GUÍA DEENSAMBLAJE Y CODIFICACIÓN PARA LA COMUNICACIÓN DE UN MÓDULO DE ARDUINO SENSADO CON UN DISPOSITIVO DETECTOR DE PULSO CARDIACO

**Resultados de aprendizaje: Una vez desarrollada esta guía, el lector estará en capacidad de:**

* Obtener un valor que corresponde al ritmo cardíaco de una persona.
* Graficar el comportamiento de la onda de pulso cardiaco.
* Hacer uso de la herramienta SERIAL PLOTTER.
* Configurar los Timer como uno de los contadores de Arduino

**Herramienta de Software:**

* Arduino
* Bluetooth Terminal/Graphics

**Herramienta de Hardware:**

* Arduino Mega o Uno.
* Pulse Sensor.
* Protoboard
* LED(opcional)
* Zumbador
* Resistencia 1K ohm
* Modulo Bluetooth HC-05 / HC-06.
* Smartphone con OS Android.

**¿Qué es Arduino?**

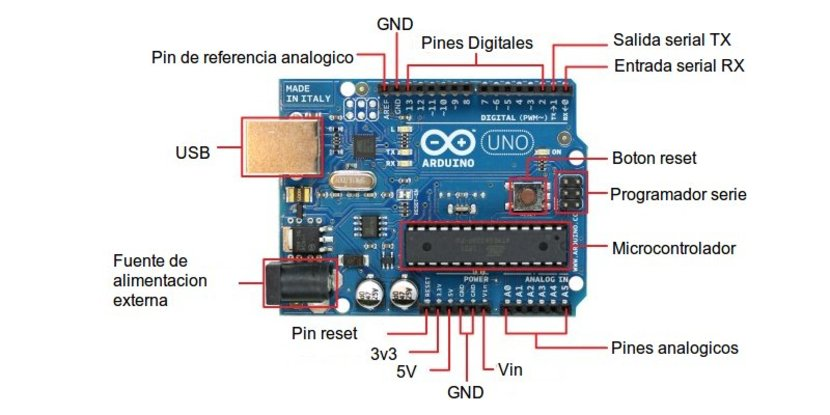
Arduino es una plataforma de prototipos electrónica de código abierto (open-source) basada en hardware y software flexibles y fáciles de usar. Está pensado para artistas, diseñadores, como hobby y para cualquiera interesado en crear objetos o entornos interactivos.

Arduino puede sentir el entorno mediante la recepción de entradas desde una variedad de sensores y puede afectar a su alrededor mediante el control de luces, motores y otros artefactos. El microcontrolador de la placa se programa usando el *Arduino Programming Language* (basado en Wiring) y el *Arduino Development Environment* (basado en Processing). Los proyectos de Arduino pueden ser autónomos o se pueden comunicar con software en ejecución en un ordenador (por ejemplo, con *Flash, Processing, MaxMSP*, etc.).

Las placas se pueden ensamblar a mano o encargarlas preensambladas; el software se puede descargar gratuitamente. Los diseños de referencia del hardware (archivos CAD)

están disponibles bajo licencia open-source, por lo que eres libre de adaptarlas a tus necesidades.

La tarjeta de desarrollo se presenta en la Figura 1:

