# 8. Robótica Educativa

# Objetivos y contenidos

Esta parte del curso pretende aplicar la parte teórica (programación) en clases prácticas (robótica) a través de proyectos transversales que incluso pueden permitir la realización de proyectos STEAM interdisciplinares con otras materias como la Física, Biología, Matemáticas, Arte, etc...

Una vez adquiridos los contenidos, el alumnado desarrollará la creatividad y el emprendimiento ya que aporta técnicas y herramientas al alumnado para idear y diseñar soluciones a problemas definidos que tienen que cumplir una serie de requisitos, y lo orienta a la organización de las tareas que deberá desempeñar de manera personal o en grupo.

Financiado por el Ministerio de Educación y Formación Profesional y por la Unión Europea - NextGenerationEU

logo.png

# Materiales

Para la realización de esta parte del curso, se requieren los siguientes materiales, incluidos en el kit de préstamo de CATEDU.

Recibidos (440) - raguado@juandelanuza.org - Correo de Colegio Juan de Lanuza.png

- Arduino UNO
- Sensor de ultrasonidos
- Sensor de presencia PIR
- Servomotor
- Botón/pulsador
- Tira de 8 leds Neopixel
- Conector Gnd
- Conector 5v
- Zumbador
- Cables de conexión (13 macho-hembra, 3 macho-macho, cable USB)

Para simplificar las conexiones, se han incluido los cables necesarios para realizar las conexiones directamente de los sensores/actuadores al Arduino. Igualmente, en cada kit de préstamo, se incluyen dos adaptadores para **Gnd** y **5v** que multiplican estos pines, para que en aquellas practicas en las que se requieren más tomas de las que dispone Arduino, evitar de la utilización de protoboards que complican las conexiones.

En la siguiente fotografía, observamos como al utilizar estos conectores, obtenemos 4 tomas a 5v y otras 4 para Gnd

20221119_072027.jpg

Financiado por el Ministerio de Educación y Formación Profesional y por la Unión Europea - NextGenerationEU

logo.png

# Ejercicios propuestos

A continuación se proponen una serie de ejercicios (situaciones de aprendizaje) que van a permitir al alumnado tener una visión real de como aplicar la programación a la robótica y de este modo resolver situaciones reales. Desde el punto de vista de la **LOMLOE**, se describen a continuación los **saberes básicos**, **competencias específicas** y **criterios de evaluación** que se van a ver implicados en el desarrollo de los mismos.

**Saberes básicos, competencias específicas y criterios de evaluación del bloque de robótica**

**Objetivos didácticos:**

- Conocer y analizar el funcionamiento de los semáforos, distinguiendo el de peatones y el de coches.
- Comprender el funcionamiento de los actuadores en los semáforos y los principios existentes en la electrónica que los constituye.
- Programar mediante software tanto la recogida de datos como el accionamiento de dispositivos físicos de forma que respondan al comportamiento deseado.
- Analizar de forma crítica la irrupción de la automatización de procesos en nuestra cotidianeidad, introduciendo perspectiva accesibilidad e inclusión en dicho análisis.

**Elementos curriculares involucrados (Saberes básicos):**

Bloque A: Proceso de resolución de problemas.

- Estrategias de búsqueda crítica de información durante la investigación y definición de problemas planteados.
- Electricidad y electrónica básica para el montaje de esquemas y circuitos físicos o simulados. Interpretación, diseño y aplicación en proyectos.

Bloque B: Comunicación y difusión de ideas.

- Aplicaciones CAD en dos dimensiones para la representación de esquemas y circuitos electrónicos. (Uso de Tinkercad)

Bloque C: Pensamiento computacional. Programación y robótica.

- Algorítmica y diagramas de flujo.
- Sistemas de control programado. Montaje físico y programación sencilla de dispositivos.
- Fundamentos de la robótica. Montaje, control programado de robots de manera física.
- Autoconfianza e iniciativa: el error, la reevaluación y la depuración de errores como parte del proceso de aprendizaje.

Bloque D: Tecnología sostenible.

- Desarrollo tecnológico: creatividad, innovación, investigación, obsolescencia e impacto social y ambiental. Ética y aplicaciones de las tecnologías emergentes.

**Relación con las competencias específicas:**

- CE.PR.1. Abordar problemas tecnológicos con autonomía y actitud creativa, aplicando conocimientos interdisciplinares y trabajando de forma cooperativa y colaborativa, para diseñar y planificar soluciones a un problema o necesidad de forma eficaz, innovadora y sostenible.
- CE.PR.2. Aplicar de forma apropiada y segura distintas técnicas y conocimientos interdisciplinares utilizando operadores, sistemas tecnológicos y herramientas, teniendo en cuenta la planificación y el diseño previo, para construir o fabricar soluciones tecnológicas y sostenibles que den respuesta a necesidades en diferentes contextos.
- CE.PR.3. Describir, representar e intercambiar ideas o soluciones a problemas tecnológicos o digitales, utilizando medios de representación, simbología y vocabulario adecuados, así como los instrumentos y recursos disponibles y valorando la utilidad de las herramientas digitales, para comunicar y difundir información y propuestas.
- CE.PR.4. Desarrollar algoritmos y aplicaciones informáticas en distintos entornos, aplicando los principios del pensamiento computacional e incorporando las tecnologías emergentes, para crear soluciones a problemas concretos, automatizar procesos y aplicarlos en sistemas de control o en robótica.
- CE.PR.5. Hacer un uso responsable y ético de la tecnología, mostrando interés por un desarrollo sostenible, identificando sus repercusiones y valorando la contribución de las tecnologías emergentes, para identificar las aportaciones y el impacto del desarrollo tecnológico en la sociedad y en el entorno.

