Trabajo Fin de Grado
Grado en Ingeniería Electrónica, Robótica y Mecatrónica (UMA-US)

Desarrollo de aplicación software para equipo de monitorización médica basado en Arduino

Autor: Ana Isabel Gilibert Valdés

Tutor: Ascensión Zafra Cabeza

Dpto. Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2019

Departamento de Teoría de la Señal y Comunicaciones

Trabajo Fin de Grado
Grado en Ingeniería Electrónica, Robótica y Mecatrónica (UMA-US)

Desarrollo de aplicación software para equipo de monitorización médica basado en Arduino

Autor:
Ana Isabel Gilibert Valdés

Tutor:
Ascensión Zafra Cabeza
Profesor Titular

Dpto. Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2019

Trabajo Fin de Grado: Desarrollo de aplicación software para equipo de monitorización médica basado en Arduino

Autor: Ana Isabel Gilibert Valdés
Tutor: Ascensión Zafra Cabeza

El tribunal nombrado para juzgar el trabajo arriba indicado, compuesto por los siguientes profesores:

Presidente:

Vocal/es:

Secretario:

acuerdan otorgarle la calificación de:

El Secretario del Tribunal

Fecha:

Agradecimientos

Escribo estas palabras aún sin haber terminado la carrera, pues aún me queda prácticamente un año por delante. El motivo es que a veces la vida no es tan fácil como creemos, y de un segundo para otro te puede cambiar la vida completamente.

La etapa que voy a comenzar ahora, el comienzo del último año de carrera ya la viví el año pasado, pero a mí una "simple" llamada me cambió la vida. Sin embargo, eso fue un motivo más para aprender a disfrutar de las pequeñas cosas de la vida, de los pequeños momentos que día tras día te hacen crecer como persona, y sobre todo fue un motivo más para aprender a ser feliz.

Así que en primer lugar me gustaría dar gracias a la vida, que aunque a veces no sea tan justa me ha demostrado que después de lo malo siempre viene algo bueno. Además, me ha regalado unos padres maravillosos que han luchado toda su vida por el sueño de sus hijos: mi hermano y yo. Gracias a mi madre, mi ejemplo de superación y fortaleza, que a pesar de todo hace lo posible por vernos siempre felices. A mi padre, gracias por todos esos años que nos diste, por tu bondad, y por brillar siempre con luz propia, aunque ahora lo hagas en forma de estrella. Y a mi hermano, por ayudarme siempre en todo lo posible.

Gracias a mis amigos de la carrera, en especial a Paula, Rimsen y Carri, por acompañarme en tantos momentos buenos, pero también en momentos de agobio y tristeza.

Gracias a mi "pequeña" amiga Paula, la que siempre está para todo y nunca falla. La que me apoya en todo lo que hago y la que sabe todo de mí, y estando juntas nada ni nadie puede con nosotras.

Gracias a Javi, por aparecer en mi vida de forma inesperada y por hacer que vuelva a creer en el amor.

Y por último, pero no menos importante, gracias a aquellos profesores con los que de verdad he aprendido en la carrera. Y a Asun, mi tutora, por prestarse a llevar mi TFG a distancia dejando que me llevara todos los dispositivos a casa, y por ser tan atenta y responder tan rápido a todos los correos.

De verdad, gracias, porque gracias a vosotros hoy por hoy soy quien quiero ser.

Resumen

Este proyecto consiste en la programación de una interfaz gráfica para el usuario con el fin de facilitar el uso de aparatos médicos en un entorno doméstico. En concreto, se centra en el tensiómetro, espirómetro y electrocardiograma. Ello permitirá conocer un diagnóstico previo de las pruebas realizadas antes de acudir al médico, así como ver un progreso temporal que quedará guardado en el archivo de usuario.

Para ello, se ha trabajado en un entorno de programación con Arduino y Matlab basado en un protocolo de comunicación por puerto serie. Es decir, Arduino se encargará de controlar los distintos sensores en función de lo que reciba por parte de Matlab y mandará el resultado de la prueba al mismo, donde se ha programado la interfaz gráfica de usuario.

Se explicará en un tutorial el procedimiento para poner en marcha cada sensor, así como el funcionamiento y la utilización de la interfaz gráfica de cada uno.

Abstract

This project consists of programming a graphical user interface to facilitate the use of medical devices in a domestic environment. Specifically, it focuses on the blood pressure monitor, spirometer and electrocardiogram. This will allow to know a previous diagnosis of the tests carried out before going to the doctor, as well as to see a temporal progress that will be saved in the user file.

To this end, work has been done in a programming environment with Arduino and Matlab based on a serial communication protocol. In other words, Arduino will control the different sensors according to what it receives from Matlab and will send the test result to Matlab, where the graphical user interface has been programmed.

A tutorial will explain the procedure for starting up each sensor, as well as the use of the graphical user interface of each one.

Índice

Resumen III
Abstract V

1 Introducción 1
1.1 Bioingeniería en la Historia 1
1.2 Motivación y Objetivos 2

2 Hardware 3
2.1 Arduino Uno 3
2.2 MySignals HW 4
2.3 Ordenador utilizado 5

3 Tutorial de instalación y puesta en marcha 7
3.1 Instalación de Arduino y librerías de Mysignals 7
3.2 Puesta en marcha 10

4 Tensiómetro 19
4.1 Qué es 19
4.2 Funcionamiento 20
4.3 Valores normales 20
4.4 Interfaz gráfica 21
4.5 Diagnóstico 25

5 Espirómetro 29
5.1 Qué es 29
5.2 Funcionamiento 30
5.3 Valores normales 30
5.4 Interfaz gráfica 31
5.5 Diagnóstico 35

6 Electrocardiógrafo 41
6.1 Qué es 41
6.2 Funcionamiento 42
6.2.1 El sistema de conducción 42
6.3 ECG normal 43
6.4 Interfaz gráfica 44
6.5 Diagnóstico 48

7 Impresión 3D 51
7.1 Diseño 51
7.2 Vista 3D de la caja protectora 53

8 Conclusiones líneas futuras **55**
8.1 Conclusiones 55
8.2 Líneas futuras 55

Apéndice A Código Arduino **57**

Apéndice B Código Matlab - GUI tensiómetro **67**

Apéndice C Código Matlab - GUI espirómetro **75**

Apéndice D Código Matlab - GUI ECG **85**

Apéndice E Código Matlab - GUI INICIO **93**

Apéndice F Código Matlab - Funciones **97**

Índice de Figuras 111
Índice de Códigos 113
Bibliografía 115

1 Introducción

La bioingeniería se ha convertido en una gran protagonista de nuestro día a día, ya que sin la tecnología sería prácticamente imposible avanzar en los distintos campos de la ciencia, y, respecto al tema que nos concierne, juega un papel imprescindible en el campo de la medicina. Sólo basta con fijarse en todos los aparatos que hay en los hospitales, ambulatorios, laboratorios, e incluso en aquellos pequeños aparatos que utilizamos en casa para comprobar si estamos enfermos.

Esta inclusión de la tecnología en la medicina brinda una mayor atención al usuario, pues, además de facilitar el trabajo a los profesionales de salud realizando las distintas pruebas de una manera más rápida y precisa, nos ha permitido incluir numerosos tratamientos que sin la tecnología no sería posible.

1.1 Bioingeniería en la Historia

El desarrollo de la bioingeniería comenzó cuando los ingenieros encontraron aplicación de la física en el diagnóstico médico.

Así, la década de los cuarenta se conoce como el periodo formativo de la bioingeniería, ya que se desarrollaron numerosos instrumentos y conceptos. Al principio de esta década, el ingeniero Donald Sproule utilizó el ultrasonido por primera vez para visualizar imágenes de tejidos blandos; y al final de esta década, el médico holandés Willem Kolff diseñó el primer equipo para diálisis, estableciendo las bases para el tratamiento de la falla renal crónica.

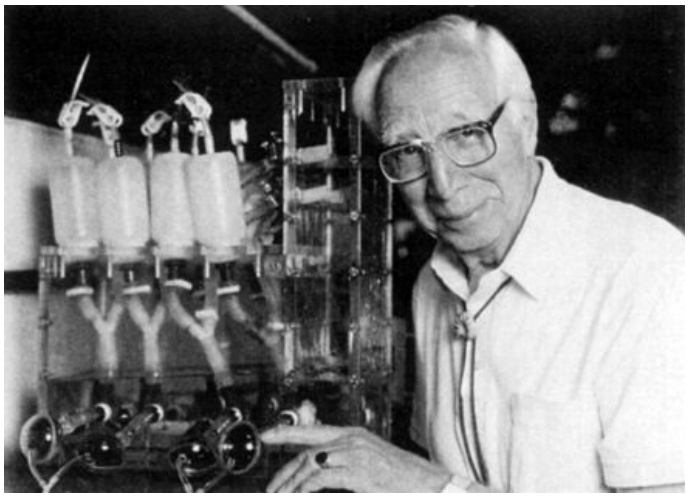

Figura 1.1 Willem Kolff, diseñador del primer equipo para diálisis.

En la década de los cincuenta, la bioingeniería jugó un papel muy importante al desarrollar dispositivos como el marcapasos, siendo en 1958 cuando se instaló el primer marcapasos interno con una fuente de energía incluída, lo cual fue posible gracias a la aparición del transistor.

A partir de entonces, la bioingeniería forma parte del día a día, ya que sin tecnología resulta imposible avanzar en cualquier investigación.

1.2 Motivación y Objetivos

Debido a la importancia que tiene la tecnología en el ámbito de la salud, sería interesante aportar un granito de arena a la bioingeniería que hay actualmente, ya que la salud es un problema de todos, y la salud se cuida mejor de la mano de la ingeniería.

Por ello, se va a desarrollar una interfaz gráfica con el fin de facilitar el uso de aparatos médicos, así como poder guardar el resultado para ver un progreso a lo largo del tiempo. De esta manera el usuario podría tener un diagnóstico previo al profesional de la salud, que por supuesto será el que tenga la última palabra para un diagnóstico completo y fiable.

A lo largo de este proyecto se explicará lo que se ha utilizado para la programación de la interfaz, y además se explicará el funcionamiento de cada dispositivo con el fin de facilitar su uso.

2 Hardware

En este capítulo se detallarán los elementos de hardware utilizados para el desarrollo del proyecto.

2.1 Arduino Uno

Arduino es una plataforma de código abierto enfocada en facilitar el uso de la electrónica.
En concreto, se va a utilizar Arduino Uno, representada en la Figura 2.1 que es la primera placa electrónica de Arduino basada en USB, y además es la más utilizada por su bajo coste y versatilidad para iniciarse en el mundo de la programación electrónica.

Figura 2.1 Arduino Uno.

Destacando las siguientes especificaciones técnicas:

- **Microcontrolador:** Microchip ATmega328P
- **Voltaje de funcionamiento:** 5 V
- **Voltaje de entrada:** 7-20 V
- **Pines de E/S digitales:** 14 (6 con salida PWM)
- **Pines de entrada analógica:** 6
- **Corriente DC por Pin de E/S:** 20 mA
- **Corriente CC para Pin de 3.3V:** 50 mA
- **Memoria Flash:** 32 KB (0.5 KB utilizados por el gestor de arranque)
- **SRAM:** 2 KB
- **EEPROM:** 1 KB
- **Velocidad del reloj:** 16 MHz
- **Longitud:** 68.6mm
- **Ancho:** 53,4mm
- **Peso:** 25g

Éste se utilizará para conectar el escudo de Mysignals, que se detallará a continuación.

2.2 MySignals HW

Mysignals es un escudo que se conecta a Arduino Uno, que permitirá conectar 11 sensores diferentes, como se puede ver en la Figura 2.2.