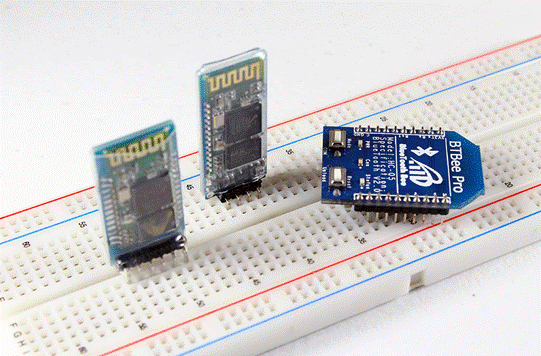


**Figura 1** Componentes de la Placa Arduino.

**¿Porque Usar Arduino?**

* **Barato:** Las placas Arduino son relativamente baratas comparadas con otras plataformas microcontroladoras. La versión menos cara del módulo Arduino puede ser ensamblada a mano, e incluso los módulos de Arduino preensamblados cuestan menos de 50$.
* **Multiplataforma:** El software de Arduino se ejecuta en sistemas operativos Windows, Macintosh OSX y GNU/Linux. La mayoría de los sistemas microcontroladores están limitados a Windows.
* **Entorno de programación simple y claro:** El entorno de programación de Arduino es fácil de usar para principiantes, pero suficientemente flexible para que usuarios avanzados puedan aprovecharlo también. Para profesores, está convenientemente basado en el entorno de programación Processing, de manera que estudiantes aprendiendo a programar en ese entorno estarán familiarizados con el aspecto y la imagen de Arduino.
* **Código abierto y software extensible:** El software Arduino está publicado como herramientas de código abierto, disponible para extensión por programadores experimentados. El lenguaje puede ser expandido mediante librerías C++, y la gente que quiera entender los detalles técnicos pueden hacer el salto desde Arduino a la programación en lenguaje AVR C en el cual está basado. De forma similar, puedes añadir código AVR-C directamente en tus programas Arduino si quieres.
* **Código abierto y hardware extensible:** El Arduino está basado en microcontroladores ATMEGA8 y ATMEGA168 de Atmel. Los planos para los módulos están publicados bajo licencia Creative Commons, por lo que diseñadores experimentados de circuitos pueden hacer su propia versión del módulo, extendiéndose y mejorándolo. Incluso usuarios relativamente inexpertos pueden construir la versión de la placa del módulo para entender cómo funciona y ahorrar dinero.

**¿Qué es Module Bluetooth?**

Es una tecnología creada por la compañía Ericsson para la comunicación inalámbrica entre dispositivos a corta distancia y en la banda libre 2.4GHz (ISM-Industrial, scientific and medical). Se utiliza en diversos productos tales como teléfonos celulares, laptops, vehículos, cámaras, impresoras entre otros (García A, 2011). Bluetooth se originó de la siguiente manera: Surgió como resultado de un proyecto de la compañía Ericsson en 1994. El nombre de la tecnología es en honor al rey danés Harald Blatand (“diente azul”). Las compañías Intel, IBM, Nokia, Toshiba se unieron a Ericsson para conformar el Bluetooth SIG (Special Interest Group) - Mayo de 1998. La versión 1.0 de la especificación del estándar se presentó en 1999. Opera sobre la banda ISM (Industrial Scientific Medical) de 2.45 Ghz, aunque hay restricciones en algunos países (Francia, España y Japón).

**Figura 2** Módulos Bluetooth.

Existen en el mercado varios tipos de módulos bluetooth para arduino y otras placas, como lo podemos apreciar en la imagen **anterior,** Como lo es el BT o bluetooth de bajo consumo (1.8 v), y los más comerciales HC-06 y HC-05.

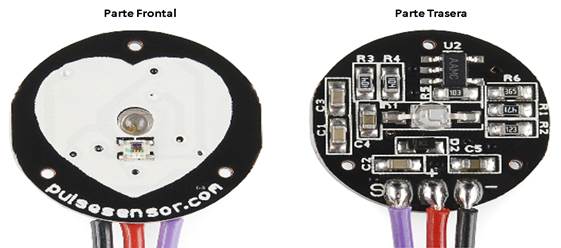
**EL HC-06** tiene 4 pines: del **HC-05** solo se diferencia físicamente porque tiene soldado sus pines **EN** y **STATE**.

* **Vcc**: Voltaje positivo que funciona a 3.6 v - 6 v.
* **GND:** Polo a tierra o voltaje cero, se tienen que conectar al GND del Arduino.
* **TX**: Pin de Transmisión de datos, por este pin el HC-06 transmite los datos que le llegan desde la PC o Móvil mediante bluetooth, este pin debe ir conectado al pin RX del Arduino.
* **RX**: pin de Recepción, a través de este pin el HC-06 recibirá los datos del Arduino los cuales se transmitirán por Bluetooth, este pin va conectado al Pin TX del Arduino.

El **HC-06** se diferencia del 05 ya que el 06 funciona solo como transmisión esclava (solo puede recibir señal) y el 05 como esclava y master (master puede recibir y transmitir señal).

