Índice de imágenes

Contenido

[Imagen 1: Protoboard y pulsador (Fuente: Fritzing) 2](#_Toc156932318)

[Imagen 2: Pines, (PI, PII, PInS y PS), y LED conectado 3](#_Toc156932319)

[Imagen 3: Colocando la resistencia 4](#_Toc156932320)

[Imagen 4: Definiendo pin de entrada 5](#_Toc156932321)

[Imagen 5: Definiendo pin de salida 5](#_Toc156932322)

[Imagen 6: Definiendo la lectura del botón en el pulsador 6](#_Toc156932323)

[Imagen 7: Completando el “if” 7](#_Toc156932324)

Fuente:

<https://www.youtube.com/watch?v=orbNupI92KU&list=PLEzmH7aN82FEh2JjYuNCvFu6wFolHai32&index=7>

El Arduino puede controlarse con software externo mediante distintos componentes que sean compatibles con él, de forma que el Arduino pueda leer esos componentes y poder realizar diversas acciones.

Algunos de dichos componentes son los protoboard y los pulsadores:

* **Pulsador:** Posee 2 filamentos que actúan como conductores. Se ha de tener mucho cuidado a la hora de conectarlo al protoboard ya que en función de cómo lo pongamos podrá pasar o no la electricidad. Para que pueda pasar la electricidad deberá colocarse un filamento en C, (common), y otro en NC, (normally close). Si lo ponemos en NA, (normally open), no pasara la corriente.
* **Protoboard:** Es una placa que une los pines entre si para evitar tener que soldar y desoldar los componentes auxiliares, (resistencias, condensadores, …)

En la Imagen 1 podemos ver un ejemplo de protoboard y de pulsador:

|  |
| --- |
|  |
| Imagen : Protoboard y pulsador (Fuente: Fritzing) |

Al no disponer de los componentes físicos para realizar el proyecto, explicaré de forma teórica cómo funciona. Si nos fijamos en la Imagen 1 vemos que en la protoboard hay 2 filas superiores, (delimitados por la línea roja), 2 filas inferiores, (delimitados por la línea azul) y 8 filas de pines separadas 4 a 4 y delimitadas por una línea más gruesa de color gris.

Para mejor comprensión de esta parte se denominará a las 2 filas superiores de pines como pines superiores, (PS), a los 2 filas inferiores como pines inferiores (PI), a las 4 filas intermedias más cercanas a los pines superiores como pines intermedios superiores, (PInS) y a las 4 filas intermedias más cercanas a los pines inferiores como pines intermedios inferiores (PII).

La conectividad de los pines en la protoboard funciona de la siguiente manera:

* **Horizontal:** Los PI están conectados de forma horizontal entre sí. Lo mismo ocurre con los PS.
* **Vertical:** Los PInS están conectados entre sí de forma vertical. Lo mismo ocurre con los PII.

Sabiendo de la forma en la que pasa la corriente entre los pines podemos conectar un LED de la siguiente forma:

|  |
| --- |
|  |
| Imagen : Pines, (PI, PII, PInS y PS), y LED conectado |

En la Imagen 2 vemos como el LED es conectado por uno de sus filamentos a uno de los PInS y por el otro a uno de los PII. Para poder limitar la corriente que circula por el LED es necesario colocarle una resistencia en serie.

Para ello uno de los extremos de la resistencia, (R en la imagen), ira en la misma fila de pines que vaya colocado uno de los extremos del LED. El otro extremo ira en uno de los PI para que la resistencia cumpla su función. Tal y como se muestra en la Imagen 3:

|  |
| --- |
|  |
| Imagen : Colocando la resistencia |

Traduciendo la Imagen 3 al Arduino lo que estamos haciendo es pasar la corriente del Arduino a la placa con el cable rojo y el pin de 5 V y con el pin GND y el cable negro pasamos los 0 V de los PI a los PS. La resistencia hace de vehículo para conectar los pines con el pulsador.