Dicho escudo es comercializado por © Libelium Comunicaciones Distribuidas S.L. y se puede comprar en: *http://www.my-signals.com/*

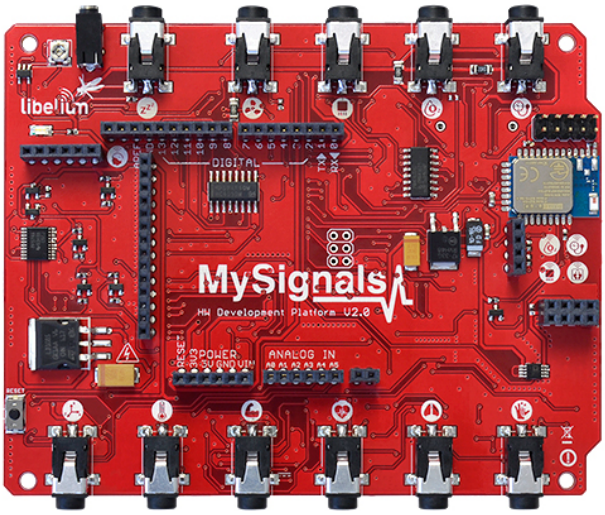

Figura 2.2 MySignals HW.

Los sensores disponibles son los siguientes:

- Pulsioxímetro
- Sensor de Electrocardiograma
- Sensor de flujo de aire (respiración)
- Tensiómetro
- Glucómetro
- Sensor de temperatura corporal
- Sensor de electromiografía
- Espirómetro
- Sensor de respuesta galvánica de la piel
- Sensor de posición corporal
- Sensor de ronquido

En este proyecto se utilizarán, por la importancia de sus datos desde el punto de vista médico, los siguientes sensores:

- **Tensiómetro**
- **Sensor de Electrocardiograma**
- **Espirómetro**

2.3 Ordenador utilizado

Además de lo mencionado, se ha utilizado para la conexión del Arduino y el trabajo de programación un ordenador con las siguientes características:

- Procesador Intel®Core™ i7-7700HQ 2.81 GHz
- 8 GB de RAM DDR4
- Disco duro de 1 TB
- SSD de 500 GB
- Tarjeta gráfica NVIDIA® GeForce® GTX 1050
- x64 bits

3 Tutorial de instalación y puesta en marcha

En este capítulo se detallará el procedimiento para la instalación de Arduino y de las librerías de Mysignals, que serán necesarias para la puesta en marcha de los sensores.

3.1 Instalación de Arduino y librerías de Mysignals

1. Si no está instalado, se debe descargar Arduino IDE de la página oficial e instalarlo: *https://www.arduino.cc/en/main/software*
2. Se procede a descargar las librerías de MySignals del siguiente enlace y se descomprime: *http://www.cooking-hacks.com/media/cooking/images/documentation/mysignals_hardware/MySignals_HW_SDK_V2.0.2.zip*
3. Hay que incluir las librerías en Arduino IDE: (Programa > Incluir Librería > Añadir biblioteca ZIP…)

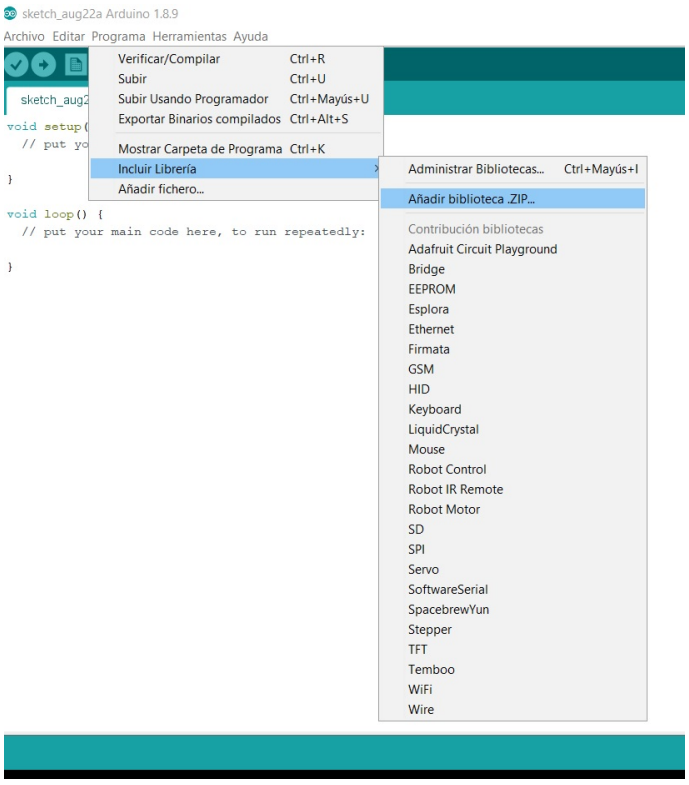

Figura 3.1 Incluir librerías en Arduino.

Se abre la carpeta descomprimida:

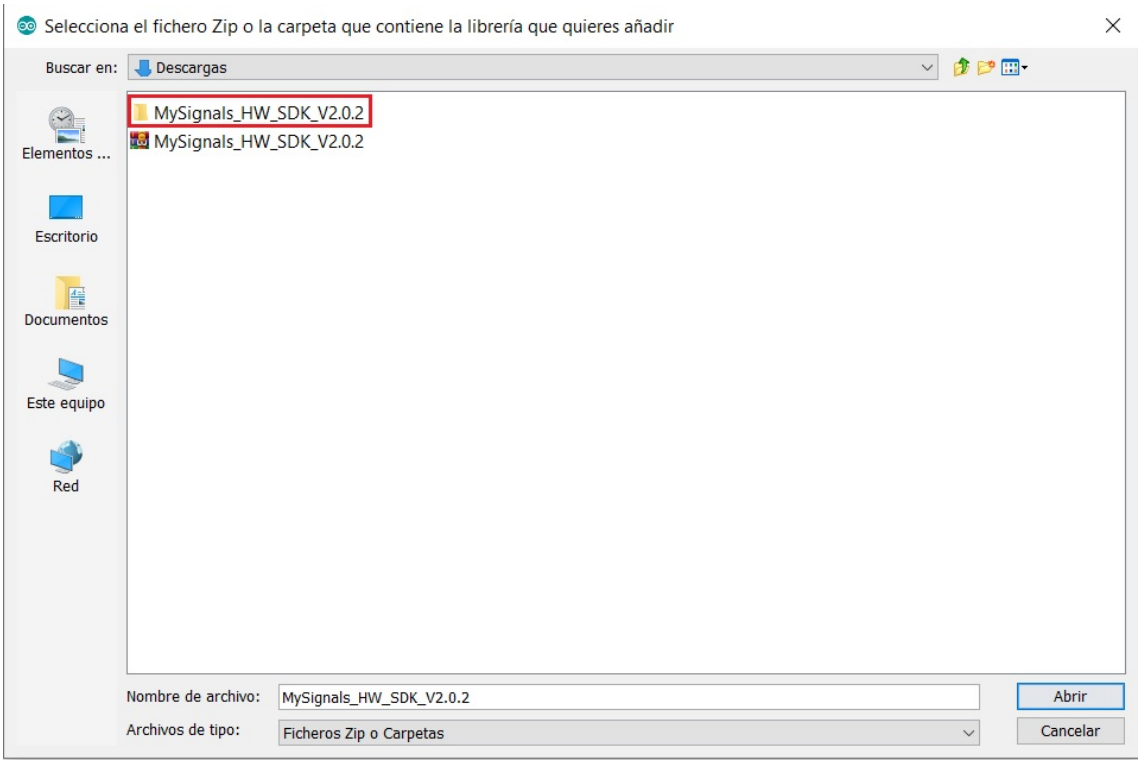

Figura 3.2 Incluir librerías en Arduino.

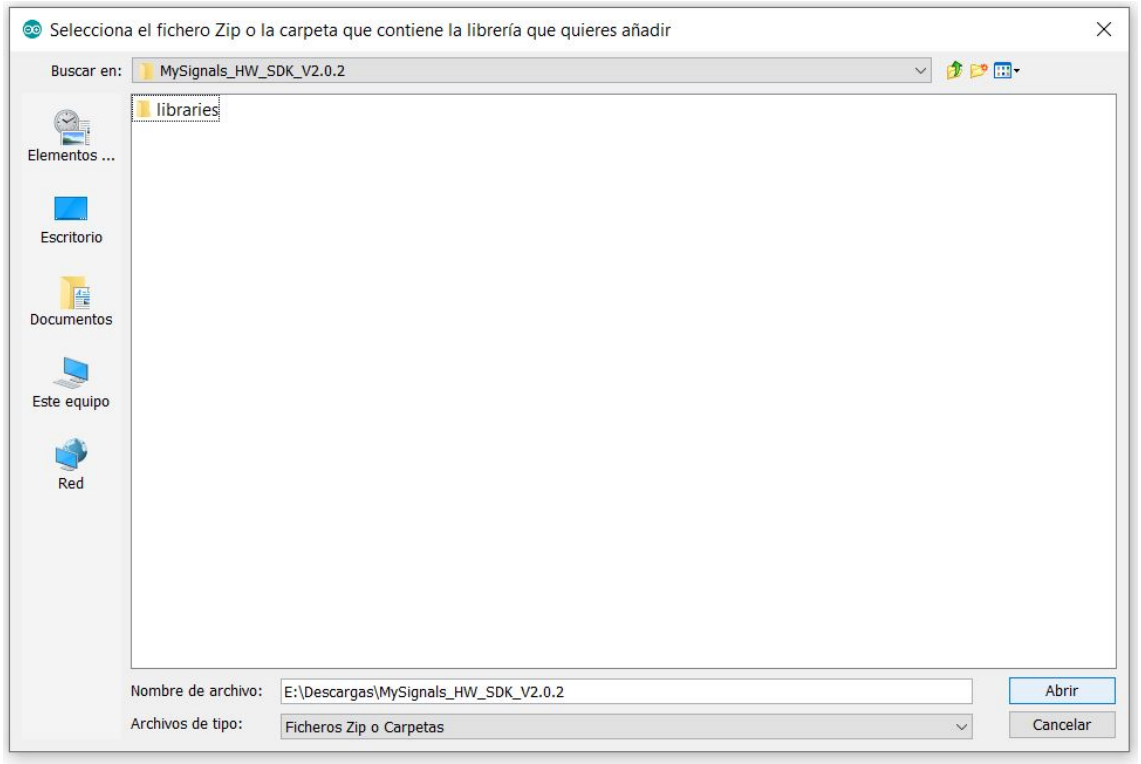

Figura 3.3 Incluir librerías en Arduino.

Y se selecciona la carpeta que se quiere añadir, pulsando **Abrir**:

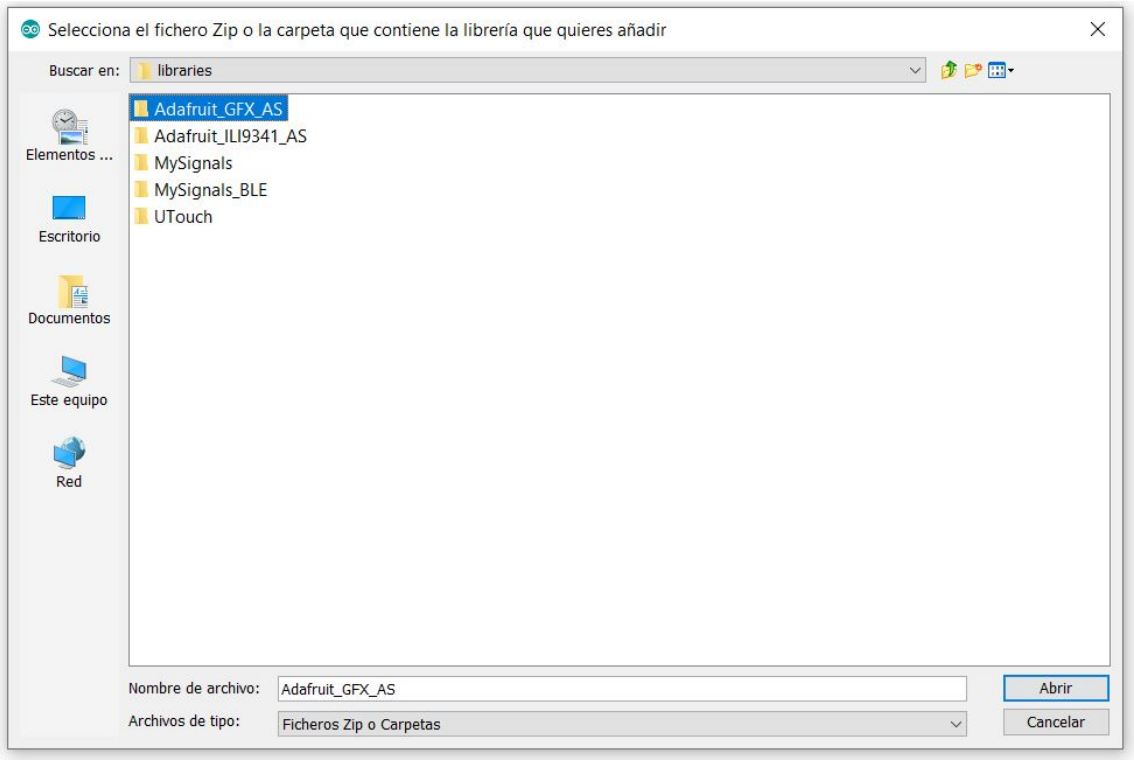

Figura 3.4 Incluir librerías en Arduino.