Para el emparejamiento con cualquier dispositivo las contraseñas por defecto para estos módulos son los **Password**: 1234 o 0000.

**¿Qué es Pulse Sensor?**



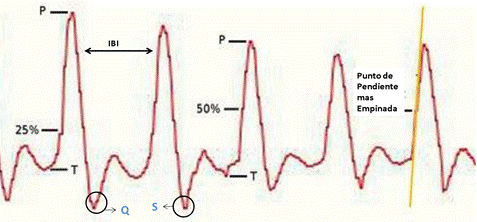
**Figura 3** Pulse Sensor.

El Pulse Sensor es un sensor de ritmo cardiaco para Arduino. Este sensor es usado habitualmente, para introducir de algún modo la frecuencia cardiaca en un proyecto. Como se puede observar en la Figura 2 en la parte frontal del sensor, hay un agujero por donde sale la luz emitida desde un LED que se encuentra en la parte posterior. Justo debajo del agujero se encuentra un sensor de luz ambiente, como el que se usa en los smartphones para atenuar o subir automáticamente el brillo de la pantalla dependiendo de la luminosidad del espacio donde te encuentras. Esta parte frontal es la que está en contacto con la piel y preferiblemente en alguna parte translúcida del cuerpo ya sea la punta del dedo o el lóbulo de la oreja.

**Funcionamiento**

Cuando el corazón bombea sangre a través del cuerpo, con cada latido se genera una onda de pulso que viaja por todas las arterias y venas del tejido capilar, que es donde se une con el Pulse Sensor. Este sensor responde a los cambios relativos en la intensidad de luz. Estos cambios de luz se generan porque en el paso de esta onda hay más cantidad de sangre que bloqueará la luz emitida por el LED y que es la percibida por el fotosensor. Si la cantidad de luz que incide sobre el sensor se mantiene constante, el valor de la señal permanecerá cerca de 512 que es la mitad del rango de medición. Si recibe más luz la señal sube y si recibe menos la señal baja.

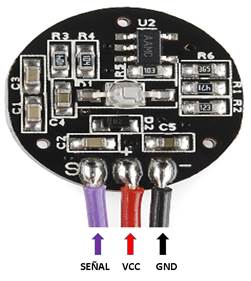
El pulse Sensor es esencialmente un fotopletismógrafo, que es un dispositivo médico conocido utilizado para el monitoreo de la frecuencia cardiaca no invasiva. La señal del pulso del corazón que sale de un fotopletismógrafo, no es la señal eléctrica del corazón como se tendría en un electrocardiograma, sino que es una fluctuación en el voltaje analógico, y tiene una forma de onda predecible como se muestra en la Figura 3. La representación de la onda de pulso se llama fotopletismografía o PPG.



**Figura 4** Fotopletismografía.

El objetivo de este sensor es encontrar los momentos sucesivos de cada latido y calcular el intervalo de tiempo que ha habido entre ellos llamado IBI (Inter-Beat Intervale), haciendo uso de la forma predecible y del patrón de onda del PPG.

**Conexiones:**

Las conexiones de este sensor se muestran en la Figura 5.

**Figura 5** conexiones.

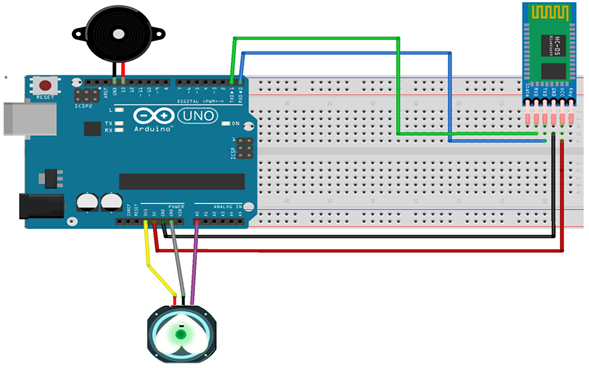
* **VCC** (rojo): Cable de alimentación del sensor, se puede alimentar desde 3V hasta 5V por lo tanto puede ir conectado al pin de 3.3V como al de 5V.
* **GND** (negro): Cable ground, irá conectado al pin GND de la placa Arduino.
* **SEÑAL** (morado): Cable por donde se transfieren los datos recogidos por el fotosensor. Es una señal analógica, por lo tanto, irá a una de las 6 entradas analógicas de la placa.

