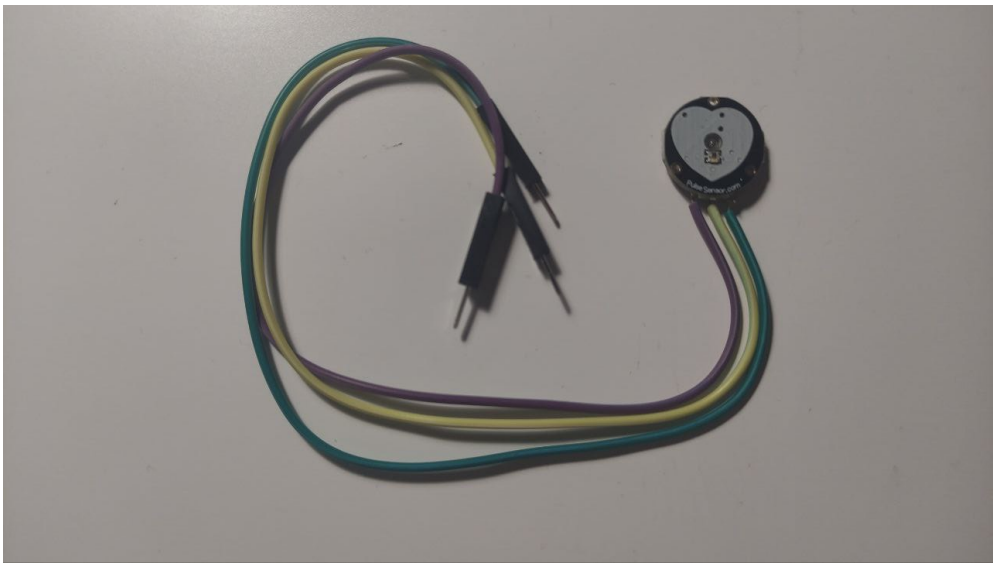
Práctica 9.1 Medimos nuestras pulsaciones

El pulsómetro

Uno de los sensores que vienen con nuestro kit es este:

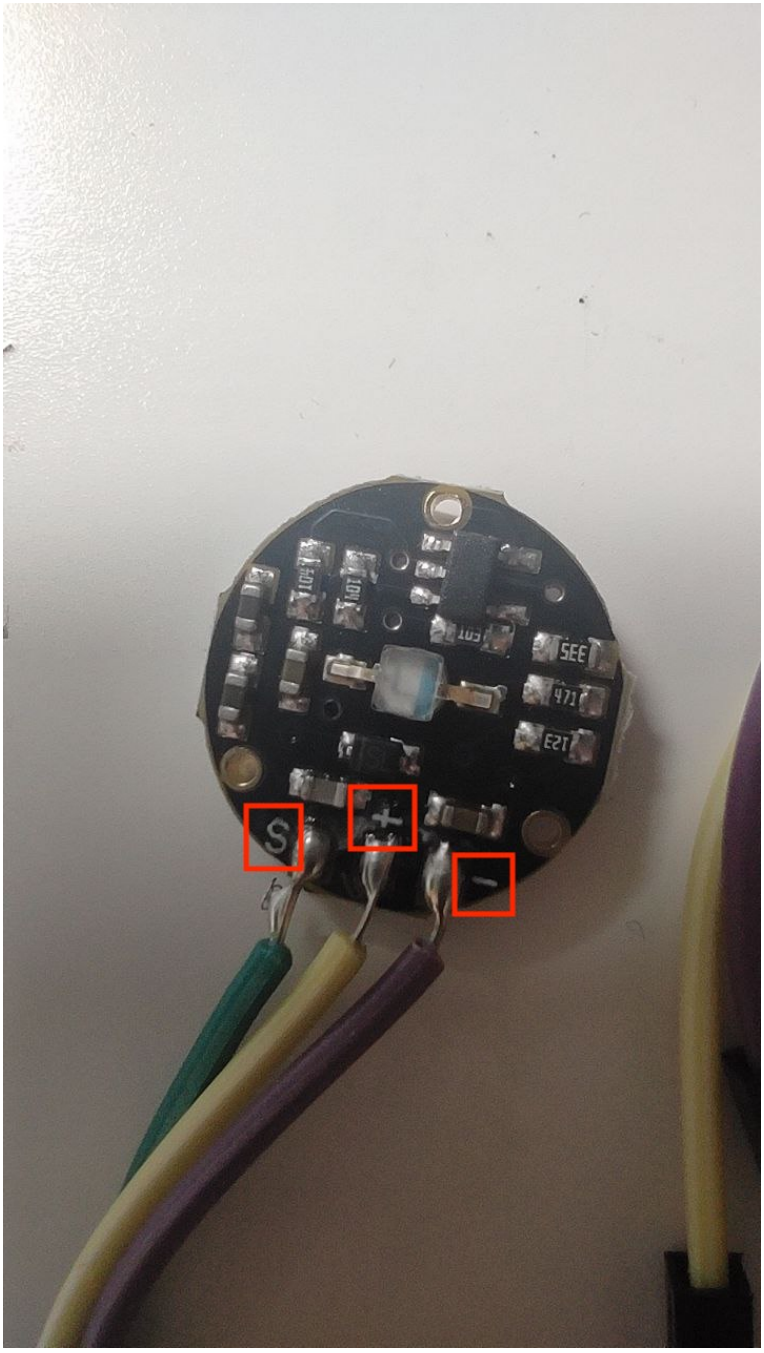


Este sensor se va a encargar de detectar nuestras pulsaciones. Aunque no vamos a entrar a explicar su funcionamiento.

> Si quieres saber más, puedes hacer click en este enlace.

Para detectar nuestras pulsaciones, tendremos que colocar nuestro dedo pulgar sobre la imagen del corazón blanco, sin presionar muy fuerte. Conectarlo a nuestro Arduino no puede ser más sencillo. Si miras en el lado contrario en el que está dibujado el corazón, te encontrarás esto:

Copyright 2025



Verás que, probablemente, el color de los cables de tu pulsómetro es distinto al mío, pero eso no importa. Te tienes que fijar en las letras que hay escritas al lado de cada cable.

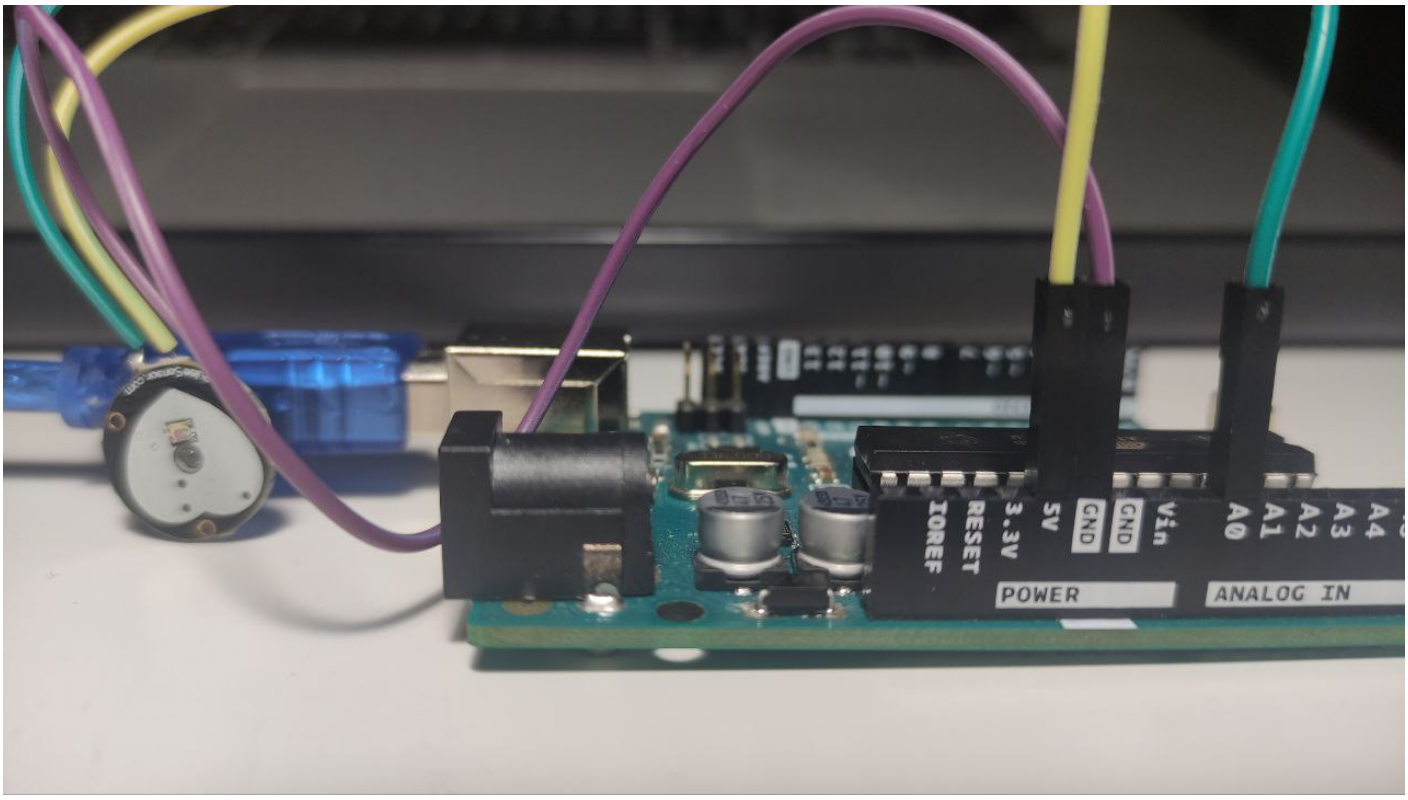
La letra **S** es la que e encarga de transportar la señal desde nuestro sensor hasta Arduino y para esta práctica irá conectada a nuestro UNO en el pin **A0**.