**Relación con los criterios de evaluación:**

- **CE.E.1.**

1.1. Idear y diseñar soluciones eficaces, innovadoras y sostenibles a problemas definidos, aplicando conceptos, técnicas y procedimientos interdisciplinares, así como criterios de sostenibilidad, con actitud emprendedora, perseverante y creativa.
1.2. Seleccionar, planificar y organizar los materiales y herramientas, así como las tareas necesarias para la construcción de una solución a un problema planteado, trabajando individualmente o en grupo de manera cooperativa y colaborativa.

- **CE.E.2.**

2.1. Fabricar objetos o sistemas robóticos mediante la manipulación y conformación de materiales, empleando herramientas y máquinas adecuadas, aplicando los fundamentos de estructuras, mecanismos, electricidad y fundamentalmente electrónica, respetando las normas de seguridad y salud correspondientes.

- **CE.E.3.**

3.1. Representar y comunicar el proceso de creación de un producto desde su diseño hasta su difusión, elaborando documentación técnica y gráfica con la ayuda de herramientas digitales, empleando los formatos y el vocabulario técnico adecuados, de manera colaborativa, tanto presencialmente como en remoto.

- **CE.E.4.**

4.1. Describir, interpretar y diseñar soluciones a problemas informáticos a través de algoritmos y diagramas de flujo, aplicando los elementos y técnicas de programación de manera creativa.
4.2. Programar aplicaciones sencillas para distintos dispositivos (ordenadores, dispositivos móviles y otros) empleando los elementos de programación de manera apropiada y aplicando herramientas de edición, así como módulos de inteligencia artificial que añadan funcionalidades a la solución.
4.3. Automatizar procesos, máquinas y objetos de manera autónoma, con conexión a internet, mediante el análisis, construcción y programación de robots y sistemas de control.

- **CE.E.5.**

5.1. Reconocer la influencia de la actividad tecnológica en la sociedad y en la sostenibilidad ambiental a lo largo de su historia, identificando sus aportaciones y repercusiones y valorando su importancia para el desarrollo sostenible.
5.2. Identificar las aportaciones de las tecnologías emergentes al bienestar, a la igualdad social y a la disminución del impacto ambiental, haciendo un uso responsable y ético de las mismas.

**Situación de aprendizaje 9: Semáforo adaptado.**

Programación con Arduino y posterior prototipado de un semáforo que incluya una señal auditiva cuando el semáforo esté en rojo (verde para peatones), para adaptarlo a personas con deficiencia visual.

**Situación de aprendizaje 10: Montando un sistema de aparcamiento**

Programación con Arduino y posterior prototipado de un sistema de aparcamiento por ultrasonidos con avisador visual y sonoro.

**Situación de aprendizaje 11: Creando un sistema de alarma**

Programación con Arduino y posterior prototipado de un sistema de  alarma con sensor de presencia, avisador visual y sonoro y pulsador para detener la alarma.

**Situación de aprendizaje 12: Creando un túnel de lavado con sensores**

Programación con Arduino y posterior prototipado de un túnel del lavado, con semáforo por detección de ultrasonidos para regular la entrada y barrera automática de salida, dotado a su vez de un pulsador de emergencia para detener el proceso por medio de interrupción.

.

Financiado por el Ministerio de Educación y Formación Profesional y por la Unión Europea - NextGenerationEU

logo.png

# Situación de aprendizaje 9: Semáforo adaptado

Programación con Arduino y posterior prototipado de un semáforo que incluya una señal auditiva cuando el semáforo esté en rojo (verde para peatones), para adaptarlo a personas con deficiencia visual.

semaforo.jpg

| | |
|---|---|
| COMPONENTES | **Arduino Uno**<br>**ArduinoUno.png** |
| | **Tira leds NeoPixel**<br>**TIraNeopixel.jpg** |
| | **Zumbador**<br>**rZumbadorrecortado.png** |
| | **Conectores 5v y Gnd**<br>**Recibidos (442) - raguado@juandelanuza.org - C** |

**Trabajo de indagación:** En cualquier proyecto debemos investigar sobre el funcionamiento y comportamiento esperado para nuestro prototipo. En este caso, sabemos que los semáforos adaptados emiten un tono discontinuo cuando los peatones pueden cruzar, posibilitando de este modo a personas con dificultades de visión el paso de un lado a otro de forma segura. Los semáforos mediante código de colores (rojo, ámbar y verde), indican a los conductores/peatones si es momento de esperar o avanzar. La secuencia que siguen los semáforos de conductores es: **rojo**, **verde** y **ámbar**. Cuando esta rojo para coches, el de peatones esta en verde y al contrario. En el caso que nos toca, tendremos

> entonces que contemplar emitir los tonos, cuando el semáforo de coches este en **rojo**.