* **Consideraciones:**
* **Q** Es el punto inicial de la onda de pulso.
* **S** Es el punto final de la onda de pulso.
* **T** Es el punto más bajo de la onda.
* **P** Es el punto más Alto de la onda.

Para calcular el número de latidos por minuto a través de la señal del ECG basta con saber el intervalo de tiempo transcurrido entre latido y latido y multiplicarlo por 60.

**Procedimiento:**

* Realizar el montaje del circuito como se muestra en la Figura 6 y conectar la placa al computador.

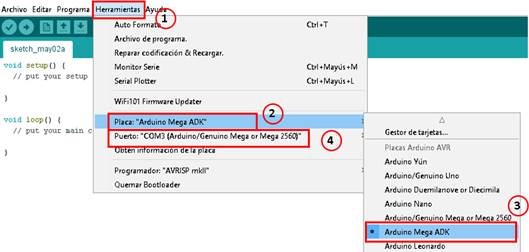


**Figura 6** Montaje de los componentes .

* Una vez el montaje este hecho abrirás la IDE de arduino, conectaras la placa al puerto USB, en la pestaña Herramientas seleccionas la placa con la que vamos a trabajar y verificar que el puerto COM ya se encuentre disponible, como se muestra en la Figura 7.

Recomendación: Si quieres disminuir la alteración de la señal (ruidos), podrías aplicar silicona en la parte de los circuitos, con el cuidados de no tapar mucho el led. (la silicona es la más óptima por su “transparencia”)

**Configuración Inicial para la IDE de Arduino (el arduino debe estar conectado):**



**Figura 7** Selección de Placa.

**Pasos**

1. Dirigirse a la pestaña de herramientas.
2. Seleccionar la opción de Placa.
3. Buscar la placa a usar (Arduino Uno o Arduino Mega).
4. Posteriormente verificar y seleccionar un puerto USB COM disponible.

**Declaración de variables:**

Variables globales del código (se pueden usar en cualquier parte sin ser declaradas)

En esta primera parte se definen las variables a usar dentro del código cada una con su respectivo comentario acerca de qué es lo que debe hacer. Es importante comentar que estas son variables globales (se pueden usar en cualquier parte del código).

En este punto de la definición del código es importante saber a qué tipo de dato es al que se refiere cada variable, por ejemplo las más básicas a explicar son INT y BOOLEAN, INT es un dato numérico entero otro dato numérico es LONG, y BOOLEAN es una dato lógico “verdadero” o “falso” (true;false).Por otro lado, hay unas variable cuya combinación es acompañada de un tipo de declaración diferente como lo es el.

* **Variable:** Una variable es definida como el lugar de almacenamiento de un dato, cuyo valor puede cambiar en distintas ocasiones, se distingue diferentes tipos de variables que se pueden clasificar por tipo; por ejemplo las más básica a explicar son INT y BOOLEAN. Int es un dato numérico entero otro dato numérico es Long, y Boolean es un dato lógico “verdadero” o “falso” (true; false).
* **Volatile:** Es declarado cuando el valor que debe resguardar puede ser cambiado por hilos de procesos que no necesariamente siguen la línea del código, o cuando existen algunos procesos simultáneos, tiende a ser usados para almacenar lo valores de los interrupts o interrupciones en el Microprocesador del Arduino. Son llamadas técnicamente como rutinas de servicio de interrupción.
* **Unsigned**: Son los mismos que enteros (INT) en qué almacenan un valor de 2 bytes. En lugar de almacenar números negativos, sin embargo, sólo almacenan valores positivos, dando un rango útil de 0 a 65.535.
* **RX:** Recepción de datos.
* **TX:** Transmisión de datos.
* El Rx y TX del módulo bluetooth hacia la tarjeta arduino deben estar invertidos TX con RX y el RX con TX.

**Definición del Loop y del Setup:**

**El loop** es una parte cual se encarga de poner en marcha el código en la totalidad de su funcionamiento, este proceso es repetitivo y se ejecuta de manera cíclica.