Si se ha añadido correctamente, sale la siguiente notificación en la parte inferior de Arduino IDE:

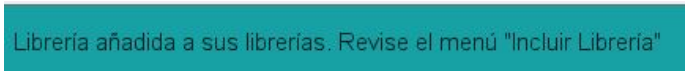

Figura 3.5 Notificación en Arduino.

4. Repetir el paso 3 con todas las librerías del archivo descargado:
   - **Microcontrolador:** Microchip ATmega328P
   - Adafruit GFX AS
   - Adafruit ILI9341 AS
   - Mysignals
   - Mysignals BLE
   - UTouch

3.2 Puesta en marcha

1. **Conexión:**

   Para colocar el escudo sobre Arduino, sólo hay una posible posición para que todos los pines queden conectados. De esta manera no hay posibilidad de error al conectarlo:

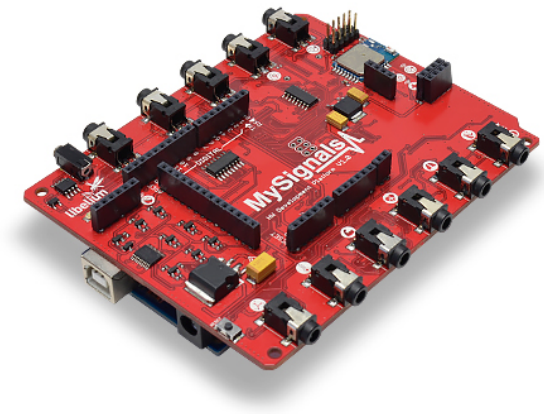

Figura 3.6 Escudo Mysignals Arduino.

   Finalmente, se alimenta el Arduino mediante el puerto USB.

2. **Sensores**

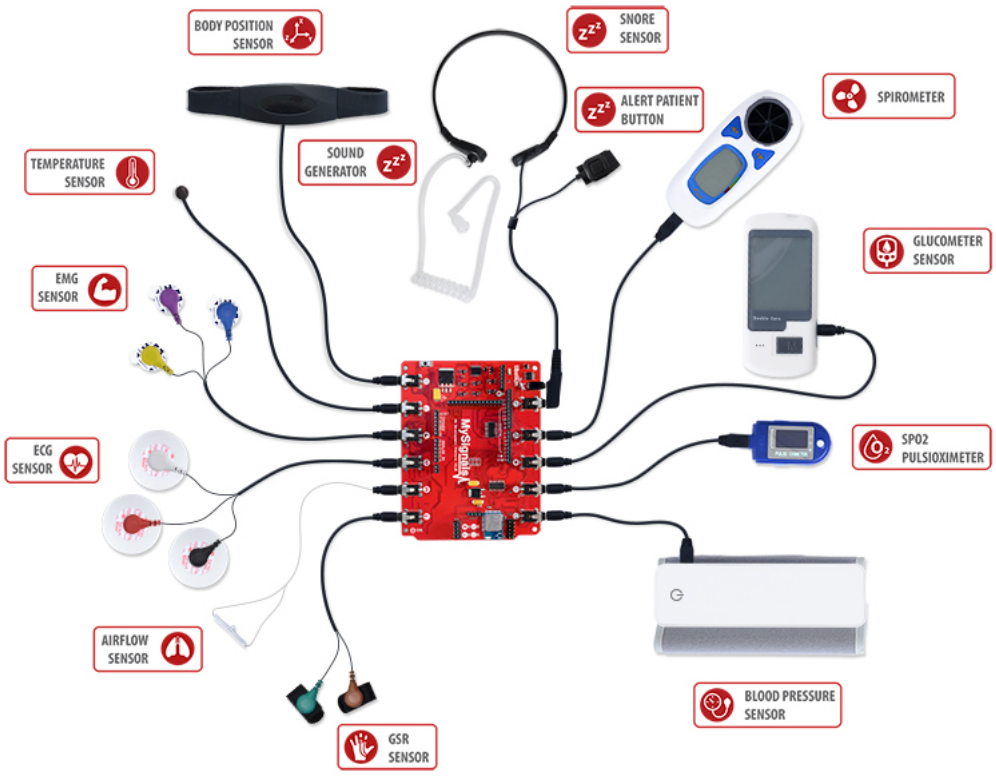

Figura 3.7 Conexionado de Sensores.

En la Figura 3.7 se muestra dónde debe ir colocado el sensor que se va a utilizar. Hay que tener en cuenta el dibujo que representa cada sensor, ya que en el escudo se muestra exactamente el mismo dibujo para tener más claro dónde debe ir conectado.

Así, para el caso del pulsioxímetro se conectará de la siguiente manera:

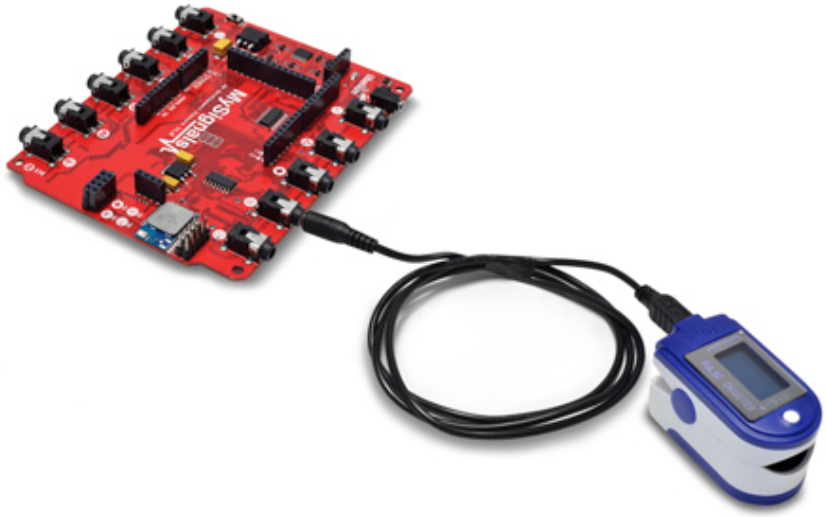

Figura 3.8 Conexión pulsioxímetro.

Y para el caso del tensiómetro:

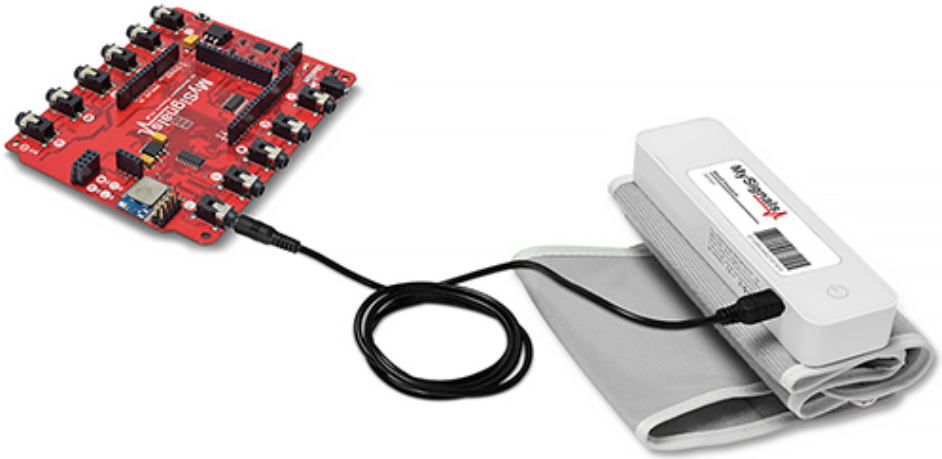

Figura 3.9 Conexión tensiómetro.

3. Ejemplo tensiómetro

Teniendo ya las librerías incluídas, se dispone de los ejemplos del fabricante de MySignals.

A modo de ejemplo, se va a ver en funcionamiento el tensiómetro, el cual indicará a través del puerto serie la tensión sistólica, diastólica y el pulso.

Para ello, se debe compilar y subir el siguiente archivo:
(Ejemplos > MySignals > Sensors > sensor bloodPressure > sensor bloodPressure)

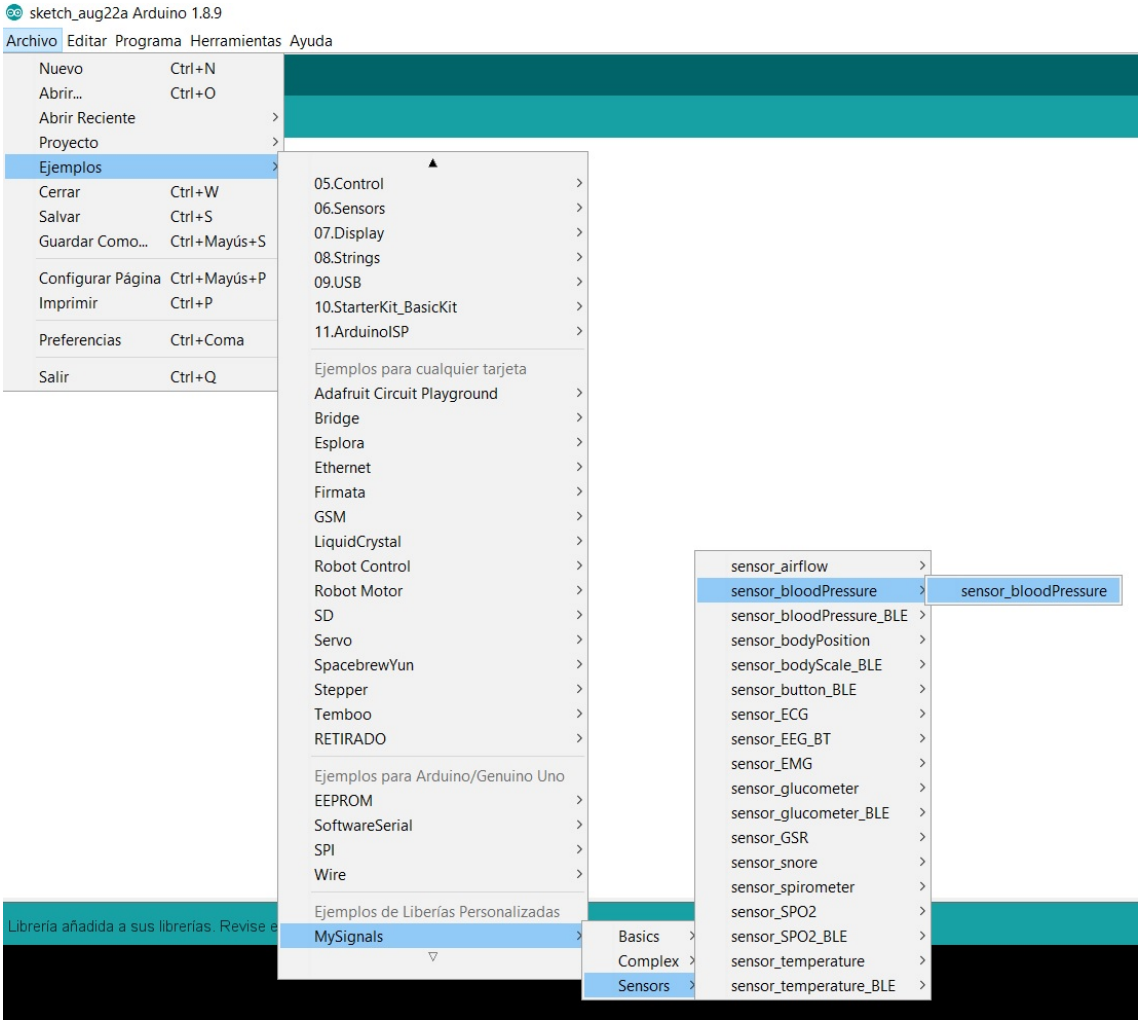

Figura 3.10 Ejemplo tensiómetro.

