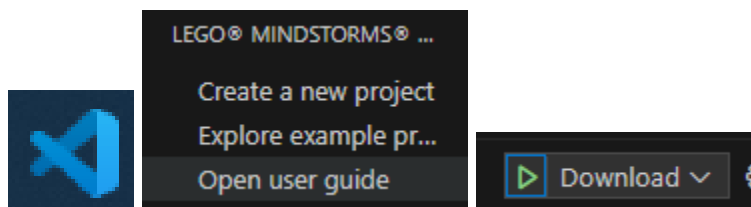


Una vez conectado el Lego Brick a Internet en la sección "All connections"-> wired deberá aparecer Online:



Si solo aparece Connected, no tendrá acceso a internet. Se recomienda resetear y reiniciar el proceso anterior.

5. Instalación de Visual Studio Code y extensión Lego Mindstorms. Para ejecutar el código en el brick haremos click en Download and Run.



6. Programación:

- a. **Conexión con el broker:** Se ha usado la librería `Mqtt.simple` ya que era la única compatible con Micropython.

```
from umqtt.simple import MQTTClient

# Configurar el cliente MQTT
mqtt_server = "platinumvulture693.cloud.shiftr.io"
client_id = "EV3"
username = "platinumvulture693"
password = "VQrRjf9gXs2Exnmi"

# Crear un objeto MQTTClient y conectarse al broker MQTT
client = MQTTClient(client_id, mqtt_server, user=username, password=password)
client.connect()
```

```
client.set_callback(callback)
client.subscribe("subir")
client.subscribe("bajar")
client.subscribe("parar")
client.subscribe("apagar")
client.subscribe("mitad")
client.subscribe("mitad1")
client.subscribe("mitad2")
while True:
    client.check_msg()
```

- b. **Control del motor:** Por cada mensaje recibido en los canales a los que estamos suscritos según la orden que hayamos recibido realizaremos una rotación del motor hacia un ángulo.
- Con la instrucción **then=Stop.HOLD** conseguiremos que cuando el motor llegue al ángulo objetivo se pare y guarde el ángulo en el que nos encontramos.
 - El parámetro **speed=1000** indica la velocidad con la que realizaremos la rotación.
 - Con el parámetro **wait=false** conseguimos que sea interrumpible en mitad de una rotación sin tener que esperar a acabar el desplazamiento. Esto es de especial utilidad para poder cambiar de

dirección durante el seguimiento de la barra. Se ha usado la librería PyBricks.

```
def callback(topic, message):
    #print("Mensaje recibido en el tema {}: {}".format(topic, message))
    orden=str(message)
    ordenProcesada=orden[2:3]
    print(ordenProcesada)
    #ev3.speaker.beep()
    if ordenProcesada=="s":
        motor.run_target(speed=1000, target_angle=370, then=Stop.HOLD, wait=False) #aceleración progresiva
        #motor.track_target(speed=1000, target_angle=-2000, then=Stop.HOLD, wait=False)
        #print(motor.angle())
```

```
    if ordenProcesada=="b":
        motor.run_target(speed=1000, target_angle=-410, then=Stop.HOLD, wait=False) #aceleración progresiva

    if ordenProcesada=="o":
        motor.run_target(speed=1000, target_angle=0, then=Stop.HOLD, wait=False) #aceleración progresiva

    if ordenProcesada=="m":
        motor.run_target(speed=1000, target_angle=-250, then=Stop.HOLD, wait=False) #aceleración progresiva

    if ordenProcesada=="n":
        motor.run_target(speed=1000, target_angle=180, then=Stop.HOLD, wait=False) #aceleración progresiva

    if ordenProcesada=="p":
        You, 3 weeks ago • girando motores ...
        ev3.speaker.beep()
        motor.hold()
        print(motor.angle())
```

```
    if ordenProcesada=="a":
        client.publish("posicionfinal", str(motor.angle()))
        print("Motor apagado, posicion final enviada: ",motor.angle())
        f = open('data.txt', 'w')
        f.write(str(motor.angle()))
        f.close()
```