**El setup** es el lugar del código donde se instancian preferiblemente los recurso que se usarán inicialmente para el Arduino.

**El timer** es un contador muy similar al **delay()** pero este puede ser ejecutado ejerciendo diferentes funciones sin hacer pausa de todo el Arduino, Esto quiere decir que puede ser ejecutado como un proceso multihilos. (el **delay()** es una retardo de ejecución contado en milisegundos ) .

**El interruptSetup** al igual que el setup convencional o por default de el Arduino este es usado para establecer las configuraciones previas del Timer tales como los ciclos, y la activación de las Pausas.

**Sobre el Subproceso ISR:**

Desactiva la interrupción  **cli();**  Activa la interrupción **sei();**

Exactamente en el interrupSetup se usa un sei() para darle inicio a las interrupciones y con el cli() se detiene en el momento en el que se ingresa a el TIMER\_2. (Al Final del ISR se inicia nuevamente las interrupciones con Sei() ).

**IBI:** Amplitud de pulso a pulso.

**Código:**

Con el montaje de la Figura 6 realizado, y la configuración del IDE de arduino listo copiar el siguiente código, y lo pegarlo en el editor de Arduino para cargarlo a la placa.

**//VARIABLES:**

**volatile int Transcurrido\_Array[10]; // Arreglo para registrar el tiempo transcurrido entre cada pulso(IBI).**

**volatile unsigned long Respaldo\_Conteo = 0; // Contador de respaldo**

**volatile unsigned long Momento\_UltimoPulso = 0; // Tiempo transcurrido hasta el ultimo pulso.**

**volatile int P =512; // Punto mas alto de una onda de pulso.**

**volatile int T = 512; // Punto inicial del pico de onda.**

**volatile int ValorMedioREF = 530; // Punto medio del pico de onda.**

**volatile int AnchoOndaValido = 0; // Ancho del pico de onda.**

**volatile boolean Primer\_Pulso = true; // Primera porcion de pulso para el calculo del BPM (arreglo).**

**volatile boolean Segundo\_Pulso = false; // Segunda porcion de pulso para el registro del BPM (arreglo).**

**int Pin\_Entrada\_sensor = 0; // Pin A0, donde se recibe la señal del sensor.**

**int Pin\_Desvanecimiento\_LED\_BUZZER = 13; // Pin 13, donde emite un sonido o luz en cada pulso.**

**// Volatile Variables, es un tipo variable que cuando ya es usada se desecha el valor que contine y toma uno nuevo.**

**// Mediante volatile se actualizan los datos tomados**

**volatile boolean Q\_S = false;**

**volatile int BPM; // Pulsos por Minuto se actualiza constantemente.**

**volatile int signal\_sensor; // Almacena todos los datos enviados por el sensor.**

**volatile int IBI = 600; // Almacena el tiempo entre cada de pulso (Valor 600 de referencia).**

**volatile boolean Pulso = false; // Para la deteccion de pulso valido.**

**void setup() { //setup() es la parte encargada de recoger la configuración.**

**pinMode(Pin\_Desvanecimiento\_LED\_BUZZER,OUTPUT); // Declaramos el LED o Buzzer que se acciona por cada pulso.**

**Serial.begin(9600); // Velocidad muestreo o toma de datos del arduino**

**interruptSetup(); // Metodo que ejecuta la configuracion inicial del interrupt.**

**}**

**void interruptSetup(){ //CONFIGURACION**

**TCCR2A = 2; // Desactiva la modulacion de onda en los pines 3 y 11, y va al CTC modo**

**TCCR2B = 6; // no forzar la comparacio preescalar a 256**

**OCR2A = 124; // Cantidad de ciclos 124 a 500Mhz**

**TIMSK2 = 2; // ACTIVA LAS PAUSA DEL MARCADOR ENTRE EL (timer)**

**sei(); // DEFINE GLOBALMENTE LOS PARAMETROS ESTALECIDOS ARRIBA INICIANDOSE**

**}**

**void loop() { // Metodo principal que se ejecuta varias veces.**

**// NOTA! ESTE SE EJECUTA SIMULTANEAMENTE CON EL INTERRUPT.**

**serialOutput();// Metodo de salida (muestra los valores finales).**

**delay(1.67); } // Tiempo de espera en mili-segundos.**

**void serialOutput(){**

**Serial.print("E"); // "E" indica el inicio de valores en la app.**

**Serial.print(BPM); // Valor de ulsos por minuto.**

**Serial.print(","); // "," Separacion de un dato en la app y monitor Serie.**

**Serial.print(IBI); // Tiempo transcurrido entre pulsos(ms).**

**Serial.print(","); // "," Separacion de un dato en la app y monitor Serie.**

**Serial.println(signal\_sensor);**

**}**

**// Rutina de servicio de interrupcion:ISR.**

**ISR(TIMER2\_COMPA\_vect){**

**cli(); // Detiene el servicio de interrupcion de rutina (ISR).**

**signal\_sensor= analogRead(Pin\_Entrada\_sensor); // Lectura del valor arrojado por el sensor.**

**Respaldo\_Conteo += 2; // Lleva un conteo auxiliar acumulado a 2 por ciclo.**

**int N = Respaldo\_Conteo - Momento\_UltimoPulso; // "N" registra el tiempo despues de el ultimo pulso.**

**// encuetra un pico optimo donte empezar los calculos del pulso cardiaco.**

**if(signal\_sensor< ValorMedioREF && N > (IBI/5)\*3){ // Evita los falsos latidos o latidos no validos.**

**if (signal\_sensor< T){ // SI la lectura del pulso es mayor a 512.**

**T = signal\_sensor; // Pulso de pico mas bajo.**

**}**

**}**

**if(signal\_sensor> ValorMedioREF && signal\_sensor> P){ // Condicion para captar picos superiores (por encima de 530).**