Antes de compilar y subir, hay que verificar que el Arduino está correctamente conectado en el puerto correspondiente. **(Herramientas > Puerto: …)**

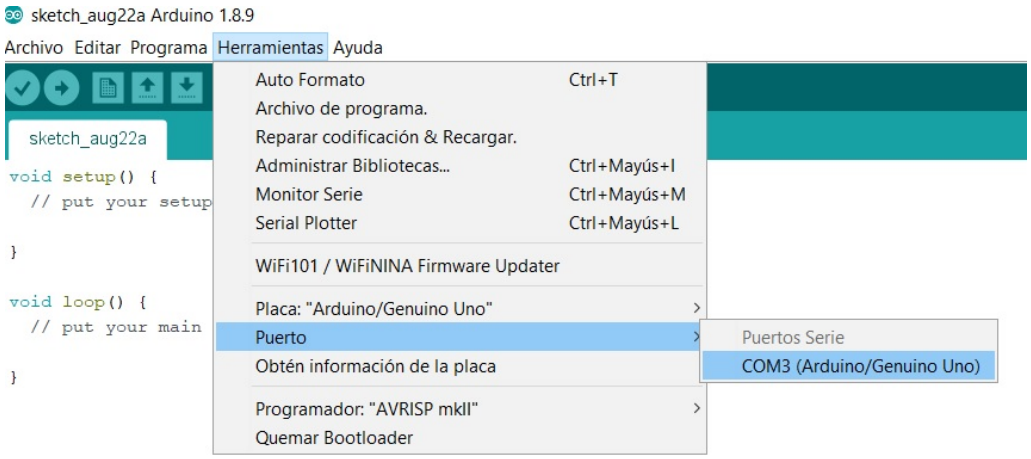

Figura 3.11 Comprobación puerto.

Finalmente, le damos a subir:

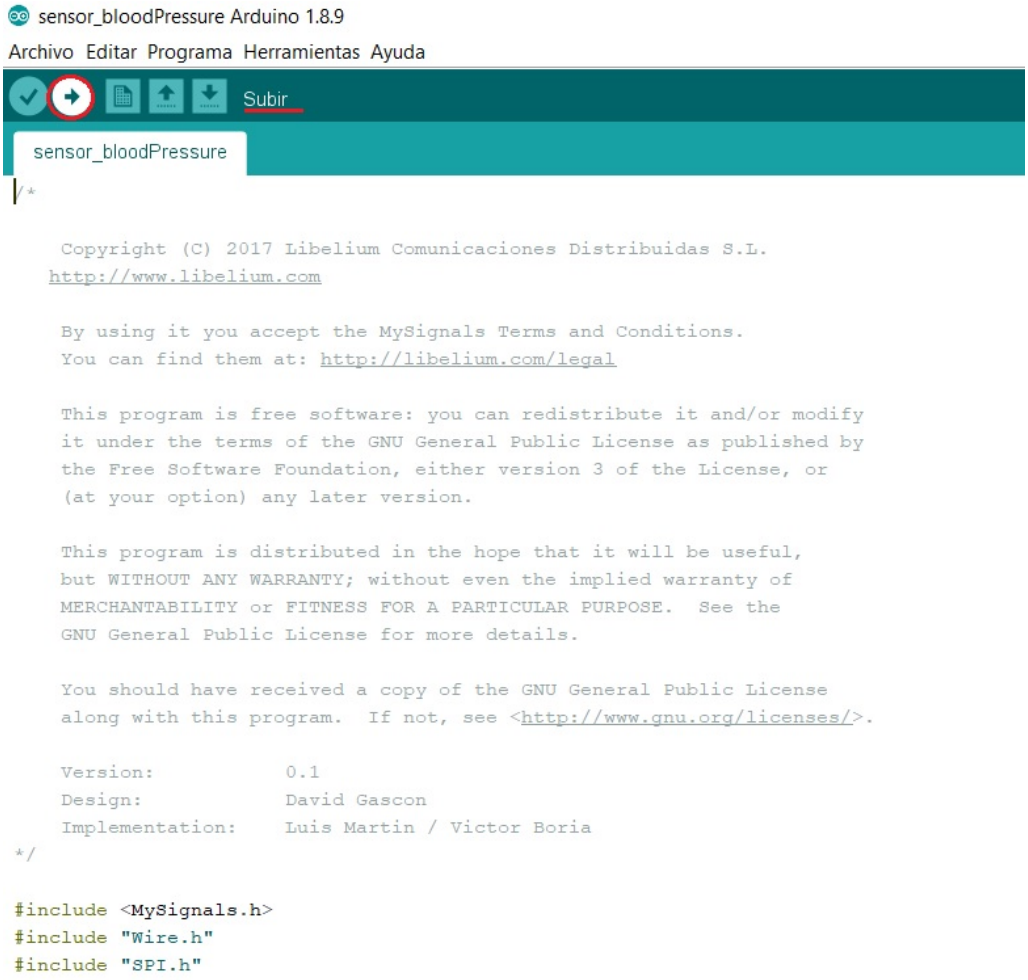

Figura 3.12 Subir ejemplo.

Una vez subido el ejemplo, si no se ha hecho anteriormente, se debe conectar el tensiómetro en el lugar correspondiente, como se muestra en la Figura 3.9. De esta forma, si está correctamente conectado, se encenderá el LED de color rojo del tensiómetro.

Como el ejemplo subido indica los parámetros mediante el puerto serie, se debe abrir el Monitor Serie del Arduino para ver el resultado: **(Herramientas > Monitor Serie)**

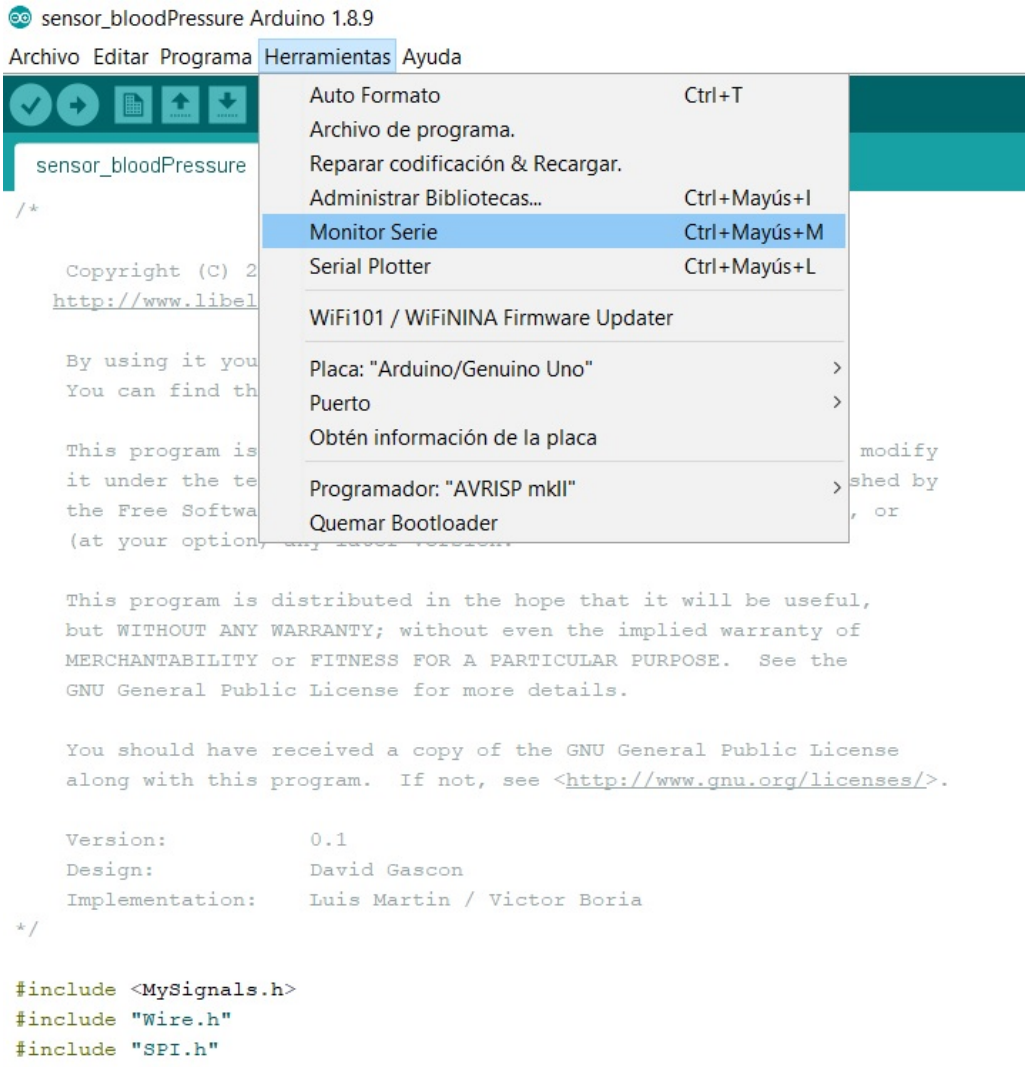

Figura 3.13 Ejecución ejemplo.

Y se configura la velocidad a 19200 baudio, ya que en el ejemplo está inicializado el puerto serie a dicha velocidad:

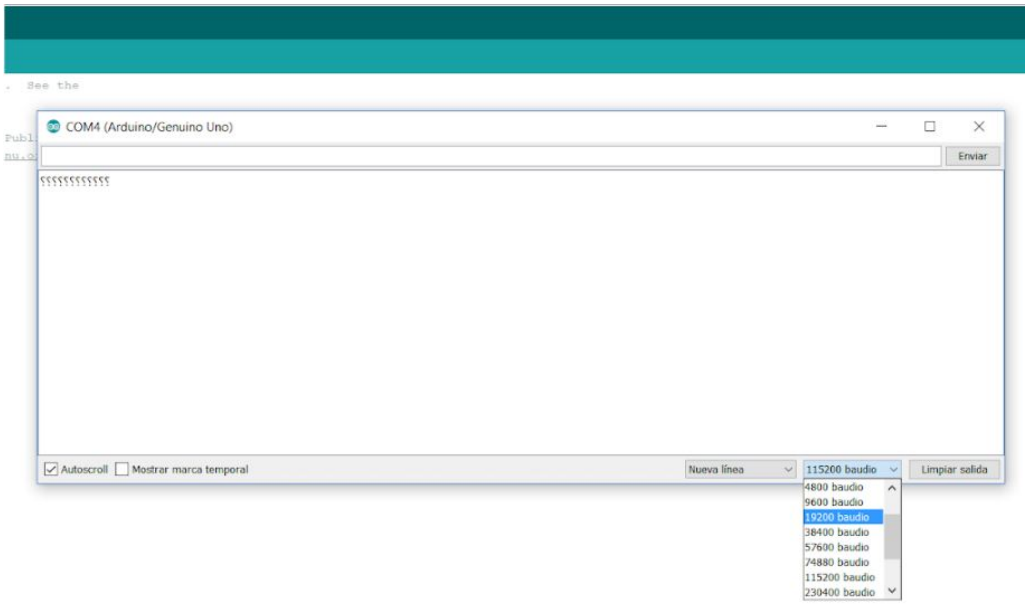

Figura 3.14 Monitor Serie ejemplo.

Finalmente, el usuario se coloca el tensiómetro como aparece en la Figura 3.15, y se enciende el mismo para tomar los parámetros (se encenderá el LED de color azul):

Figura 3.15 Colocación correcta del tensiómetro. Imagen disponible en: https://www.cooking-hacks.com/mysignals-hw-ehealth-medical-biometric-iot-platform-arduino-tutorial/ [Consultado el 22-08-2019].