Cabe destacar que también se implementó una interfaz de prueba del motor usando Java Swing:



```

initComponents();
jButton1.setText("Subir");
jButton1.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        //System.out.println("Encender rojo");
        String mensaje="s";
        try {
            client.publish("subir", new MqttMessage(mensaje.getBytes()));
        } catch (MqttException ex) {
            Logger.getLogger(Interfaz.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
});
jButton2.setText("Bajar");

```

7.4- Programación en Arduino

Para la programación del Arduino se ha instalado arduino IDE donde se hará nuestro programa. Arduino IDE es una aplicación de uso libre que se puede descargar a través de su página web: [ArduinoIDE](https://www.arduino.cc/en/software) (<https://www.arduino.cc/en/software>)

Una vez descargado se puede implementar la lógica del programa dentro de la función `loop()`, la cual es la siguiente:

1. Recibe la distancia del **sensor Ultrasónico** y del **sensor IMU**.

```

duration = pulseIn(echoPin, HIGH);
// Calculating the distance
distance = duration * 0.034 / 2;
sensors_event_t accelerometerData;
bno.getEvent(&accelerometerData, Adafruit_BNO055::VECTOR_ACCELEROMETER);

```

2. Comprueba si la lectura de **IMU** de la barra indica que esta está cayendo.

```

if((x<-umbralCaida || x>umbralCaida)|| (y<-umbralCaida || y>umbralCaida) || (z<-umbralCaida || z>umbralCaida)){
    caidaB=true;
}

```

3. Si no está cayendo, manda la acción de movimiento al canal **MQTT**.

```

if(distance>=base+3*incremento){
    client.publish("subir", "s");
    Serial.print("\n Subir");
}
if(distance>=base+2*incremento && distance<base+3*incremento){
    client.publish("mitad2", "n");
    Serial.print("\n Mitad2");
}
if(distance>=base+1*incremento && distance<base+2*incremento){
    client.publish("mitad", "o");
    Serial.print("\n Mitad");
}
if(distance<base+1*incremento && distance>=base){
    client.publish("mitad1", "m");
    Serial.print("\n Mitad1");
}
if(distance<base){
    client.publish("bajar", "b");
    Serial.print("\n Bajar");
}

```

4. Si está cayendo, manda la señal de parada y se resetea.

```

client.publish("parar", "p");
reset();

```

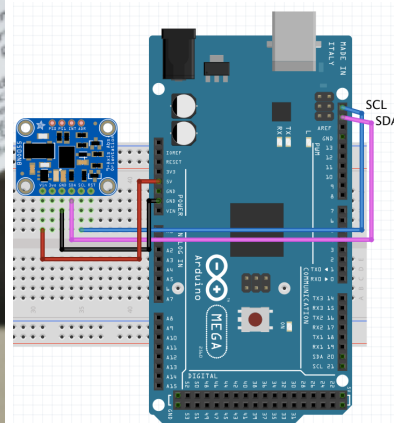
Para la lectura de datos del sensor IMU se ha usado la librería Adafruit que permite obtener el vector de aceleración.

Para conectar el Arduino a los canales de comunicación del servidor **Shiftr.io** es necesario instalar las librerías `WiFi101` y `MQTT`. Posteriormente, en la función `setup()` se llaman a los métodos para que se conecte a la red WIFI y a los métodos para mandar las órdenes de movimiento. También son necesarias las librerías `SPI` y `Wire` para los protocolos de comunicación.

```
// start wifi and mqtt
WiFi.begin(ssid, pass);
client.begin(link, net);
client.onMessage(messageReceived);
```