**P = signal\_sensor; // P es el punto mas alto.**

**}**

**if (N > 250){ // Condicion cuando han transcurrido mas de 250 ms.**

**if ( (signal\_sensor> ValorMedioREF) && (Pulso == false) && (N > (IBI/5)\*3) ){**

**Pulso = true; // Se activa para posibles pulsos.**

**digitalWrite(Pin\_Desvanecimiento\_LED\_BUZZER,HIGH); // Activa el pin 13.**

**IBI = Respaldo\_Conteo - Momento\_UltimoPulso; // Medida entre pico a pico de un pulso en ms.**

**Momento\_UltimoPulso = Respaldo\_Conteo; //Guarda el ultimo pulso en otra variale.**

**if(Segundo\_Pulso){ // Evalua si se registro un pulso antes (si ya paso por el primero)**

**Segundo\_Pulso = false; // Pone el valor en falso para evaluar nuevamente la condicion.**

**for(int i=0; i<=9; i++){ // Toma todos los 10 valores validos de IBI.**

**// los valores de IBI guardados son referencia para el BPM.**

**// IBI esta medido en ms y los almacena en una variable.**

**Transcurrido\_Array[i] = IBI; // Guarda el valor de los IBI obtenidos.**

**}**

**}**

**if(Primer\_Pulso){ // Evalua el primer pulso y prepara el segundo pulso.**

**Primer\_Pulso = false; // Cambiamos a false para que evalue nuevamente la condicion.**

**Segundo\_Pulso = true; // Da aviso que el pulso siguiente es el segundo.**

**sei(); // Reanuda la interrupciones.**

**}**

**word runningTotal = 0; // Guarda la sumatoria de todos los valores del IBI ms.**

**for(int i=0; i<=8; i++){ // Ciclo para filtrar los datos de "Transcurrido\_Array".**

**Transcurrido\_Array[i] = Transcurrido\_Array[i+1]; // Organizamos el arreglo para dejar solo los valores actuales.**

**runningTotal += Transcurrido\_Array[i]; // Suma de IBI contenidos en las primeras 8 posiciones del arreglo.**

**}**

**Transcurrido\_Array[9] = IBI; // Ingresa el ultimo valor de IBI.**

**runningTotal += Transcurrido\_Array[9]; // Agrega el ultimo IBI a runningTotal.