Y en el monitor serie se muestran los parámetros deseados:

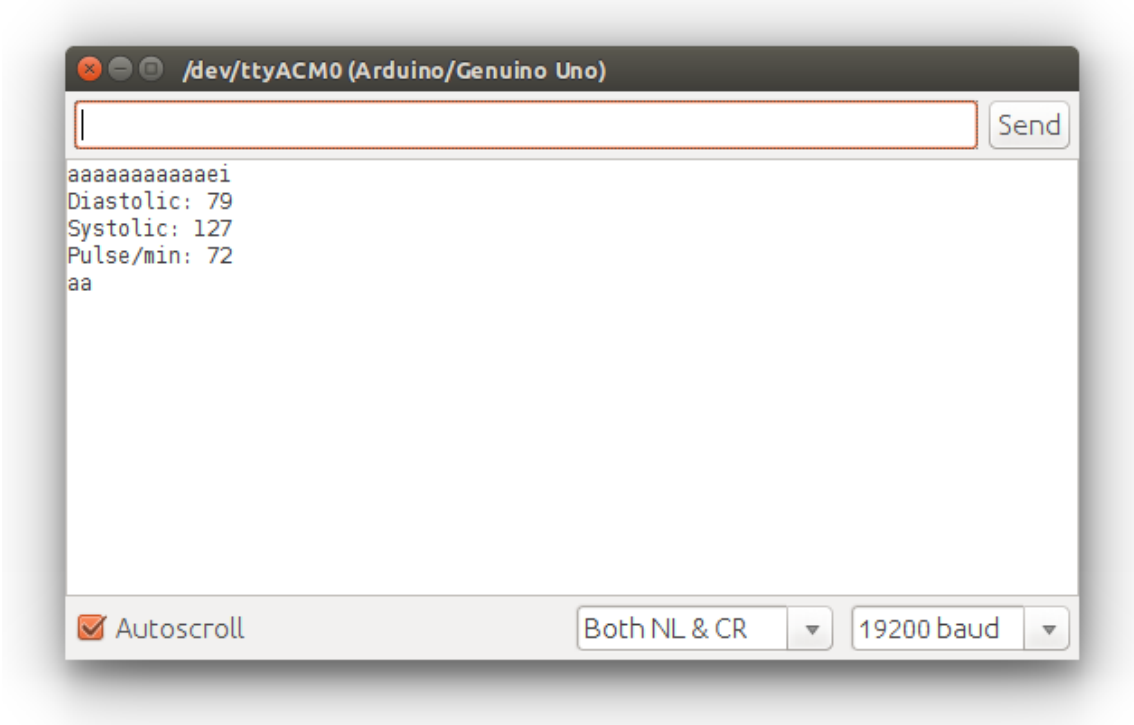

Figura 3.16 Resultados tensiómetro. Imagen disponible en: https://www.cooking-hacks.com/mysignals-hw-ehealth-medical-biometric-iot-platform-arduino-tutorial/ [Consultado el 22-08-2019].

4. Aplicación médica

Para el uso que se le va a dar en este proyecto, será necesario subir a Arduino Uno el programa que se ha realizado para poder utilizar la interfaz gráfica.

Todos los códigos utilizados, tanto el de Arduino como los de Matlab se incluirán en el Anexo. En concreto, en la programación del código de Arduino, en el apéndice A, se ha incluído que el resultado del tensiómetro salga por la pantalla del escudo, la llamada TFT, pero no ha podido probarse experimentalmente.

El código en el que se ha implementado la TFT se puede ver a continuación.

Código 3.1 Programación resultado tensión por TFT.

```
    // Tensión y pulso por pantalla:
      tft.drawString("Diastolic:", 0, 15, 2);
      tft.drawNumber(MySignals.bloodPressureData.diastolic, 100, 15, 2);

      tft.drawString("Systolic:", 0, 30, 2);
      tft.drawNumber(MySignals.bloodPressureData.systolic, 100, 30, 2);

      tft.drawString("Pulse/min:", 0, 45, 2);
      tft.drawNumber(MySignals.bloodPressureData.pulse, 100, 45, 2);
```

Por otra parte, la comunicación de Arduino con Matlab se lleva cabo a través del puerto serie, cuyo protocolo de comunicación funciona igual que un multiplexor. Por ejemplo, si Arduino recibe por puerto serie el número '111', a éste se le ha programado que se debe medir la tensión.

4 Tensiómetro

En este capítulo se detallará el funcionamiento del tensiómetro, así como su puesta en marcha en la interfaz gráfica del usuario.

4.1 Qué es

El tensiómetro es un instrumento médico que se utiliza para medir la presión arterial, normalmente expresada en milímetros de mercurio (mmHg).

El que se va a utilizar para este proyecto es el que se puede ver en la Figura 4.1.

Figura 4.1 Tensiómetro. Imagen disponible en: https://www.cooking-hacks.com/mysignals-hw-ehealth-medical-biometric-iot-platform-arduino-tutorial/ [Consultado el 19-08-2019].

4.2 Funcionamiento

Una vez situado el tensiómetro en el brazo e iniciado, éste comienza a añadir presión con aire hasta superar la presión sanguínea, momento en el que se aplasta la arteria braquial impidiendo durante un instante el flujo de sangre al antebrazo.
Posteriormente, el tensiómetro va soltando aire de la bolsa de forma gradual hasta alcanzar la presión sistólica del paciente, que es cuando se oye el primer sonido de Korotkov.

```
Sonido de Korotkov: Sonido que se oye cuando la sangre empieza a fluir por una
    arteria tras eliminar una obstrucción.
```

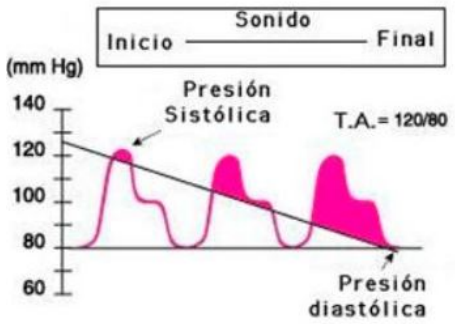

Figura 4.2 Funcionamiento del Tensiómetro. Imagen disponible en: https://mitensiometro.com/como-funciona-un-tensiometro/ [Consultado el 19-08-2019].

El tensiómetro continúa disminuyendo la presión del brazalete hasta llegar al silencio, que ocurre cuando la presión del brazalete cae por debajo de la presión diastólica.

4.3 Valores normales

Los valores normales para la presión **sistólica** o tensión máxima es de **80-120 mmHg**. Y para la presión mínima o **diastólica**, los valores normales oscilan entre **60-80 mmHg**.

Según el rango en el que se encuentre la presión arterial, estaría en un grado u otro de gravedad, como se puede observar en la Figura 4.3.

Grados de presión arterial

Categoría	Sistólica (mmHg)		Diastólica (mmHg)
Hipotensión	menor de 80	o	menor de 60
Normal	80-120	y	60-80
Prehipertensión	120-139	o	80-89
Hipertensión grado 1 (HTA 1)	140-159	o	90-99
Hipertensión grado 2 (HTA 2)	160 o superior	o	100 o superior
Crisis hipertensiva (emergencia médica)	superior a 180	o	superior a 110

Fuente: American Heart Association

Figura 4.3 Grados de presión arterial. Imagen disponible en: https://www.comprartensiometro.net/tension-arterial/cuales-son-los-valores-normales-de-tension-arterial/ [Consultado el 19-08-2019].

4.4 Interfaz gráfica

La interfaz gráfica que se ha implementado en este trabajo permite al usuario utilizar el tensiómetro de una manera muy sencilla. Además, da la posibilidad de ver el progreso semanal de la tensión, ya que al realizar la prueba el resultado queda guardado en el archivo propio del usuario.

A continuación se muestran varias imágenes con el uso de la interfaz gráfica referente al tensiómetro.

Para iniciar la interfaz gráfica del usuario se debe ejecutar el fichero **INICIO**, y si no se conecta el aparato aparece lo que se puede observar en la Figura 4.4. En este caso, habría que conectarlo al ordenador y volver a iniciar la interfaz.

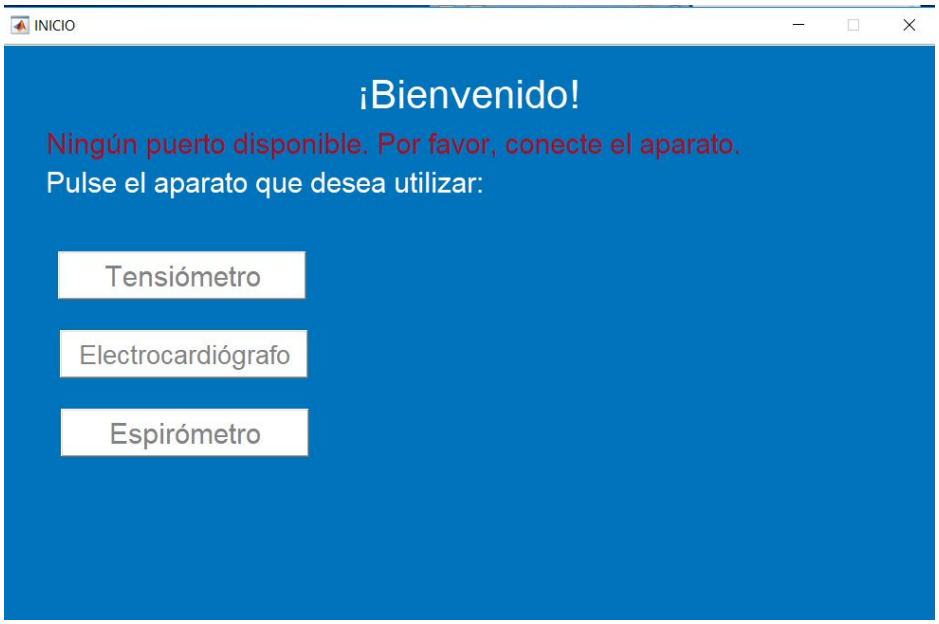

Figura 4.4 GUI Inicio.

Una vez conectado el aparato, Figura 4.5, si todo está correcto aparece el puerto disponible en la lista de arriba a la izquierda, y una vez seleccionado, ya es posible acceder a la interfaz del aparato deseado.
Para el caso del tensiómetro, en primer lugar aparece lo que se puede observar en la Figura 4.6. Una vez introducido el usuario (escribir el nombre y darle a Intro), hay dos posibilidades según lo que quiera el usuario: Darle a conectar para realizar una prueba o ver el progreso de la semana seleccionada por el usuario.

Supongamos que se va a realizar el proceso completo:

1. Darle a **Conectar** sin estar conectado el tensiómetro, Figura 4.7, cuyo Estado aparece como Desconectado.

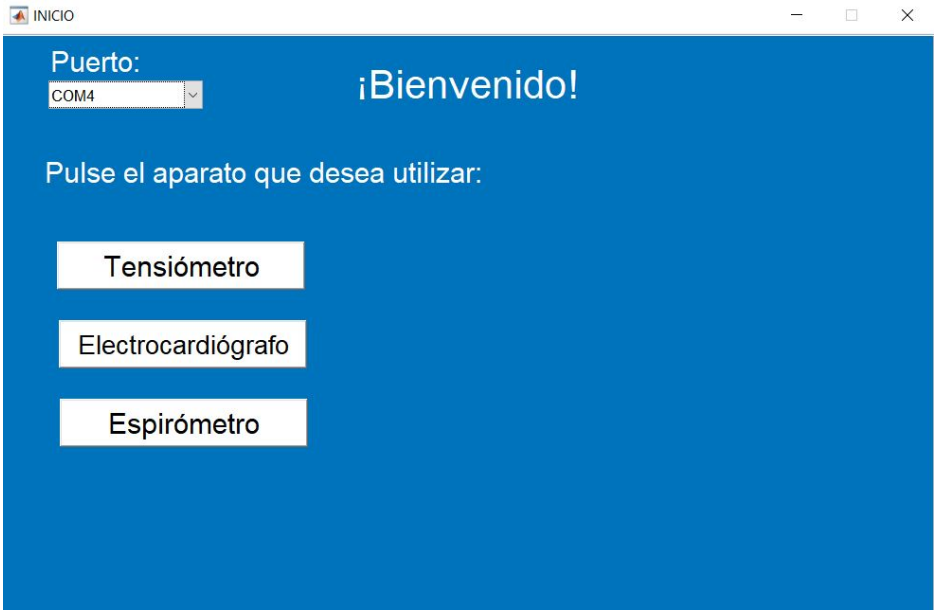

Figura 4.5 GUI Inicio.