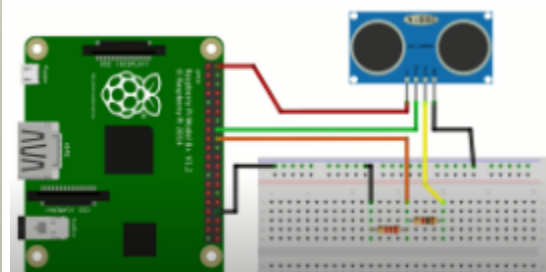
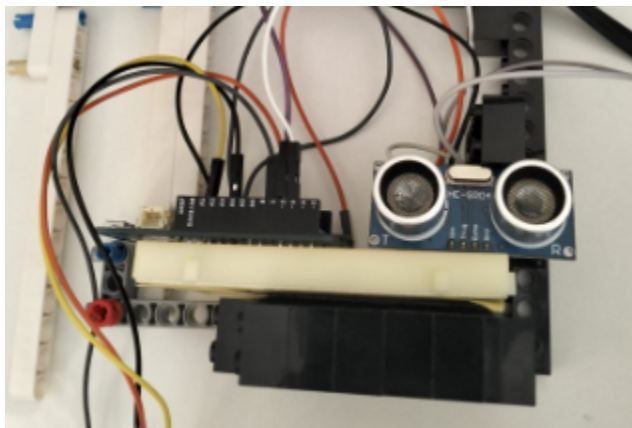
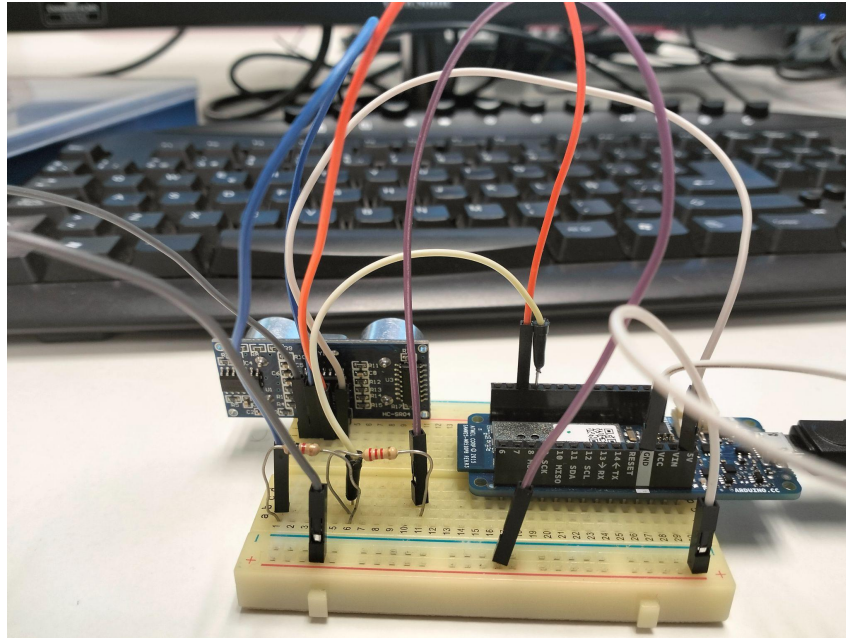
7.5- Construcción

- **Barra:** Para la barra se ha usado la caja de un rollo de papel aluminio, es algo simple pero presenta varias ventajas para este proyecto:
 - Es muy ligera, por lo que no supone un desafío para el motor Lego.
 - Tiene una superficie relativamente ancha y lisa lo que facilita al sensor Ultrasónico la lectura de su posición.
 - El sensor IMU es plano y se acopla bien a la superficie de la caja.