**

**runningTotal /= 10; // Promedio de los ultimos 10 IBI.**

**BPM = 60000/runningTotal; // Cantidad de pulsaciones por minuto.**

**}**

**}**

**if ((signal\_sensor < ValorMedioREF) && (Pulso == true)){ // Cuando capture el final de un pulso CARDIACO valido.**

**digitalWrite(Pin\_Desvanecimiento\_LED\_BUZZER,LOW); // Apaga el pin 13 LED (PRENDE Y APAGA SEGUN EL PULSO).**

**Pulso = false; // Pone el pulso en "false" por que ya esta en el punto final del pulso (S)**

**AnchoOndaValido = P - T; // Mide la distancia que existe eNtre el puNto del pulso mas alto y el puNto mas bajo de la lectura de pulso**

**ValorMedioREF = AnchoOndaValido/2 + T; // una pendiente cuya longitud determina la altura de la onda**

**P = ValorMedioREF; // Actualiza valores.**

**T = ValorMedioREF;**

**}**

**if (N > 2500){ // Si han pasado 2.5 segundos y no hay un pulso, tome valores iniciales**

**ValorMedioREF = 530; // ValorMedioREF POR DEFECTO EL MISMO VALOR De INICIO**

**P = 512; // VARIABLE P Y T a valor 512 como al inicio**

**T = 512;**

**Momento\_UltimoPulso = Respaldo\_Conteo; // Llama el ultimo pulso que exista del contador de muestra**

**Primer\_Pulso = true; // El primer pulso toma el valor inicial por defecto**

**Segundo\_Pulso = false; // El segundo pulso toma el valor inicial por defecto**

**}**

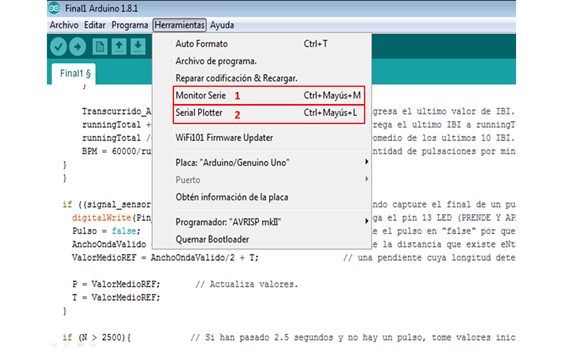
**// FIN DEL METODO DE CUANTIFICACION DE DATOS VALIDOS**

**sei(); } // Inicia de nuevamente las Interrupciones.**



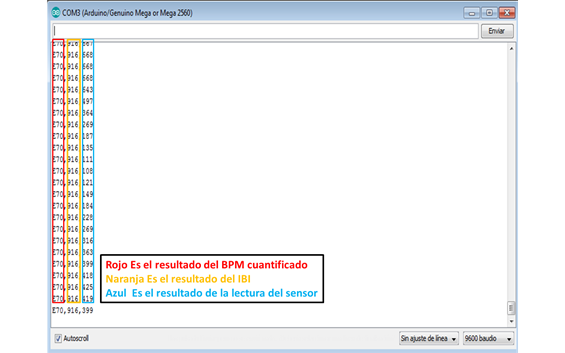
**Figura 8** Verificar y Subir.

1. Verifica que el código no contenga errores.
2. Sube el código a la placa Arduino.

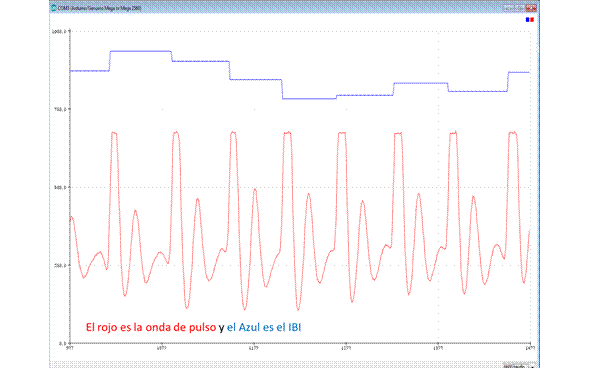


**Figura 9** Herramientas de observación de la señal.

1. El monitor serie sirve para ver los datos numéricos del sensor cardiaco.
2. El Serial Plotter sirve para ver la graficación del ritmo cardiaco.