El símbolo **+** es el que se encarga de alimentar a nuestro Arduino e irá en el pin **5V**.

Por último, el signo **-** es el que va a masa (ground) y en nuestro UNO irá conectado a uno de los dos pines **GND**.

Copyright 2025

Las conexiones nos quedarán como las de esta imagen:



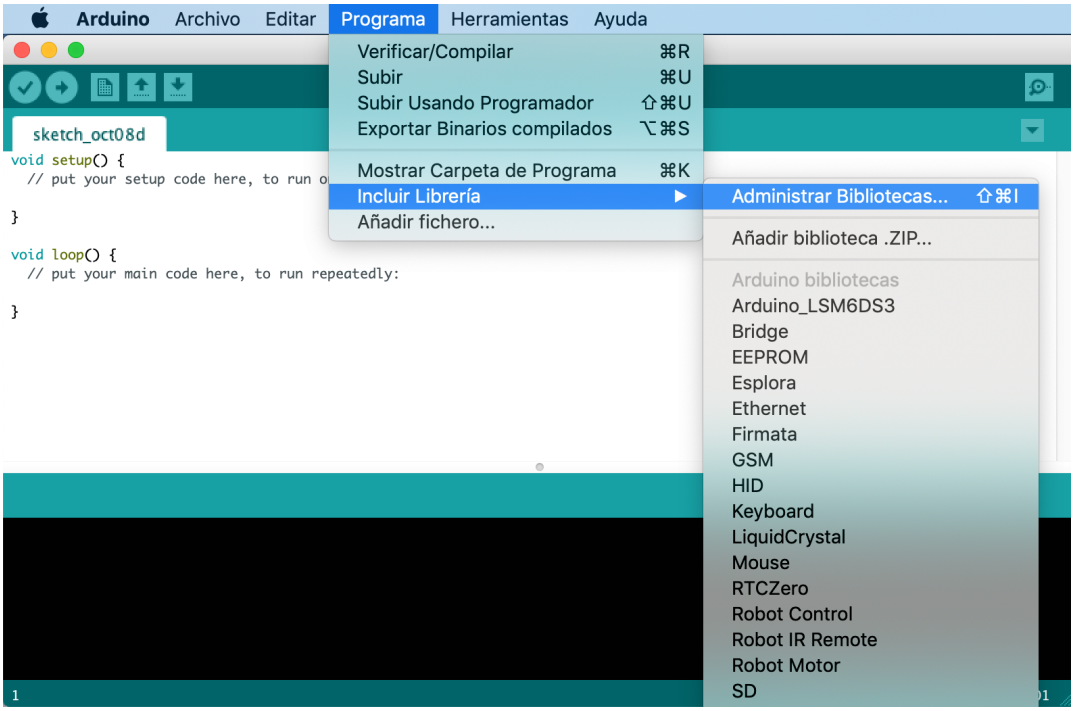
Una vez hemos conectado nuestro sensor a Arduino, pasaremos a ver qué tenemos que hacer en la IDE de Arduino para hacerlo funcionar.