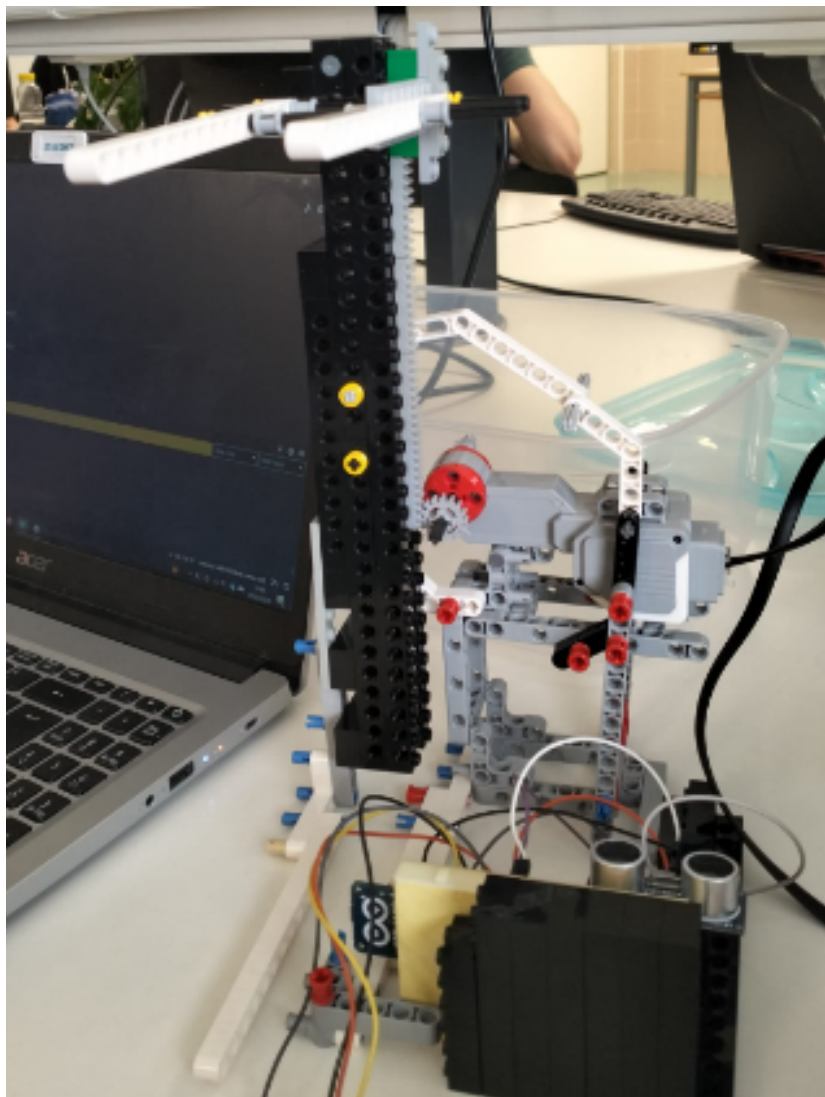
- **Placa arduino:** En esta placa está montado el circuito electrónico del arduino. La cual presenta algunas peculiaridades:
 - El sensor IMU no está integrado en la placa para que se pueda unir a la barra.

- Como es necesario que el sensor Ultrasónico “apunte” hacia la barra, la placa arduino está sujeta y posicionada verticalmente en la línea de movimiento que tendrá la barra.



- **Estructura:** Se usa un sistema de piñón-cremallera con piezas Lego:

- Para que la cremallera se deslice libremente hay que extender la viga o canal donde está la cremallera por debajo del motor. Por ello, también el motor está sobre una plataforma en la mitad de la estructura.
- De la base salen tres “pies” para dar estabilidad y dos de ellos forman la estructura que sujeta la placa con el arduino y sensor Ultrasónico
- Estos dos puntos anteriores forman la parte inmóvil de la estructura. La parte móvil está formada por la cremallera y el brazo que sale de esta. Para poder sujetar la barra esta estructura se ha extendido a un segundo brazo paralelo al original, lo que ofrece 2 puntos de apoyo a la barra.