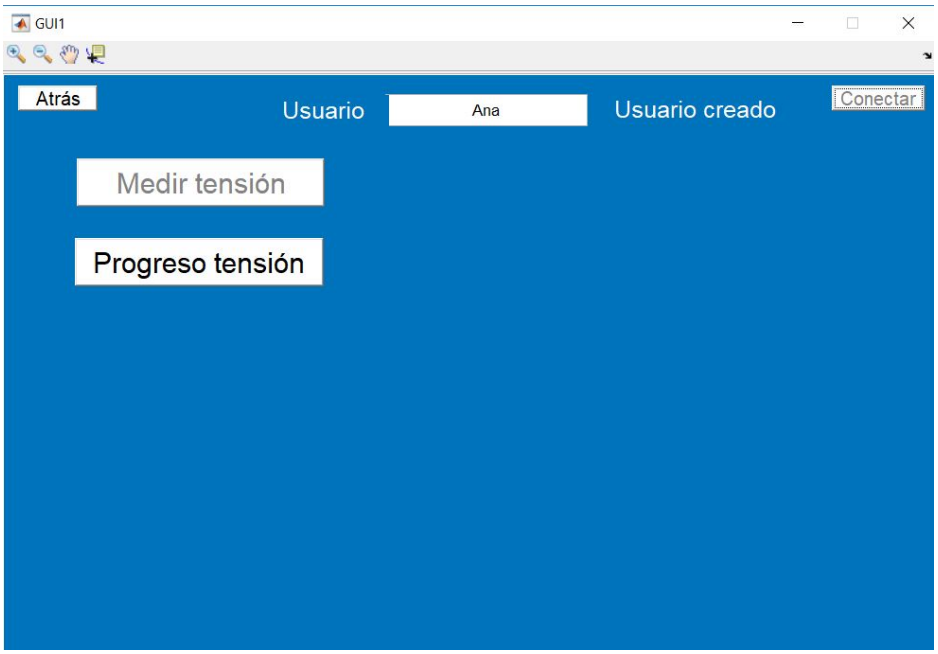

Figura 4.6 GUI Tensiómetro.

2. Darle a **Conectar** una vez conectado y encendido el tensiómetro, Figura 4.8, cuyo Estado ya sí aparece como Conectado. En este caso se activa el botón de **Medir tensión**, con lo que es posible realizar la prueba.

3. Se procede a realizar la prueba. El usuario debe colocarse el tensiómetro como aparece en la Figura 4.9 y encenderlo hasta que aparezca el Led de color azul. Posteriormente, ya podrá darle al botón de **Medir tensión** en la interfaz. Es importante que el usuario permanezca quieto durante la prueba, pues de lo contrario el resultado se vería alterado.

Figura 4.7 GUI Tensiómetro.

Figura 4.8 GUI Tensiómetro.

4. Una vez realizada la prueba ya se puede hacer el diagnóstico según los datos obtenidos. Para ello, hay que pulsar el botón **Diagnóstico**, como se puede ver en la Figura 4.10. Los datos utilizados para la programación del diagnóstico se verá en el siguiente apartado.

   Para el caso del ejemplo, el resultado es normal (Figura 4.11).

Figura 4.9 Posición correcta del tensiómetro. Imagen disponible en: https://www.cooking-hacks.com/mysignals-hw-ehealth-medical-biometric-iot-platform-arduino-tutorial/ [Consultado el 20-08-2019].

Figura 4.10 GUI Tensiómetro.

5. Finalmente, si se quiere ver el progreso semanal de la tensión basta con pulsar el botón de **Progreso tensión**. En ese momento se despliega en el menú las semanas en las que se han realizado las pruebas, por lo que el usuario puede elegir la semana a mostrar en la gráfica. (Ver Figura 4.12).

   Una vez seleccionada la semana, se representa gráficamente la tensión sistólica, diastólica y el pulso de esa semana, como se puede ver en la Figura 4.13.

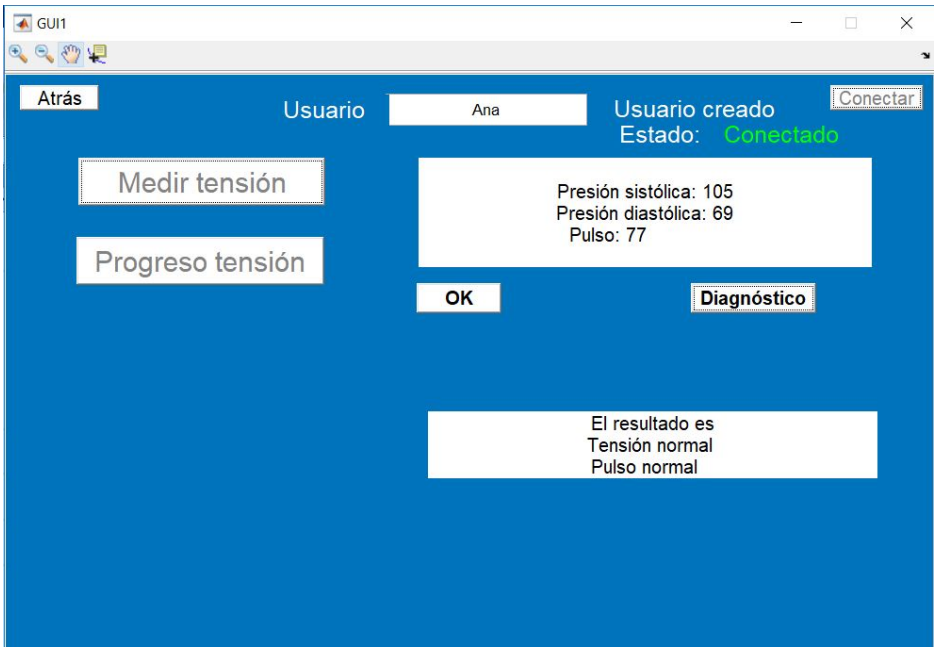

Figura 4.11 GUI Tensiómetro.

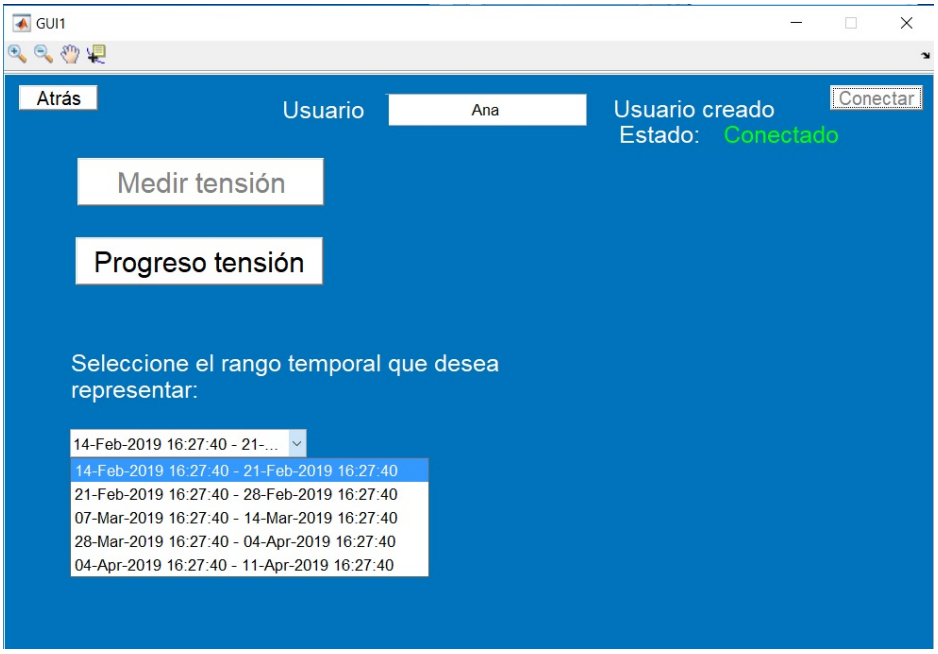

Figura 4.12 GUI Tensiómetro.

4.5 Diagnóstico

Para el diagnóstico de la interfaz gráfica se han utilizado los datos de la Figura 4.3 para la tensión, es decir:

- **Hipotensión:** Sistólica<80 mmHg o Diastólica<60 mmHg.
- **Normal:** Sistólica = 80-120 mmHg y Diastólica=60-80 mmHg.
- **Prehipertensión:** Sistólica = 120-139 mmHg o Diastólica=80-89 mmHg.
- **Hipertensión Grado 1:** Sistólica = 140-159 mmHg o Diastólica=90-99 mmHg.
- **Hipertensión Grado 2:** Sistólica>160 mmHg o Diastólica>100 mmHg.
- **Crisis hipertensiva:** Sistólica>180 mmHg o Diastólica>110 mmHg.

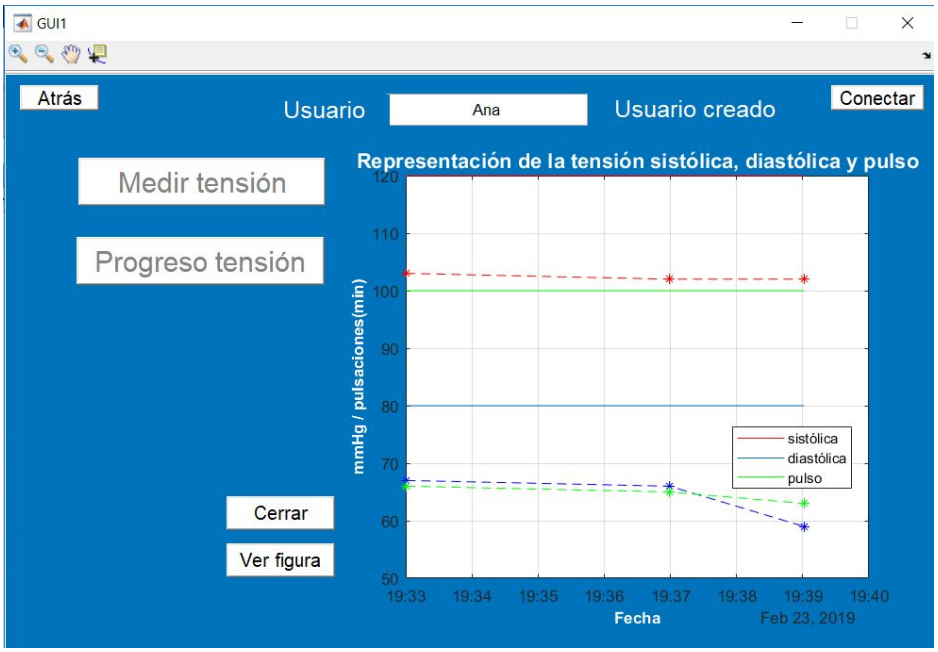

Figura 4.13 GUI Tensiómetro.

Y respecto al pulso, se tiene en cuenta el siguiente rango:

- **Braquicardia:** Pulso<50 latidos por minuto.
- **Taquicardia:** Pulso>100 latidos por minuto.
- **Otro caso:** Pulso normal.

El código en el que se ha implementado el diagnóstico en Matlab es el siguiente:

Código 4.1 Programación diagnóstico tensión.

```
nombre=get(handles.edit1,'string'); %Nombre de usuario
filename=strcat(nombre,'.mat');
load(filename,'progresoDato'); %Carga el fichero del usuario
pruebas = getByName(progresoDato, 'tension'); %Busca las pruebas del tensió
    metro
systolic = pruebas{end,1}.systolic %Coge la tensión sistólica de la última
    prueba
diastolic = pruebas{end,1}.diastolic %Coge la tensión diastólica de la última
    prueba
pulse = pruebas{end,1}.pulse %Coge el pulso de la última prueba

if ((systolic<80)||(diastolic<60))
    diagnostico='Hipotensión';
elseif ((systolic<120)&&(diastolic<80))
    diagnostico='Tensión normal';
elseif ((systolic<140)||(diastolic<90))
    diagnostico='Prehipertension';
elseif ((systolic<160)||(diastolic<100))
    diagnostico='Hipertensión grado 1';
elseif ((systolic<180)||(diastolic<110))
    diagnostico='Hipertensión grado 2';
else
    diagnostico='Crisis hipertensiva (EMERGENCIA)';
```

```
end


if (pulse>100)
    diagnostico = strvcat(diagnostico,'Taquicardia');
elseif (pulse<50)
    diagnostico = strvcat(diagnostico,'Braquicardia');
else
    diagnostico = strvcat(diagnostico,'Pulso normal');
end
```

5 Espirómetro

En este capítulo se detallará el funcionamiento del espirómetro, así como su puesta en marcha en la interfaz gráfica del usuario.