Para todos nuestros prototipos, vamos a utilizar una **herramienta** de **diseño 2D** con la que representaremos nuestro esquema de conexiones.

### Diseño 2D

Es conveniente utilizar alguna herramienta para diseñar nuestro circuito antes de proceder al montaje. De este modo evitaremos cometer errores con el cableado. Existen muchas herramientas online para este propósito y algunas de ellas como Tinkercad incluso permiten su simulación, pero tienen mayor limitación en las librerías de componentes, por lo que en este caso utilizaremos EasyEDA.

> Puedes encontrar un tutorial para crear tus diseños con EasyEDA pinchando aquí.

EasyEDA(Standard) - A Simple and Powerful Electronic Circuit Design Tool.png

Como podemos observar en nuestro esquemático, vamos a conectar dos actuadores a nuestro Arduino. Ambos tienen un pin a **GND** y otro a **5V**, y cada uno de ellos tiene un pin de datos que asignaremos a los **pines 2 y 3**.

La placa de Arduino tiene 3 pines conectados a GND y uno a 5v, por lo que haremos uso del adaptador incluido en el kit para la conexión a 5v o en su defecto se podría utilizar una placa de prototipado.

Para abordar la programación de este proyecto, en primer lugar **inicializaremos** los componentes y **crearemos 4 funciones**: "*encender verde*"," *encender ámbar*"," *encender rojo*" *y* "*sonido*". De este modo, será tan sencillo como llamar a cada una de ellas en **el bucle principal**, las cuales se repetirán indefinidamente.

### Construcción

Arduino tiene únicamente 1 pin a 5v y 3 a Gnd.

En esta práctica, utilizamos el Zumbador que tiene 3 pines (5v, Gnd y Señal) y la tira led Neopixel que igualmente tiene otros 3 pines (5v, Gnd y datos).

Para alimentar a 5v ambos actuadores, necesitaremos utilizar el conector multiplicador (1 a 4). El pin de datos de la tira led irá según nuestra programación al Pin2 de Arduino y el del zumbador al Pin3 de Arduino. El Gnd del zumbador y el de la tira led, los conectaremos a cualquiera de los 3 pines habilitados para ello en Arduino sin necesidad de utilizar el multiplicador.

20221119_074429.jpg

## Programación - Inicialización de componentes

Representamos lo primero de todo un diagrama de flujo que nos ayude a comprender el comportamiento de nuestro proyecto:

Bitbloq Robotics - Documento sin título.png

El siguiente paso, una vez hemos diseñado nuestro circuito es programarlo. Para ello abrimos el editor web de ArduinoBlocks y nos conectamos con nuestro usuario. Partiendo de un proyecto en blanco eligiendo como placa **Arduino Uno**, lo primero que haremos será sacar en pantalla nuestros componentes.

En primer lugar arrastraremos al área de trabajo el Zumbador, que lo encontraremos dentro del bloque de actuadores.

ArduinoBlocks.png

y haremos lo mismo con la tira led de NeoPixel, que la encontraremos en Visualización, NeoPixel.

ArduinoBlocks (1).png

Con esto ya tenemos todos los componentes que intervienen en nuestro circuito.

Ahora debemos asignarle a cada uno de ellos el Pin al que, en el paso anterior, hemos decidido que debe ir conectado. Para el **Zumbador** tendremos que modificarlo al **Pin3** y deberemos asegurarnos que la Inicialización de la tira **Led** la hacemos sobre el **Pin2**. Además, podemos observar que la tira Led tiene otros 3 parámetros a los que debemos prestar atención. El primero de ellos es la codificación de colores que tiene nuestro componente. En nuestro caso "GRB" (green, red, blue). El segundo de ellos es la frecuencia, que fijaremos en 800Khz, y el

tercero, indica el número de leds que compone la tira. En nuestro caso 8, pero debemos tener en cuenta que la posición del primer led se cuenta como 0, por tanto deberemos indicar 7.

Todas las opciones que podemos realizar con la tira de leds, las podemos encontrar al seleccionar en "Visualización" el apartado "NeoPixel": Las que vamos a utilizar son: *iniciar, establecer brillo, limpiar, mostrar y establecer pixel # a un determinado color.*

- ***Establecer brillo*** **-** Como su nombre indica, haciendo uso de este parámetro vamos a poder determinar la intensidad del brillo de los leds. Su valor puede ir de 0 a 255, siendo 0 la intensidad mínima (apagado) y 255 la máxima. A mayor intensidad de brillo, mayor demanda de corriente tendrá nuestro circuito, por lo que este valor es muy importante a la hora de elegir el método de alimentación de la tira NeoPixel. Cada Led consume unos 60mA (0,3W), dando color blanco intenso (20mA por cada componente de color) Esto supone un consumo de 9W para 30 LED, y 18W para 60 LED, lo que es mucha potencia en una fuente de 5V. Si nuestra tira demanda más potencia que la que puede obtener del propio Arduino, estaremos obligados a alimentar los leds con una fuente externa. Para no tener que recurrir a una fuente externa, en nuestro ejercicio, fijaremos la intensidad del brillo en 10.
- ***Establecer pixel # " " color*** **-** Gracias a este componente podemos determinar, teniendo en cuenta que el primer led ocupa la posición 0, de que color queremos que se ilumine un determinado Led. Si quiero que el led 2 de mi tira de NeoPixel se encienda de color rojo, bastará con configurarlo como: *Establecer pixel # 1 color rojo.*
- ***Mostrar*** - Cualquier configuarción que haga, no será visible hasta que la muestre. Este componente lo utilizaremos cada vez que queramos mostrar un cambio. Para que el Pin 2 del apartado anterior se muestre en color rojo, tendremos que utilizar el bloque "mostrar".
- ***Limpiar*** - Utilizaremos este componente cada vez que queramos eliminar cualquier configuración anterior. Sí quiero que el led 2 de mi tira que anteriormente lo hemos fijado a color rojo se apague, utilizaré este bloque.