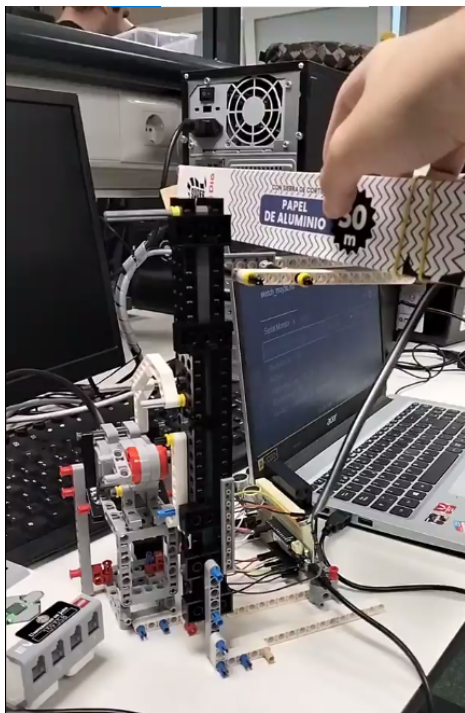


8- Resultados

Finalmente se cumplieron los objetivos principales del proyecto. Esta máquina es capaz de detectar la posición de la barra en tiempo real y mantener los brazos cerca de ella para que en el caso de detectar que la barra cae, se detenga y bloquee el paso. Al realizar esta parada emite un pitido notificando esta acción. Seguidamente, se resetea dejando un tiempo de espera para que el supuesto atleta pueda reaccionar.

Presenta algunos problemas. Uno de ellos es que está diseñada para que tenga un sólo brazo que se ha extendido a uno doble para poder sujetar la barra. Pero para garantizar que no se caiga la barra lo correcto es usar un sólo brazo y duplicar esta estructura. En ese caso la parte software seguiría siendo la misma, sólo cambiaría si se añade un segundo motor si con uno solo no es suficiente para mover los 2 brazos.

La estructura del proyecto no está preparada para soportar cargas elevadas, se puede desarmar e incluso romper. Sin embargo, este proyecto demuestra que es posible construir esta máquina y es escalable a una de mayor tamaño.



[Enlace al vídeo con los resultados obtenidos.](#)

9- Conclusiones

Tras la realización de este proyecto se puede vislumbrar que la Inteligencia Ambiental permite aumentar la seguridad de una persona, en este caso en la realización de ejercicio con pesas. Normalmente de esta tarea se encargaba una persona.

Con este proyecto nos hemos dado cuenta que es importante la experimentación con el uso de los sensores, ya que al exponerlos a un mundo real impreciso y variable, se debe manejar un umbral esperado de error. Por consiguiente, también es clave determinar que ese umbral de error sea aceptable o se requiera de más precisión.

Al trabajar con piezas de Lego hemos tenido que reconstruir el proyecto con frecuencia. Esto es importante a tener en cuenta, ya que las máquinas hay que mantenerlas y no hay que exponerlas a esfuerzos más allá de sus posibilidades.

Debido a nuestra dificultad para seguir la planificación inicial, se pueden concluir algunos errores cometidos de los que aprender. Debemos tener más claro qué se va a hacer y cómo se va a hacer. En un principio no teníamos correctamente definidos los materiales, mecanismos y estructura del proyecto ya que estábamos acostumbrados a trabajar con proyectos físicos.

Para concluir, durante la realización de este proyecto hemos aprendido mucho. En varias ocasiones hemos tenido que investigar para resolver los problemas que se ponían por delante, en lugar de seguir un guión preestablecido como suele ocurrir en la gran mayoría de las asignaturas, logrando así, un mayor acercamiento al mundo laboral. Además, nos ha abierto las puertas a otro tipo de informática menos intangible en la que podemos tocar y visualizar en formato físico lo que programamos.

10- Referencias

Enlace ISO MicroPython - <https://pybricks.com/ev3-micropython/startinstall.html>

Descargar de Software Rufus - <https://rufus.ie/es/>

Conexión WIFI del Lego brick mediante cable - <https://www.ev3dev.org/docs/tutorials/>

Documentación PyBricks - <https://docs.pybricks.com/en/stable/pupdevices/motor.html>

Documentación arduino: <https://docs.arduino.cc/tutorials/>

Sensor Ultrasonico: <https://www3.gobiernodecanarias.org/medusa/ecoblog/fsancac/2>

Documentación Umqtt.simple - <https://mpython.readthedocs.io/en/m>

Broker Remoto Utilizado - <https://www.shiftr.io/cloud>