La librería Pulse Sensor Playground

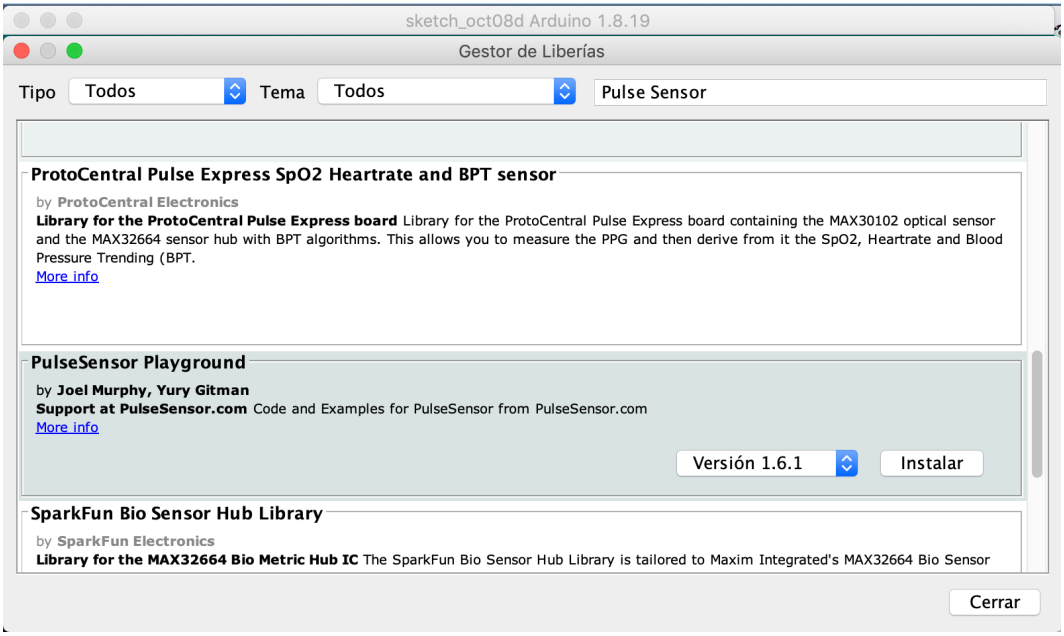
Como ya sabes, una librería (también te la puedes encontrar denominada como biblioteca) de Arduino es una colección de código y ejemplos sobre un tema o dispositivo específico. Por ejemplo, para controlar el sensor que vamos a utilizar en esta práctica, un **pulsómetro**, vamos a emplear la biblioteca/librería **PulseSensor Playground.** En ella, vamos a encontrar una **colección de código y proyectos** creados exclusivamente para facilitar el uso de este sensor junto con Arduino.

Lo primero que haremos será instalar esta librería. Para ello, iremos a **Programa > Incluir Librería > Administrar Bibliotecas...** :