Todos estos valores, los actualizaremos y pondremos en el bloque de "Inicializar", para que de este modo, **Arduino** sepa que en el **Pin2** va a tener conectada una tira **NeoPixel** de 8 leds.

ArduinoBlocks (5).png

Para crear funciones, debemos abrir el menú de funciones y arrastrar el bloque al área de trabajo:

ArduinoBlocks (5).png

**Programación - Funciones**

Vemos el código para las funciones. Todas se van a comportar igual salvo la diferencia de los colores, los leds que encendemos y el tiempo de espera para cada color. Definimos los **pines 1 y 2 para** encender en color **verde**. Los **pines 3 y 4 para** el color **ámbar** y por último, los **pines 5 y 6 para** el color **rojo**. El tiempo que permanece la luz de color roja y verde lo definimos en 10 segundos, (10.000 milisegundos) y el de la luz en ámbar lo fijamos en 3 segundos (3.000 milisegundos).

Hay que tener en cuenta que la posición del vector de Leds para BitBlock comienza en 0. Por tanto, el primer led ocupa la posición 0, el segundo la posición 1, y así sucesivamente.

Como nuestra tira tiene 8 pines, en nuestro ejercicio no utilizaremos ni el primero ni el último.

***encender verde:*** *Esablecemos pixeles de las posiciones 2 y 3 a color verde, los mostramos y lo mantenemos así durante 10 segundos.*

***ArduinoBlocks (6).png***

***encender ámbar :*** *Esablecemos pixeles de las posiciones 3 y 4 a color ámbar, los mostramos y lo mantenemos así durante 3 segundos.*

***ArduinoBlocks (7).png***

***encender rojo:*** *Esablecemos pixeles de las posiciones 5 y 6 a color rojo, los mostramos y lo mantenemos así durante 10 segundos.*

ArduinoBlocks (8).png

***sonar zumbador:*** *Teniendo el zumbador conectado al Pin3 de Arduino, emitimos un sonido con frecuencia de 500Hz durante 1 segundo y al finalizar, esperamos 1 segundo.*

***ArduinoBlocks (10).png***

Teniendo ya definidas nuestras funciones, tan solo nos quedaría incluirlas en el Bucle principal para que se repitan indefinidamente, pero para cumplir con las especificaciones del programa, debemos realizar algunas modificaciones para que mientras el semáforo esta en rojo se escuche el sonido del zumbador. Por tanto, tenemos que hacer un llamamiento a la función "*sonar zumbador*" desde

la función "*encender rojo*". Si observamos bien, para mantener las luces rojas, hemos tenido que utilizar una espera de 10 segundos, que ahora podremos sustituir por la espera que se produce cada vez que se produce un sonido por el zumbador (suena 1 segundo y espera en silencio otro segundo). En concreto, cada vez que llamamos a la función del zumbador, transcurren 2 segundos, por lo que si la repetimos 5 veces, estaremos empleando 10 segundos que son los que queremos que la luz roja permanezca encendida mientras escuchamos un "beep" de manera intermitente. La función modificada quedaría de la siguiente manera:

***encender rojo actualizada***

***ArduinoBlocks (11).png***

Con esta actualización ya estariamos cumpliendo los requisitos del enunciado, y nuestro programa finalmente quedaría del siguiente modo:

ArduinoBlocks (6).png

https://www.youtube.com/embed/hXE8xaI2WKQ

Financiado por el Ministerio de Educación y Formación Profesional y por la Unión Europea - NextGenerationEU

logo.png

# Situación de aprendizaje 10: Sistema de aparcamiento para coches

Programación con Arduino y posterior prototipado de un sistema de aparcamiento para coches que incluya un avisador acústico que advierta al conductor del peligro de colisión por cercanía.

sistema de aparcamiento - Búsqueda de Google.png

| | |
|---|---|
| COMPONENTES | **Arduino Uno**<br>**ArduinoUno.png** |
| | **Zumbador**<br>**Zumbadorrecortado.png** |
| | **Sensor Ultrasonidos**<br>**ultrasonido.png** |
| | **Conector 5v**<br>**Recibidos (442) - raguado@juandelanuza.org - Co** |

**Trabajo de indagación:** Gracias a los sensores de ultrasonidos, podemos averiguar la distancia que hay a un objeto. Estos emiten un ultrasonido, reciben el rebote y miden el tiempo que ha transcurrido desde la emisión hasta la recepción permitiéndonos calcular la distancia con la formula velocidad=espacio*tiempo. Utilizando un ultrasonido en la parte trasera de un vehículo, podemos emitir un sonido cuando detectemos una distancia prudencial al objeto y evitar de este modo chocar con él.