5.1 Qué es

El espirómetro es un instrumento médico que se utiliza para medir el volumen de aire expirado en una inspiración forzada. Esto sirve para detectar enfermedades pulmonares obstructivas, Asma y EPOC.

El que se va a utilizar para este proyecto es el que se puede ver en la Figura 5.1.

Figura 5.1 Espirómetro. Imagen disponible en: https://www.cooking-hacks.com/mysignals-hw-ehealth-medical-biometric-iot-platform-arduino-tutorial/ [Consultado el 20-08-2019].

5.2 Funcionamiento

La medida se realiza a través de un neumotacógrafo, siendo en este caso de turbina.

Los neumotacógrafos de turbina tienen un eje sobre el que gira una pequeña hélice con el flujo del aire. Las aspas de dicha hélice interrumpen una fuente de luz en cada paso que realizan, por lo que a más flujo de aire, más velocidad alcanzan las aspas y más veces se interrumpe la señal luminosa. Esta información se envía al microprocesador, que calcula el flujo de aire en función de las revoluciones de la hélice, y, por integración, calcula el volumen.

5.3 Valores normales

En primer lugar, cabe destacar que el dato que da el espirómetro utilizado para este proyecto es el llamado **Fev1**.

```
Fev1: es el volumen de aire que se expulsa durante el primer segundo de la
    espiración forzada.
```

Los valores normales para Fev1 depende del sexo, edad y altura del usuario, según las tablas siguientes:

FEV_1 (Litres)											MALE
Height	145	150	155	160	165	170	175	180	185	190	195
Age 10	2.31	2.54	2.77	3.00	3.23	3.46	3.69	3.92	4.15	4.38	4.61
12	2.40	2.63	2.86	3.09	3.32	3.55	3.78	4.01	4.24	4.47	4.70
14	2.49	2.72	2.95	3.18	3.41	3.64	3.87	4.10	4.33	4.56	4.79
16	2.58	2.81	3.04	3.27	3.50	3.73	3.96	4.19	4.42	4.65	4.88
18	2.67	2.90	3.13	3.36	3.59	3.82	4.05	4.28	4.51	4.74	4.97
20	2.76	2.99	3.22	3.45	3.68	3.91	4.14	4.37	4.60	4.83	5.06
25	2.66	2.92	3.18	3.44	3.70	3.96	4.22	4.48	4.74	5.00	5.26
30	2.53	2.79	3.05	3.31	3.57	3.83	4.09	4.35	4.61	4.87	5.13
40	2.26	2.52	2.78	3.04	3.30	3.56	3.82	4.08	4.34	4.60	4.86
50	1.99	2.25	2.51	2.77	3.03	3.29	3.55	3.81	4.07	4.33	4.59
60	1.72	1.98	2.24	2.50	2.76	3.02	3.28	3.54	3.80	4.06	4.32
70	1.45	1.71	1.97	2.23	2.49	2.75	3.01	3.27	3.53	3.79	4.05
80	1.18	1.44	1.70	1.96	2.22	2.48	2.74	3.00	3.26	3.52	3.78

Figura 5.2 Fev1 en hombres. Imagen disponible en: https://www.ugr.es/ jhuertas/EvaluacionFisiologica/Espirometria/volteoricostablas.htm [Consultado el 20-08-2019].

FEV_1 (litres)											FEMALE
Height	145	150	155	160	165	170	175	180	185	190	195
Age 10	2.06	2.20	2.33	2.47	2.60	2.74	2.87	3.01	3.14	3.28	3.41
12	2.23	2.37	2.50	2.64	2.77	2.91	3.04	3.18	3.31	3.45	3.58
14	2.40	2.54	2.67	2.81	2.94	3.08	3.21	3.35	3.48	3.62	3.75
16	2.57	2.71	2.84	2.98	3.11	3.25	3.38	3.52	3.65	3.79	3.92
18	2.74	2.88	3.01	3.15	3.28	3.42	3.55	3.69	3.82	3.96	4.09
20	2.70	2.84	2.97	3.11	3.24	3.38	3.51	3.65	3.78	3.92	4.05
25	2.60	2.73	2.87	3.00	3.14	3.27	3.41	3.54	3.68	3.81	3.95
30	2.49	2.63	2.76	2.90	3.03	3.17	3.30	3.44	3.57	3.71	3.84
40	2.28	2.42	2.55	2.69	2.82	2.96	3.09	3.23	3.36	3.50	3.63
50	2.07	2.21	2.34	2.48	2.61	2.75	2.88	3.02	3.15	3.29	3.42
60	1.86	2.00	2.13	2.27	2.40	2.54	2.67	2.81	2.94	3.08	3.21
70	1.65	1.79	1.92	2.06	2.19	2.33	2.46	2.60	2.73	2.87	3.00
80	1.44	1.58	1.71	1.85	1.98	2.12	2.25	2.39	2.52	2.66	2.79

Figura 5.3 Fev1 en mujeres. Imagen disponible en: https://www.ugr.es/ jhuertas/EvaluacionFisiologica/Espirometria/volteoricostablas.htm [Consultado el 20-08-2019].

5.4 Interfaz gráfica

La interfaz gráfica que se ha desarrollado permite al usuario utilizar el espirómetro de una manera muy sencilla. En este caso, también se almacena el dato de la prueba en el archivo del usuario, pero a diferencia del tensiómetro, sólo se puede ver el dato del día seleccionado. Se ha programado de esta manera porque la espirometría es una prueba que no se realiza todos los días y no necesita un progreso semanal.

Para el caso del espirómetro, primero será necesario realizar la prueba y almacenar los datos en el propio espirómetro y, posteriormente, leer los datos desde la interfaz gráfica.

A continuación se explica el procedimiento:

1. Colocar la boquilla tal y como aparece en la Figura 5.4 y encenderlo.
2. Posicionarlo como en la Figura 5.5 y realizar la espirometría forzada, es decir, soltar todo el aire bruscamente hasta no poder más.
3. Guardar la prueba en el dispositivo manteniendo pulsado el botón de la derecha (Figura 5.6) hasta oír un pitido.

Figura 5.4 Primer paso espirómetro.

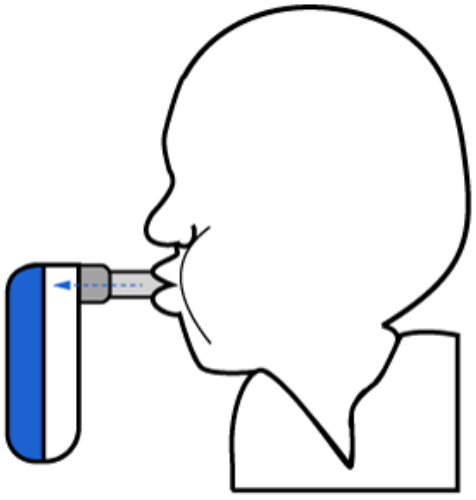

Figura 5.5 Colocación correcta del espirómetro.

4. Guardar la prueba a través de la interfaz gráfica. Se va a explicar el procedimiento desde el Inicio al igual que con el tensiómetro:

   Al iniciar la interfaz gráfica del usuario, si no se conecta el aparato, aparece lo que se puede observar en la Figura 5.7. En este caso, habría que conectarlo al ordenador y volver a iniciar la interfaz.

Figura 5.6 Colocación correcta del espirómetro.

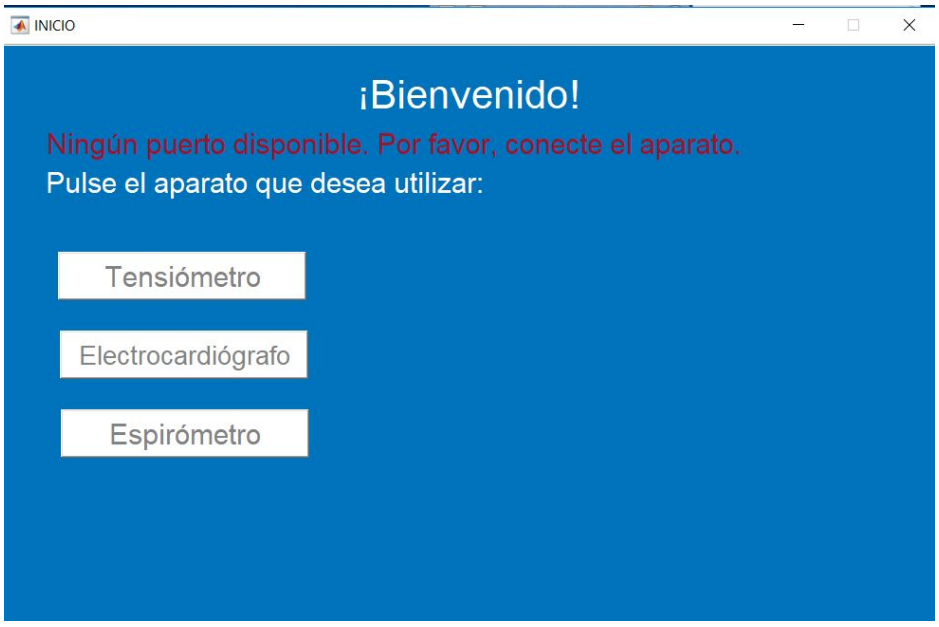

Figura 5.7 GUI Inicio.

Una vez conectado el aparato, Figura 5.8, si todo está correcto aparece el puerto disponible en la lista de arriba a la izquierda, y una vez seleccionado, ya es posible acceder a la interfaz del aparato deseado.

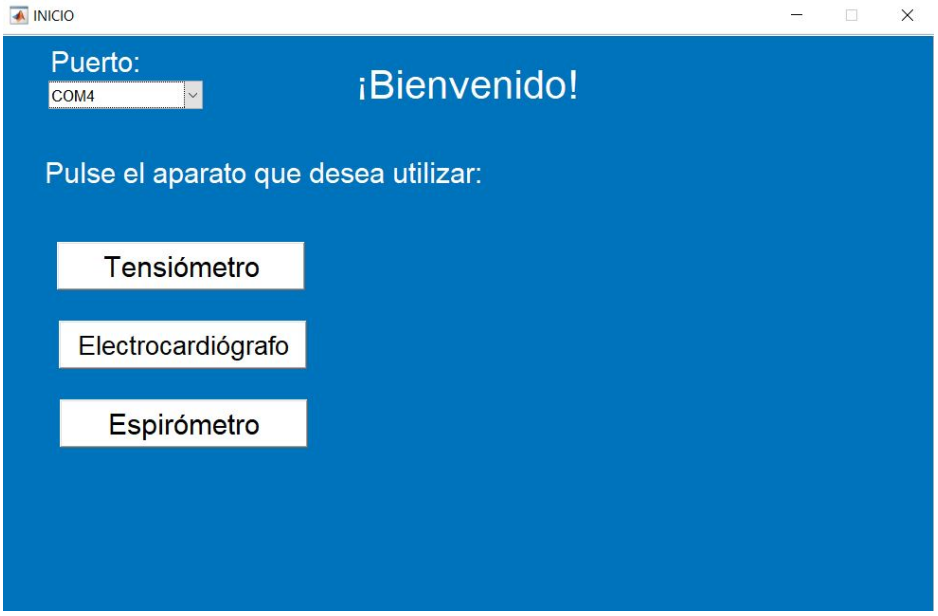

Figura 5.8 GUI Inicio.

En la Figura 5.9 se está leyendo el dato almacenado en el espirómetro. Es importante destacar que según la programación **únicamente va a leer el primer dato almacenado**, por lo que después de que el dato quede almacenado en el archivo de usuario, sería necesario borrar la memoria del espirómetro para realizar una nueva prueba. Eso se puede hacer dándole al botón de **Borrar medidas**.

Figura 5.9 GUI Inicio.

Por último, se le da a **diagnóstico**. Es necesario indicar la edad, altura y el sexo, ya que el diagnóstico varía según esos parámetros, como se puede ver en la Figura 5.10.