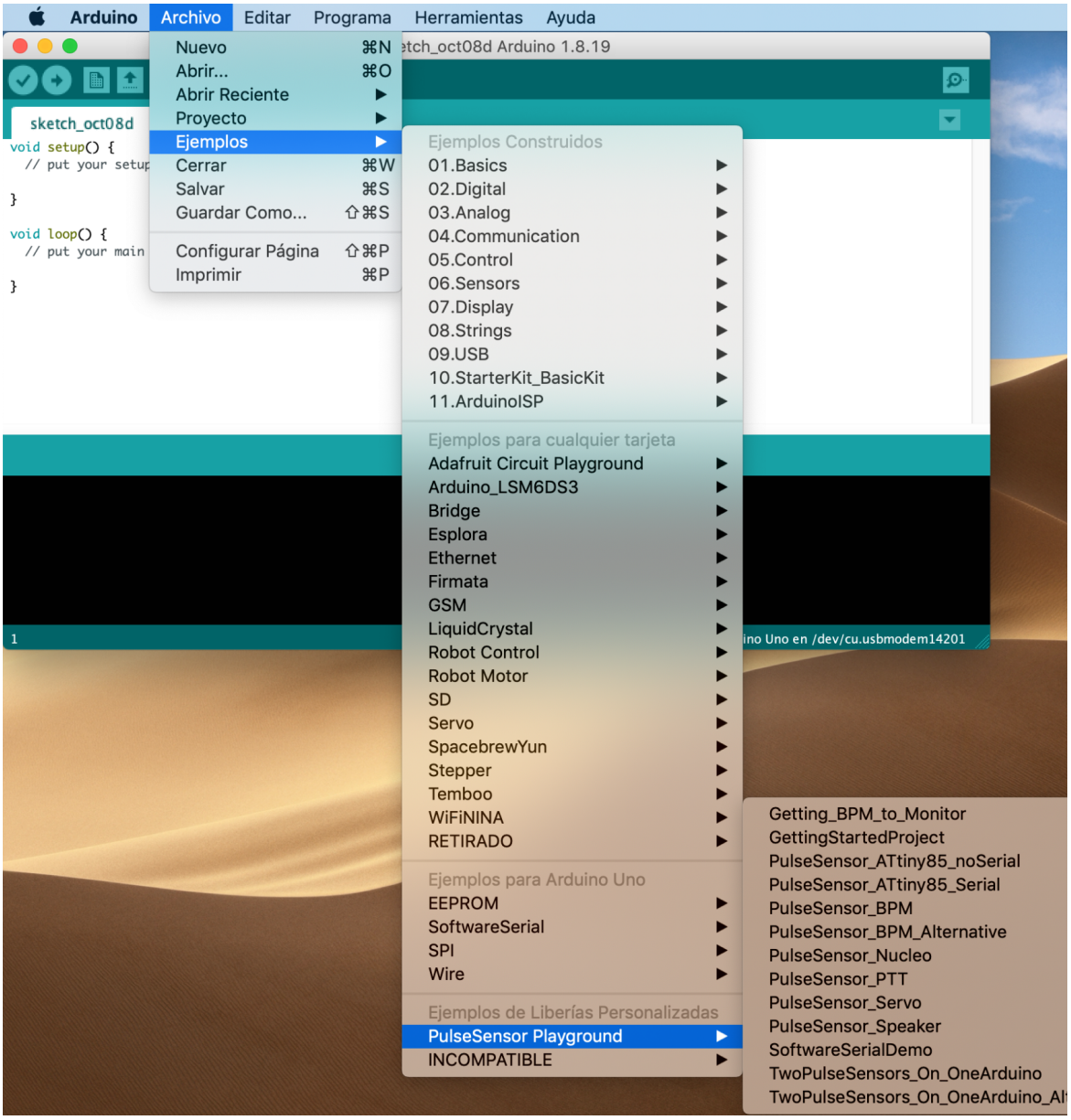
Copyright 2025



Una vez ahí, haremos click y nos aparecerá una ventana emergente en la que podremos buscar la librería, que en este caso se llama **PulseSensor Playground**, cuando aparezca, pondremos el ratón sobre ella y nos aparecerá el botón **Instalar**. La instalaremos:



Cerraremos la ventana emergente y veremos que si vamos a **Archivo > Ejemplos** nos aparecerán los ejemplos de la librería **PulseSensor Playground**:



Una vez la hayamos instalado, abriremos el ejemplo **GettingStartedProject**. Y pasaré a explicarte lo que hace cada línea de código.

¡Pulsación detectada ?!

Lo primero que voy a hacer es poner aquí el código, aunque imagino que ya lo tienes abierto en tu IDE de Arduino:

```
/*  PulseSensor Starter Project and Signal Tester
 *  The Best Way to Get Started  With, or See the Raw Signal of, your PulseSensor.com™ &
Arduino.
 *
 *  Here is a link to the tutorial
 *  https://pulsesensor.com/pages/code-and-guide
 *
 *  WATCH ME (Tutorial Video):
 *  https://www.youtube.com/watch?v=RbB8NSRa5X4
 *
 *
------------------------------------------------------------
1) This shows a live human Heartbeat Pulse.
2) Live visualization in Arduino's Cool "Serial Plotter".
3) Blink an LED on each Heartbeat.
4) This is the direct Pulse Sensor's Signal.
5) A great first-step in troubleshooting your circuit and connections.
6) "Human-readable" code that is newbie friendly."

*/


//  Variables
int PulseSensorPurplePin = 0;        // Pulse Sensor PURPLE WIRE connected to ANALOG PIN 0
int LED13 = 13;   //  The on-board Arduion LED


int Signal;                // holds the incoming raw data. Signal value can range from 0-1024
int Threshold = 550;            // Determine which Signal to "count as a beat", and which to
ingore.


// The SetUp Function:
void setup() {
  pinMode(LED13,OUTPUT);         // pin that will blink to your heartbeat!
```

```
   Serial.begin(9600);         // Set's up Serial Communication at certain speed.

}

// The Main Loop Function
void loop() {

  Signal = analogRead(PulseSensorPurplePin);  // Read the PulseSensor's value.
                                              // Assign this value to the "Signal" variable.

   Serial.println(Signal);                    // Send the Signal value to Serial Plotter.


   if(Signal > Threshold){                          // If the signal is above "550", then
"turn-on" Arduino's on-Board LED.
     digitalWrite(LED13,HIGH);
   } else {
     digitalWrite(LED13,LOW);                //  Else, the sigal must be below "550", so
"turn-off" this LED.
   }


delay(10);


}
```