## Diseño 2D

Es conveniente utilizar alguna herramienta para diseñar nuestro circuito antes de proceder al montaje. De este modo evitaremos cometer errores con el cableado. Existen muchas herramientas online para este propósito y algunas de ellas como Tinkercad incluso permiten su simulación, pero tienen mayor limitación en las librerías de componentes, por lo que en este caso utilizaremos EasyEDA.

> Puedes encontrar un tutorial para crear tus diseños con EasyEDA pinchando aquí.

Incorporamos el Arduino, buzzer y sensor ultrasónico a nuestro proyecto resultando del siguiente modo:

EasyEDA(Standard) - A Simple and Powerful Electronic Circuit Design Tool (7).png

Puedes encontrar un tutorial para crear tus diseños con EasyEDA pinchando aquí.Como podemos observar, en nuestro esquemático, vamos a conectar un **actuador** (buzzer) y un **sensor** (sensor ultrasónico). Ambos tienen un pin a **GND** y otro a **5V,** El zumbador tiene un pin de salida que conectaremos al número 2 de nuestra placa y en cambio, el sensor de ultrasonidos, tiene 2 pines, uno de salida o Trigger (le pide al sensor que envíe un tono ultrasónico)  que conectaremos al pin 3, y otro de entrada o echo (por el que se recibe el aviso de haber recibido el rebote del disparo inicial) que conectaremos al pin 4 de nuestra placa.

Para abordar la programación de este proyecto, en primer lugar **inicializaremos** el puerto serie que nos servirá para ver en pantalla las medidas del sensor, y crearemos 3 funciones "lejos", "cerca" y "muy_cerca". De este modo será tan sencillo como llamar a cada una de ellas en el bucle principal en función de los rangos de distancia que determinemos para cada uno de estos supuestos.

## Construcción

Arduino tiene únicamente 1 pin a 5v y 3 a Gnd.

En esta práctica, utilizamos el Zumbador que tiene 3 pines (5v, Gnd y Señal) y el sensor de ultrasonidos, que tiene 4 pines (2 de datos, uno para Gnd y el otro para 5v).

Para alimentar a 5v ambos, necesitaremos utilizar el conector multiplicador (1 a 4). Utilizaremos cualquiera de los 3 pines Gnd del Arduino para conectar los Gnd del sensor y del zumbador. El pin de datos del zumbador lo conectaremos al número 2 de nuestra placa y los dos pines (trigger y echo) del sensor de ultrasonidos irán respectivamente al 3 y 4 de la placa de Arduino.

20221119_084032.jpg

**Programación - Inicialización del puerto serie y asignación de pines**

Como venimos haciendo hasta ahora, representamos un sencillo diagrama de flujo que nos ayude a comprender la ejecución del bucle:

Bitbloq Robotics - Documento sin título (1).png

El siguiente paso, una vez hemos diseñado nuestro circuito es programarlo. Para ello abrimos el editor web de ArduinoBlocks y nos conectamos con nuestro usuario. Partiendo de un proyecto en blanco eligiendo como placa **Arduino Uno**, lo primero que haremos será sacar en pantalla nuestros componentes.

En primer lugar arrastraremos al área de trabajo el Zumbador que lo encontramos dentro del bloque de "actuadores".

Zumbador.png

y haremos lo mismo con el sensor ultrasónico.

ArduinoBlocks.png

. Ahora debemos asignarle a cada uno de ellos los Pines que hemos planteado en nuestro esquema electrónico. Para el **zumbador** tendremos que asignarle el **Pin2,** el **Pin3** y **Pin4** respectivamente para los pines Trig y Echo del **sensor de ultrasonidos.**

**ArduinoBlocks (3).png**

El puerto serie del **Arduino** sirve para poder comunicarse con el ordenador y de este modo, además de poder grabar en su procesador nuestros programas, nos sirve para visualizar en pantalla los valores de entradas y salidas que configuremos.

En el caso que nos afecta, dentro del bucle principal, crearemos una variable que contendrá en cada momento el valor del sensor de distancia, que se verá modificada en función de la

distancia a la que se encuentre un objeto de este. Aprovecharemos que este valor se actualiza en esta variable para mostrar su valor por pantalla gracias a la comunicación del puerto serie.

Para **inicializar el puerto serie**, lo haremos directamente sobre el bloque de *Inicializar,* de ArduinoBlocks del siguiente modo:

| | |
|---|---|
| ArduinoBlocks (2).png | ArduinoBlocks (1).png |

Para crear funciones, debemos abrir el menú de funciones y arrastra el bloque al área de trabajo:

funciones.png

**Programación de las funciones**

En el planteamiento de nuestra solución, decidimos implementar 3 rangos de actuación determinados por la distancia a la que se encuentra el objeto. Cuando el objeto esta en la zona "lejos" el zumbador emitirá un tono grave y con una pausa de un segundo, si se encuentra en la zona "cerca", aumentará la frecuencia del tono y disminuirá la pausa a medio segundo y por último, si el objeto se encuentra "muy cerca", el zumbador deberá emitir tonos sin apenas pausa (50 milisegundos) y un tono lo suficientemente agudo como para advertir del peligro.

