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 ArduinoUno.png
	Sensor PIR PIR.webp
	Zumbador Zumbadorrecortado.png
	Botón/Pulsador pushbutton.webp
	Conectores 5v y Gnd 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

Revision #11
Created 2022-10-15 09:13:56 CEST by Ricardo Aguado Vallejo
Updated 2023-01-17 15:53:37 CET by Equipo CATEDU