Ahora, vamos a ir desmenuzando qué es lo que hace el código:

Todo el primer bloque, recogido entre los símbolos **/* */** es un comentario. No volveremos a explicar su función porque ya lo vimos en este apartado.

Pasaremos directamente a los párrafos que contienen las variables:

```
//  Variables
int PulseSensorPurplePin = 0;        // Pulse Sensor PURPLE WIRE connected to ANALOG PIN 0
```

```
int LED13 = 13;   //  The on-board Arduion LED



int Signal;                // holds the incoming raw data. Signal value can range from 0-1024
int Threshold = 550;            // Determine which Signal to "count as a beat", and which to
ingore.
```

Para hacer funcionar nuestro proyecto, necesitaremos crear 4 variables de números enteros (int). La primera de ellas **PulseSensorPurplePin** tiene el valor **0**, ya que va conectado el pin analógico **A0**. Aparece el color **purple**, porque en el sensor para el que en origen se creó este ejemplo, el cable que se conecta a A0 era morado.
La variable **LED13** tiene el valor 13, ya que es el pin al que va conectado el pin que va integrado en la placa de nuestro UNO.
Signal almacenará el valor que nuestro sensor produce cuando está intentando detectar una pulsación. Como se trata de un sensor analógico, la señal se corresponderá a valores discretos, dentro del rango **0 hasta 1024**.
Por último, **Threshold** será el umbral a partir del cual podremos afirmar que se ha producido un latido, una pulsación, en este caso será **550**. Ya habíamos dicho que el umbral de nuestra señal iba de **0 a 1024**, por lo que utilizar 550 es una buena referencia.

A continuación, vamos a pasar a ver la función **setup():**

```
// The SetUp Function:
void setup() {
  pinMode(LED13,OUTPUT);         // pin that will blink to your heartbeat!
   Serial.begin(9600);         // Set's up Serial Communication at certain speed.


}
```

Es bastante breve y en ella lo único que necesitamos incluir es el modo del pin que vamos a utilizar gracias a la función pinmode(). En ella indicaremos el pin y si va a ser de **entrada** (**INPUT**) o de **salida** (**OUTPUT**). Como en este caso es un LED, le diremos que es de salida. También le diremos a nuestro UNO a que velocidad tiene que enviar la información a través del puerto serial, que en este caso será **9600 baudios.**

> Recuerda que todo lo incluido en la función setup() solamente se ejecutará una vez.

La última parte de nuestro sketch se corresponde con la función **loop()**.

```
// The Main Loop Function
void loop() {

  Signal = analogRead(PulseSensorPurplePin);  // Read the PulseSensor's value.
                                              // Assign this value to the "Signal" variable.
   Serial.println(Signal);                    // Send the Signal value to Serial Plotter.


   if(Signal > Threshold){                          // If the signal is above "550", then
"turn-on" Arduino's on-Board LED.
     digitalWrite(LED13,HIGH);
   } else {
     digitalWrite(LED13,LOW);                //  Else, the sigal must be below "550", so
"turn-off" this LED.
   }

delay(10);



}
```