Por ello configuramos **3 funciones** que llamaremos desde el programa principal en función de que se cumplan dichas condiciones y que quedarán del siguiente modo:

ArduinoBlocks (4).png

Antes de incluir las funciones en nuestro programa principal, vamos a configurar la lógica de actuación del sensor. Para ello, como indicábamos anteriormente, definimos una variable que contenga el valor de las medidas del sensor y la mostramos por el puerto serial para ver su valor.

Para crear variables, seleccionamos en el menú lateral "**Variables**" y seleccionamos el bloque "Crear variable". Estas pueden ser numéricas, de texto o lógicas (verdadera/falsa).

ArduinoBlocks (17).png

Seleccionaremos "Crear variable....(Número)" y le introduciremos por nombre "distancia". A partir de este momento, en el mismo menú lateral, al elegir la opción Variables, podremos ver bloques para establecer o tomar el valor de nuestra variable recién creada.

Para asignarle un valor a nuestra variable "distancia" elegimos el bloque "Establecer [distancia]" y lo asociamos al sensor de ultrasonidos que tenemos definido con los pines 3 Trigger y 4 Echo. De este modo, la información del sensor quedará almacenada en la variable y al estar dentro del bucle principal, cada vez que haya un cambio, quedará almacenado el nuevo valor.

ArduinoBlocks (7).png

En estos momentos podemos subir el programa a nuestro Arduino y conectar el puerto serial para ver los resultados del siguiente modo:

ArduinoBlocks (8).png

ArduinoBlocks (9).png

En estos momentos podemos observar como alejando y acercando un objeto al sensor, los valores de medida cambian. Estos sensores no son muy precisos y es aconsejable determinar cual es la máxima y la mínima distancias en las que es capaz de medir correctamente, para determinar los rangos de actuación.

En el desarrollo de esta practica, con los valores obtenidos se definieron los siguientes rangos:

**muy_cerca-->** entre 0 y 10 centímetros

**cerca-->** de 11 a 30 centímetros

**lejos-->** de 31 a 60 centímetros

Por último solo nos falta programar las condiciones para cada rango y llamar a cada función según proceda.

Para crear las condiciones, utilizamos los bloques que encontramos en **Lógica.**

**ArduinoBlocks (12).png**

Si el valor del sensor (**distancia**) es **mayor que cero** y además es **menor o igual que 10**, debe activarse la función "**muy_cerca**":

Si d**istancia es mayor que 10 y menor o igual que 30** entonces debe activarse la función "**cerca**".

Si **distancia es mayor que 30 y menor o igual que 60** entonces deberá advertirnos de la existencia de un objeto **lejano**.

ArduinoBlocks (13).png

y finalmente nuestro programa completo quedaría del siguiente modo:

ArduinoBlocks (14).png

https://www.youtube.com/embed/w0Hd0M3owI0

Financiado por el Ministerio de Educación y Formación Profesional y por la Unión Europea - NextGenerationEU

logo.png

# Situación de aprendizaje 11: Sistema de alarma (con interrupción)

Programación con Arduino y posterior prototipado de un sistema de alarma para un espacio mediante un sensor PIR, un avisador acústico y un botón de desactivado por medio de interrupción.

alarmaPIr.webp

| | |
|---|---|
| COMPONENTES | Arduino Uno<br>ArduinoUno.png |
| | Sensor PIR<br>PIR.webp |
| | Zumbador<br>Zumbadorrecortado.png |
| | Botón/Pulsador<br>pushbutton.webp |
| | Conectores 5v y Gnd<br>Recibidos (442) - raguado@juandelanuza.org - Correo |

**Trabajo de Indagación**: Un sistema de alarma se basa en la detección del movimiento de la persona que accede a un lugar cerrado y que previamente había sido armado. Si esto

sucede, se dispara la alarma que pone en marcha un plan de emergencia que pasa por el aviso a los servicios de emergencia, si tras haber contactado con el propietario este denuncia no ser él quien esta accediendo al recinto. En caso de ser un error por parte del propietario, este dispone de unos segundos para poder introducir su clave de seguridad y de este modo desarmar la alarma.

La detección del movimiento se obtiene gracias a los sensores piroélectricos o sensores PIR. Un sensor PIR o funciona **comparando la temperatura que desprende un objeto con la de su alrededor, de forma que puede detectar con precisión una presencia en un lugar determinado**. Se tratan de sensores que son los encargados de medir las variaciones de radiaciones infrarrojas que se reciben.

En este reto, utilizaremos por tanto un sensor PIR que será el encargado de enviar una señal al **Arduino** en caso de detectar movimiento. Si esto ocurre, el zumbador comenzará a sonar y solo se detendrá en caso de que el propietario desactive la alarma introduciendo su código secreto. En nuestro caso, simularemos la introducción del número secreto pulsando un botón conectado al sistema de alarmado.

### Diseño 2D

Es conveniente utilizar alguna herramienta para diseñar nuestro circuito antes de proceder al montaje. De este modo evitaremos cometer errores con el cableado. Existen muchas herramientas online para este propósito y algunas de ellas como Tinkercad incluso permiten su simulación, pero tienen mayor limitación en las librerías de componentes, por lo que en este caso utilizaremos EasyEDA.