Este sistema se monta de esta manera para que cuando el pulsador este abierto detecte GND y cuando este cerrado detecte 5 V.

Una vez montado este sistema, quedaría programarlo en Arduino.

Lo que quedaría sería programar en el IDE el encendido y apagado del LED. Para ello nos vamos al IDE de Arduino y abrimos un nuevo archivo, (File 🡪 New sketch). Como ya se ha explicado anteriormente por el “Setup()” se define las entradas y salidas. En este ejercicio se ha escogido el pin 2, (digital), como entrada. Lo que queremos es leer la electricidad que esta pasando por ese pin. Con lo que la línea de comandos inicial sería:

|  |
| --- |
|  |
| Imagen : Definiendo pin de entrada |

Debido a la colocación del LED, el pin de salida de voltaje es el 13 por lo tanto en el IDE se definiría como:

|  |
| --- |
|  |
| Imagen : Definiendo pin de salida |

Definidos los pines de entradas y salidas, pasamos a lo que queremos que el programa ejecute que en un primer paso sería la lectura del pin 2. Para escribir un pin digital era “digitalWrite()”, pues para leerlo es “digitalRead()”. Leemos el pin 2, (“digitalRead(2)”) y lo guardaremos usando una **variable**.

Una variable en Arduino sirve para almacenar información y según el tipo de información que almacene se dividen en:

* **int:** Sirve para almacenar número enteros, (4, -5, 9,0, …)
* **Float:** Es una variable flotante y se usa para números decimales, (4’0, 5’5, -6’9, 0’0, …)
* **String:** Sirve para texto, (“8”, Solo yo puedo hacer este informe, …)
* **Boolean:** Son para verdadero y falso, (0 y 1, True y False, …)

Y existen muchos otros tipos, pero estas 4 son las principales. Es necesario saber esto, ya que se debe especificar que tipo de variable es la información que vamos a guardar. Quedaría como en la Imagen 6:

|  |
| --- |
|  |
| Imagen : Definiendo la lectura del botón en el pulsador |

Lo definimos como “int” porque es un entero ya que pueden ser 0 V o 5 V. Con el estado del botón definido, tendremos que especificarle a Arduino que ocurriría si el botón está encendido y que ocurriría si está apagado. Para ello debemos de usar un “**if**” y para definirlo habría que hacer lo siguiente:

1. Escribir la palabra “if”, (especificamos que hacemos un condicional).
2. A continuación definimos en qué estado está el botón, (ON/OFF). Para ello usamos “= =”, en lugar de =, ya que para igualdad se usa =, y para validar se usa = =. Se escribiría como:

Los corchetes son necesarios para definir la acción que llevaría a cabo el Arduino de estar a 5 V o 0 V.

1. En las acciones habría que decirle a Arduino que encienda el LED cuando el botón este pulsado y que lo apague cuando no lo esté. Para ello, usamos el “digitalWrite()” explicado anteriormente.

Quedaría como en la Imagen 7:

|  |
| --- |
|  |
| Imagen : Completando el “if” |

Para no liar más no explicare el “else”, pero es una opción para acortar código, (si el botón esta encendido ocurre esto, si no lo está, (else), ocurre esto otro).

El montaje físico se haría de la siguiente manera, (todo el proceso debe de hacerse con la placa **DESCONECTADA**):

Conectamos el cable negro al GND, (en el Arduino), y a uno de los PInS o PII. Al estar conectado a uno de esos pines, estará conectado también a toda esa vertical, (tal y como se explica en la Imagen 2).

Por tanto la resistencia no podría colocarse en esa hilera vertical de pines. Bastaría con colocarla en otra hilera distinta. Enganchamos la resistencia en dos hileras de PInS o de PII distintas de donde está el cable que va al GND.

El pulsador debe ponerse en la misma vertical que la resistencia de esa manera la electricidad pasara.

Falta documentar montaje físico (De la página 8 de este documento en adelante)