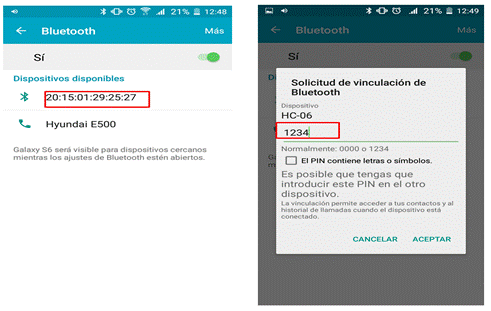
**Figura 10** Monitor Serie.



**Figura 11** Serial Plotter.

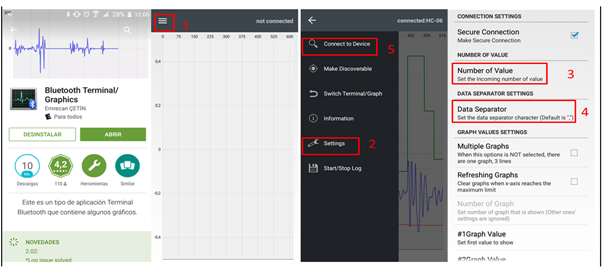
Una vez el dispositivo esté funcionando puedes calcular el tu pulso para verificar que está funcionando óptimamente, Cuentas la cantidad de pulsos en 15 segundos y multiplicas por 4. (Puedes tomar tu pulso en la oreja óptimamente ya que posee menos tejidos. )

**Conexión entre Android y Arduino:**



**Figura 12** Emparejamiento del módulo bluetooth.

Para esto lo primero es emparejar el módulo bluetooth con el smartphone, luego dirigirse a la configuración de dispositivo móvil y seleccionar la opción de bluetooth donde se debe buscar el módulo de bluetooth y vincularlo al dispositivo como en la Figura 12.



**Figura 13** Configuración de la App.

La aplicación a utilizar Bluetooth Graphics, la cual es una aplicación gratuita que permite la comunicación entre un módulo de Bluetooth y un dispositivo Android, permitiendo así la graficación de las señales que obtenemos del sensor de pulso cardiaco por medio de Arduino.

Esta app se debe configurar de la siguiente manera.

1. Abrir el menú de la app.
2. Seleccionar la opción “Settings”.
3. En “Number of Value” seleccionar la opción “3 values”.
4. En “Data Separator” introducir una coma “,”.
5. Regresar al menú, seleccionar “Connect to Device” y escoger el módulo a conectar.

**Referencias:**

* ¿QUÉ ES ARDUINO?. arduino. [Consultado el 9 de Mayo de 2017]. Disponible en internet: <http://arduino.cl/que-es-arduino/>

# **Pulse Sensor Amped. pulsesensor. 2011.** [Consultado el 9 de Mayo de 2017]. Disponible en internet:https:v //pulsesensor.com/pages/pulse-sensor-amped-arduino-v1dot1

* Imagen pulse sensor. [Consultado el 9 de Mayo de 2017].Figura 3 Pulse sensor. Disponible en internet: http://www.theorycircuit.com/pulse-sensor-arduino/
* Imagen arduino . [Consultado el 9 de Mayo de 2017]. **Figura 1** Componentes de la PlacaArduino.Disponible en internet:<http://files.tecnologiasces.webnode.es/200000067-c6eedc887c/arduino-partes1.jpg>
* Bluetooth HC-05 y HC-06 **Figura 2** Módulos Bluetooth.[Consultado el 9 de Mayo de 2017]. Disponible en internet:<http://www.geekfactory.mx/tutoriales/bluetooth-hc-05-y-hc-06-tutorial-de-configuracion/>

**Enlaces de Interés:**

* <https://pulsesensor.com/>
* <http://makezine.com/projects/ir-pulse-sensor/>
* <https://play.google.com/store/apps/details?id=com.emrctn.BluetoothGraphics&hl=es>

Puede consultar videos del funcionamiento en los siguientes enlaces:

* <https://www.youtube.com/watch?v=O7GUB821SDo>
* https://youtu.be/gw\_8C-xizT8

**Video del código**