Puedes encontrar un tutorial para crear tus diseños con EasyEDA pinchando aquí.

El botón que vamos a utilizar tiene integrada una resistencia, lo cual facilita nuestro cableado. Tiene 3 pines (-,+ y S) y para la realización del esquemático, hemos buscado "button pull down", del cual utilizaremos el Pin1 como Vin (5v) el Pin3 como Gnd y el Pin2 como señal de entrada.

En este proyecto contamos con 2 entradas, la señal de activación que proviene del sensor de movimiento que la conectaremos al Pin2 de nuestra placa y la que proviene del botón que irá conectada al Pin3 del Arduino.

Como en los proyectos anteriores, el zumbador, en este caso conectado al Pin4, será una salida.

EasyEDA(Standard) - A Simple and Powerful Electronic Circuit Design Tool (8).png

Como en los ejercicios anteriores, hemos añadido el resto de componentes y una vez cableado queda de la siguiente manera:

EasyEDA(Standard) - A Simple and Powerful Electronic Circuit Design Tool (9).png

Para la realización de este proyecto vamos a utilizar **Interrupciones**. Tomado de **esta** entrada de **Luis LLamas,**

> los microprocesadores incorporan el concepto de interrupción, que es **un mecanismo que permite asociar una función a la ocurrencia de un determinado evento**. Esta función de callback asociada se denomina ISR (Interruption Service Rutine).
>
> Cuando ocurre el evento **el procesador "sale" inmediatamente del flujo normal del programa y ejecuta la función ISR asociada** ignorando por completo cualquier otra tarea (por esto se llama interrupción). Al finalizar la función ISR asociada, el procesador vuelve al flujo principal, en el mismo punto donde había sido interrumpido.
>
> Como vemos, las interrupciones son un mecanismo muy potente y cómodo que **mejora nuestros programas y nos permite realizar acciones que no serían posibles sin el uso de interrupciones**.

Dentro de las interrupciones de hardware, **Arduino es capaz de detectar los siguientes eventos**.

- RISING, ocurre en el flanco de subida de LOW a HIGH.
- FALLING, ocurre en el flanco de bajada de HIGH a LOW.
- CHANGING, ocurre cuando el pin cambia de estado (rising + falling).
- LOW, se ejecuta continuamente mientras está en estado LOW.

Nuestro pulsador se encuentra conectado a 5v y al pulsarlo es cuando cae a 0 voltios. Por tanto, si utilizamos la **interrupción RISING**, podremos observar el flanco de subida de 0 a 5 voltios en el momento de dejar de presionar el botón.

## Construcción

El botón que vamos a utilizar tiene integrada una resistencia, lo cual facilita nuestro cableado. Tiene 3 pines (-,+ y S) y para la realización del esquemático, hemos buscado "button pull down", del cual utilizaremos el Pin1 como Vin (5v) el Pin3 como Gnd y el Pin2 como señal de entrada.

En este proyecto contamos con 2 entradas, la señal de activación que proviene del sensor de movimiento que la conectaremos al Pin2 de nuestra placa y la que proviene del botón que irá conectada al Pin3 del Arduino.

Recibidos (445) - raguado@juandelanuza.org - Correo de Colegio Juan de Lanuza.png

Como en los proyectos anteriores, el zumbador, en este caso conectado al Pin4, será una salida.

Finalmente, como ha ocurrido en los ejercicios anteriores, necesitaremos utilizar el conector 1 a 4 de 5v para poder alimentar el zumbador, el sensor PIR y el botón. En cuanto a las tomas de tierra (Gnd) podemos utilizar las 3 disponibles en la placa.

## Programación - Inicialización de variables e Interrupción

Una vez más representamos en un diagrama de flujo nuestro reto para simplificar su programación.

| | |
|---|---|
| Bitbloq Robotics - Documento sin título (2).png | Bitbloq Robotics - Documento sin título (3).png |

Utilizaremos la variable "alarma" para almacenar en ella el momento en el que la alarma se dispara por intrusión. Para crear la variable, recuerda que debes acceder en el menú lateral de **ArduinoBlocks** a la sección "Variables" y en este caso elegiremos una variable booleana (verdadera o falsa). La inicializaremos como falsa, y obligaremos a que cambie a verdadera si el sensor de presencia informa de una intrusión. Así mismo, estableceremos la Interrupción RISING en el Pin3. Arduino Uno tiene únicamente dos pines que admiten interrupciones. Estos solo pueden ser el 2 y el 3.

ArduinoBlocks (15).png

Cuando la alarma se active por una intrusión, el zumbador comenzará a sonar mientras la variable "alarma" sea verdadera. AL entrar en la interrupción, la pondremos nuevamente en estado falso y esto hará que el sonido se pare.

Veamos como implementamos esto con los bloques de **ArduinoBlocks**.

ArduinoBlocks (16).png

Como habíamos mencionado, el sensor PIR lo tenemos conectado en el Pin2. Si éste, detecta movimiento, enviará una señal (HIGH) al **Arduino,** y la almacenamos en la variable "**movimiento** ": Al igual que con la variable "alarma", definiremos la variable "movimiento" como booleana. El valor de esta, dependerá de si el sensor envía estado alto, lo cual la establecerá a **verdadera**, o si por el contrario, no hay movimiento y se encuentra siempre en estado bajo, configurándola como **falsa**.