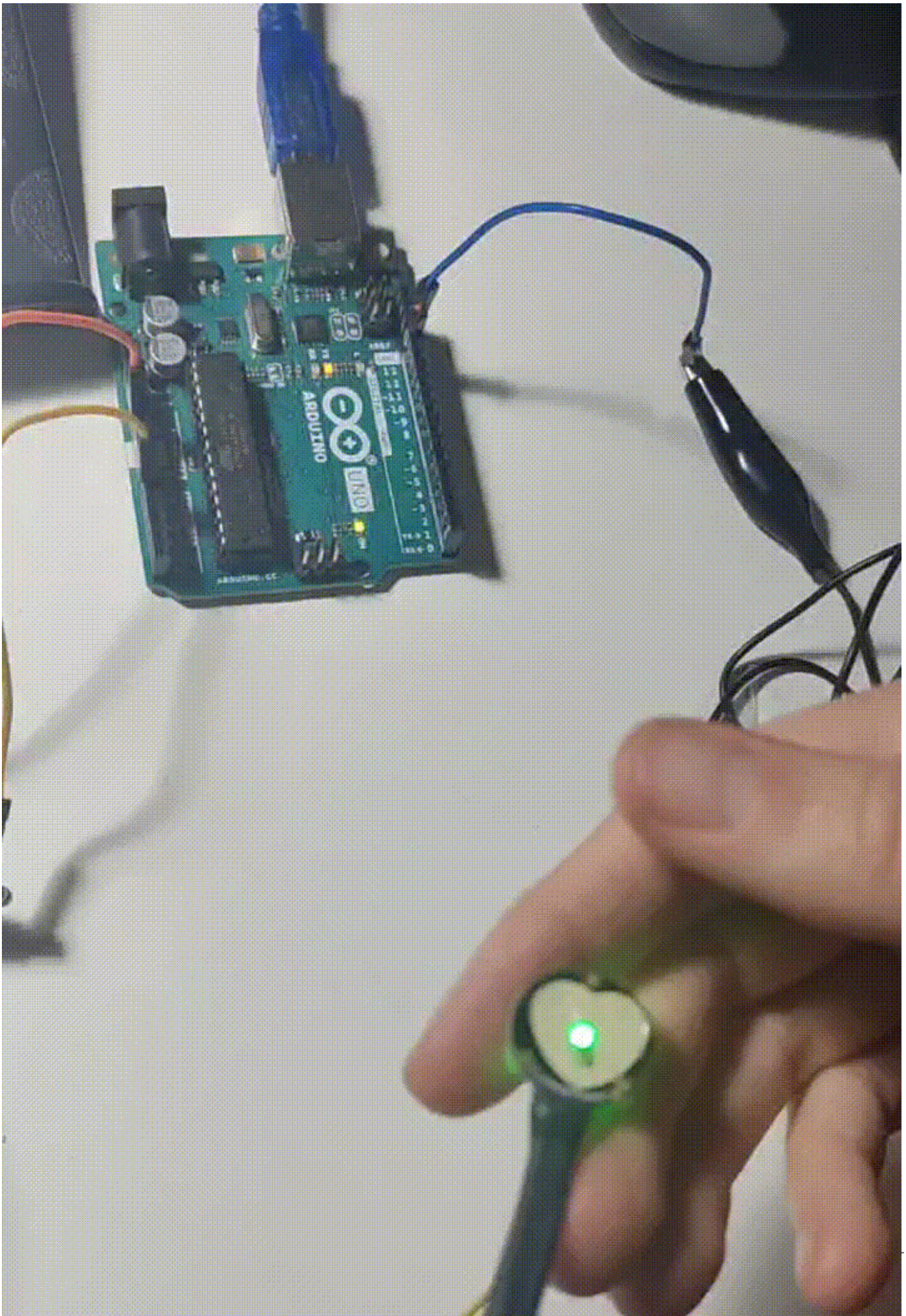
En ella, lo primero que vamos a hacer es almacenar en la variable Signal los valores que lea nuestro sensor. Para leer esos valores, usaremos la función **analogRead()**. Indicaremos que la señal analógica que ha de ser leída es la que recibamos con la variable **PulseSensor PurplePin**. A continuación, imprimiremos por nuestro puerto serial el valor obtenido, aunque para visualizar las pulsaciones no nos va a servir de mucho, porque los números aparecen a gran velocidad. Para visualizar las posiciones usaremos el LED. ¿Cómo? Pues usaremos un condicional: "Oye, Arduino, si **(if)** la señal que lees es mayor que el umbral (**Signal > Threshold**), enciende el LED **(digitalWrite(LED13,HIGH))**, si no **(else)**, apágalo (**digitalWrite(LED13,LOW)**)".

Para finalizar, usamos un **delay** de 10 milisegundos, para darle a nuestro Arduino un pequeño respiro entre lectura y lectura.

Ahora, **conectaremos nuestro UNO al ordenador** y subiremos el código.

En este GIF puedes ver cómo la luz naranja situada más a la derecha, comienza a parpadear cuando coloco mi dedo encima del pulsómetro:

Copyright 2025


ARDUINO
UNO