Con esta información, bastará con utilizar un bloque "Si <condición> es verdadera, entonces hacer..." que lo podemos encontrar en el menú que se despliega al pinchar en "Lógica". Para nosotros la condición es sencilla: Si movimiento es verdadero, entonces modificamos el valor de la variable "alarma" de falso a "verdadero".

En la siguiente instrucción, introduciremos otro bloque lógico, de manera que: **Si** la variable " **alarma**" es **verdadera**, **entonces** el **zumbador** comenzará a **emitir un tono**.

Cuando el programa vuelve a la primera instrucción del bucle principal, ya no importa si el sensor sigue notando movimiento o no, puesto que la variable "alarma" queda ya en estado verdadero y siempre se cumplirá la última condición, por lo que el zumbador no dejará de sonar a menos que "alarma" cambie a "falso" y esto solo sucederá si se activa la interrupción pulsando el botón que simula que introducimos la contraseña para desarmar la alarma. Si el botón es pulsado, "alarma" pasa a almacenar nuevamente el valor "false" y por tanto, si no hay movimiento en el interior, no se cumplirá la última condición y por tanto el zumbador no emitirá ningún sonido. Visionando todos los bloques, quedarían del siguiente modo:

ArduinoBlocks (18).png

Antes de dar por terminado este ejercicio, nos planteamos una mejora, y es que tal y como lo hemos configurado, una vez armada la alarma, cuando volvamos a nuestro local, nada más entrar la alarma se disparará. Vamos a modificar nuestro código para que al detectarse una intrusión, durante unos segundos emita unos tonos de advertencia y si en ese intervalo no se desarma, entonces es cuando comenzará a sonar la sirena.

Para ello, si el sensor detecta una intrusión, a parte de establecer "alarma" como verdadero, utilizaremos un bloque de repetición para emitir un tono muy corto, y si en ese intervalo se pulsa el botón, nuestra variable "alarma" volverá a ser falsa y no sonará la alarma. Por el contrario, si se

trata realmente de un intruso, al no saber la contraseña, finalizado el bucle de repetición de sonidos cortos, la alarma comenzará a emitir dos tonos similares a los de una sirena.

ArduinoBlocks (19).png

https://www.youtube.com/embed/MYiCP2Xnr5I

Financiado por el Ministerio de Educación y Formación Profesional y por la Unión Europea - NextGenerationEU

logo.png

# Ejercicio Final: Túnel de lavado con sensores

## Programación con Arduino y posterior prototipado de un sistema de lavado automático para coches.

tunellavado.png

| | |
|---|---|
| COMPONENTES | Arduino UNO<br>image-1668252879858.jpg |
| | Sensor de Ultrasonidos<br>ultrasonido.png |
| | Tira leds NeoPixel<br>zoomNeopixel.jpg |
| | ServoMotor<br>servocurso.jpg |

El acceso al túnel estará regulado por un semáforo (2 leds de la tira Neopixel) que permanecerá en verde hasta que el vehículo se acerque justo a la posición donde debe parar. En este instante el semáforo cambiará a rojo, y comenzará el proceso de lavado.

Durante el tiempo de lavado (6 segundos), el resto de leds de la tira, se irán encendiendo en color azul, uno a uno cada segundo, indicando de este modo el progreso del servicio.

Finalizado el proceso de limpiado y secado, la barrera de salida (un servomotor en posición 0 grados), se abrirá (posición 90 grados) para dejar salir al vehículo, y el semáforo volverá a ponerse en verde.

Al tratarse del ejercicio final, únicamente vamos a ver como accionar un servomotor, ya que se pretende ver como el alumnado ha adquirido los conocimientos, y el resto de componentes han sido ya tratados en las experiencias anteriores.

### Diseño 2D

Es conveniente utilizar alguna herramienta para diseñar nuestro circuito antes de proceder al montaje. De este modo evitaremos cometer errores con el cableado. Existen muchas herramientas online para este propósito y algunas de ellas como Tinkercad incluso permiten su simulación, pero tienen mayor limitación en las librerías de componentes, por lo que en este caso utilizaremos EasyEDA.

> Puedes encontrar un tutorial para crear tus diseños con EasyEDA pinchando aquí.

En este caso, como indicaba al principio, vamos a implementar únicamente el diseño de las conexiones del servomotor. El resto del diseño forma parte de la tarea evaluable de este curso.

Los servomotores tienen 3 Pines. Positivo, Gnd y datos. A continuación muestro el servo que he utilizado y como quedan las conexiones.

EasyEDA(Standard) - A Simple and Powerful Electronic Circuit Design Tool (10).png

EasyEDA(Standard) - A Simple and Powerful Electronic Circuit Design Tool (11).png

A continuación vemos como programar el movimiento de un servo con **ArduinoBlocks.**

### Programación - Servomotor

Los servomotores pueden mover su posición de 0 a 180 grados.  En ArduinoBlocks, hay un apartado de "Motores" donde podremos encontrar los servos. Lo arrastramos al campo de trabajo y asignamos su pin de datos.

ArduinoBlocks (20).png

Financiado por el Ministerio de Educación y Formación Profesional y por la Unión Europea - NextGenerationEU

logo.png