Figura 5.10 GUI Inicio.

5.5 Diagnóstico

Para el diagnóstico de la espirometría se tiene en cuenta el siguiente porcentaje:

$$\frac{FEV_1}{FVC}(\%) \tag{5.1}$$

```
FVC: es el volumen total que se expulsa desde la inspiración máxima hasta la
    espiración máxima.

Fev1: es el volumen de aire que se expulsa durante el primer segundo de la
    espiración forzada.

Así, la Ecuación 5.1 indica Indica el porcentaje del volumen total espirado en
    el primer segundo.
```

Según el resultado de la Ecuación 5.1, se tiene el siguiente diagnóstico:

- **Mayor a 80 %:** Diagnóstico normal
- **Entre 80 % y 50 %:** Diagnóstico leve
- **Entre 50 % y 30 %:** Diagnóstico moderado
- **Menor a 30 %:** Diagnóstico grave (Pulmonar obstructivo)

El dato **FEV1** es el que resulta de la espirometría, y el dato de **FVC** lo tomaremos de las siguientes tablas normalizadas, según el sexo, edad y altura del usuario:

FVC (Litres)											MALE
Height	145	150	155	160	165	170	175	180	185	190	195
Age 10	2.52	2.77	3.02	3.27	3.52	3.77	4.02	4.27	4.52	4.77	5.02
12	2.68	2.93	3.18	3.43	3.68	3.93	4.18	4.43	4.68	4.93	5.18
14	2.83	3.08	3.33	3.58	3.83	4.08	4.33	4.58	4.83	5.08	5.33
16	2.99	3.24	3.49	3.74	3.99	4.24	4.49	4.74	4.99	5.24	5.49
18	3.15	3.40	3.65	3.90	4.15	4.40	4.65	4.90	5.15	5.40	5.65
20	3.30	3.55	3.80	4.05	4.30	4.55	4.80	5.05	5.30	5.55	5.80
25	3.24	3.57	3.89	4.22	4.54	4.87	5.19	5.52	5.84	6.17	6.49
30	3.10	3.42	3.75	4.07	4.40	4.72	5.05	5.37	5.70	6.02	6.35
40	2.81	3.13	3.46	3.78	4.11	4.43	4.76	5.08	5.41	5.73	6.06
50	2.52	2.84	3.17	3.49	3.82	4.14	4.47	4.79	5.12	5.44	5.77
60	2.23	2.55	2.88	3.20	3.53	3.85	4.18	4.50	4.83	5.15	5.48
70	1.94	2.26	2.59	2.91	3.24	3.56	3.89	4.21	4.54	4.86	5.19
80	1.65	1.97	2.30	2.62	2.95	3.27	3.60	3.92	4.25	4.57	4.90

Figura 5.11 FVC en hombres. Imagen disponible en: https://www.ugr.es/ jhuertas/EvaluacionFisiologica/Espirometria/volteoricostablas.htm [Consultado el 21-08-2019].

FVC (litres)											FEMALE
Height	145	150	155	160	165	170	175	180	185	190	195
Age 10	2.24	2.40	2.57	2.73	2.90	3.06	3.23	3.39	3.56	3.72	3.89
12	2.42	2.59	2.75	2.92	3.08	3.25	3.41	3.58	3.74	3.91	4.07
14	2.60	2.77	2.93	3.10	3.26	3.43	3.59	3.76	3.92	4.09	4.25
16	2.79	2.95	3.12	3.28	3.45	3.61	3.78	3.94	4.11	4.27	4.44
18	2.97	3.14	3.30	3.47	3.63	3.80	3.96	4.13	4.29	4.46	4.62
20	3.15	3.34	3.52	3.71	3.89	4.08	4.26	4.45	4.63	4.82	5.00
25	3.04	3.23	3.41	3.60	3.78	3.97	4.15	4.34	4.52	4.71	4.89
30	2.93	3.12	3.30	3.49	3.67	3.86	4.04	4.23	4.41	4.60`	4.78
40	2.71	2.90	3.08	3.27	3.45	3.64	3.82	4.01	4.19	4.38	4.56
50	2.49	2.68	2.86	3.05	3.23	3.42	3.60	3.79	3.97	4.16	4.34
60	2.27	2.46	2.64	2.83	3.01	3.20	3.38	3.57	3.75	3.94	4.12
70	2.05	2.24	2.42	2.61	2.79	2.98	3.16	3.35	3.53	3.72	3.90
80	1.83	2.02	2.20	2.39	2.57	2.76	2.94	3.13	3.31	3.50	3.68

Figura 5.12 FVC en mujeres. Imagen disponible en: https://www.ugr.es/ jhuertas/EvaluacionFisiologica/Espirometria/volteoricostablas.htm [Consultado el 20-08-2019].

Así, el código en el que se ha implementado el diagnóstico en Matlab es el siguiente:

Código 5.1 Programación diagnóstico espirómetro.

```
% --- Executes on button press in pushbutton16.
function pushbutton16_Callback(hObject, eventdata, handles)
% hObject   handle to pushbutton16 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles   structure with handles and user data (see GUIDATA)

valor=get(handles.text4,'String') %Valor de FEV1 obtenido en str
[i,j]=size(valor)
Fev=str2double(valor(2,9:(j-2))) %Se pasa a numero el dato
edad=str2double(get(handles.edit5,'String'))
altura=str2double(get(handles.edit3,'String'))
hombre=get(handles.radiobutton3,'Value');
mujer=get(handles.radiobutton6,'Value');
if (mujer==1)
    Fvc=diagnosticoSPR(edad,altura,'M')
else
    Fvc=diagnosticoSPR(edad,altura,'H')
end
resultado=Fev/Fvc*100
if (resultado>80)
    diagnostico='Diagnostico normal';
elseif (resultado>50)
    diagnostico='Diagnostico leve';
elseif (resultado>30)
    diagnostico='Diagnostico moderado';
else
    diagnostico='Diagnostico grave (pulmonar obstructivo)';
end

texto=(['El resultado es',10,num2str(resultado),' %',10,diagnostico])
set(handles.text15,'String',texto);
set(handles.text15,'Visible','On');
```

Y el código donde se han programado las tablas:

Código 5.2 Programación diagnóstico espirómetro. Tablas.

```
function resultado=diagnosticoSPR(edad,altura,genero)
%UNTITLED3 Summary of this function goes here
%   Detailed explanation goes here
M=  [2.24, 2.40, 2.57, 2.73, 2.90, 3.06, 3.23, 3.39, 3.56, 3.72, 3.89;
     2.42, 2.59, 2.75, 2.92, 3.08, 3.25, 3.41, 3.58, 3.74, 3.91, 4.07;
     2.60, 2.77, 2.93, 3.10, 3.26, 3.43, 3.59, 3.76, 3.92, 4.09, 4.25;
     2.79, 2.95, 3.12, 3.28, 3.45, 3.61, 3.78, 3.94, 4.11, 4.27, 4.44;
     2.97, 3.14, 3.30, 3.47, 3.63, 3.80, 3.96, 4.13, 4.29, 4.46, 4.62;
     3.15, 3.34, 3.52, 3.71, 3.89, 4.08, 4.26, 4.45, 4.63, 4.82, 5.00;
     3.04, 3.23, 3.41, 3.60, 3.78, 3.97, 4.15, 4.34, 4.52, 4.71, 4.89;
     2.93, 3.12, 3.30, 3.49, 3.67, 3.86, 4.04, 4.23, 4.41, 4.60, 4.78;
     2.71, 2.90, 3.08, 3.27, 3.45, 3.64, 3.82, 4.01, 4.19, 4.38, 4.56;
     2.49, 2.68, 2.86, 3.05, 3.23, 3.42, 3.60, 3.79, 3.97, 4.16, 4.34;
     2.27, 2.46, 2.64, 2.83, 3.01, 3.20, 3.38, 3.57, 3.75, 3.94, 4.12;
     2.05, 2.24, 2.42, 2.61, 2.79, 2.98, 3.16, 3.35, 3.53, 3.72, 3.90;
     1.83, 2.02, 2.20, 2.39, 2.57, 2.76, 2.94, 3.13, 3.31, 3.50, 3.68];


 H= [2.52, 2.77, 3.02, 3.27, 3.52, 3.77, 4.02, 4.27, 4.52, 4.77, 5.02;
     2.68, 2.93, 3.18, 3.43, 3.68, 3.93, 4.18, 4.43, 4.68, 4.93, 5.18;
     2.83, 3.08, 3.33, 3.58, 3.83, 4.08, 4.33, 4.58, 4.83, 5.08, 5.33;
     2.99, 3.24, 3.49, 3.74, 3.99, 4.24, 4.49, 4.74, 4.99, 5.24, 5.49;
     3.15, 3.40, 3.65, 3.90, 4.15, 4.40, 4.65, 4.90, 5.15, 5.40, 5.65;
     3.30, 3.55, 3.80, 4.05, 4.30, 4.55, 4.80, 5.05, 5.30, 5.55, 5.80;
     3.24, 3.57, 3.89, 4.22, 4.54, 4.87, 5.19, 5.52, 5.84, 6.17, 6.49;
     3.10, 3.42, 3.75, 4.07, 4.40, 4.72, 5.05, 5.37, 5.70, 6.02, 6.35;
     2.81, 3.13, 3.46, 3.78, 4.11, 4.43, 4.76, 5.08, 5.41, 5.73, 6.06;
     2.52, 2.84, 3.17, 3.49, 3.82, 4.14, 4.47, 4.79, 5.12, 5.44, 5.77;
     2.23, 2.55, 2.88, 3.20, 3.53, 3.85, 4.18, 4.50, 4.83, 5.15, 5.48;
     1.94, 2.26, 2.59, 2.91, 3.24, 3.56, 3.89, 4.21, 4.54, 4.86, 5.19;
     1.65, 1.97, 2.30, 2.62, 2.95, 3.27, 3.60, 3.92, 4.25, 4.57, 4.90];


if (edad<11)
       i=1;
elseif (edad<13)
       i=2;
elseif (edad<15)
       i=3;
elseif (edad<17)
       i=4;
elseif (edad<19)
       i=5;
elseif (edad<23)
       i=6;
elseif (edad<28)
       i=7;
elseif (edad<35)
       i=8;
```

```
elseif (edad<45)
       i=9;
elseif (edad<55)
       i=10;
elseif (edad<65)
       i=11;
elseif (edad<75)
       i=12;
else
       i=13;
end



if (altura<148)
       j=1;
elseif (altura<153)
       j=2;
elseif (altura<158)
       j=3;
elseif (altura<163)
       j=4;
elseif (altura<168)
       j=5;
elseif (altura<173)
       j=6;
elseif (altura<178)
       j=7;
elseif (altura<183)
       j=8;
elseif (altura<188)
       j=9;
elseif (altura<193)
       j=10;
else
       j=11;
end

if (genero=='M')
    resultado=M(i,j);
else
    resultado=H(i,j);
end


end
```

6 Electrocardiógrafo

En este capítulo se detallará el funcionamiento del electrocardiógrafo, así como su puesta en marcha en la interfaz gráfica del usuario.

6.1 Qué es

El electrocardiógrafo es un aparato médico que se utiliza para captar la actividad eléctrica del corazón mediante electrodos, y, por tanto, se utiliza para detectar enfermedades cardíacas.

El que se va a utilizar para este proyecto es el que se puede ver en la Figura 6.1.

Figura 6.1 Electrocardiógrafo. Imagen disponible en: https://www.cooking-hacks.com/mysignals-hw-ehealth-medical-biometric-iot-platform-arduino-tutorial/ [Consultado el 24-08-2019].

6.2 Funcionamiento

El funcionamiento del electrocardiógrafo consiste en captar la actividad eléctrica del corazón a través de unos electrodos, por lo que para entender su funcionamiento se va a explicar a continuación los fundamentos por los cuales se produce el movimiento del corazón, generado a través de microcorrientes eléctricas.

6.2.1 El sistema de conducción

Es el tejido gracias al cual se inician y se transmiten los impulsos eléctricos del corazón. Dicho impulso se genera en el nodo sinusal, en la región superior de la aurícula derecha, y desde ahí viaja hasta el nódulo aurículoventricular (nódulo AV), ubicado entre las aurículas y los ventrículos. Durante el paso por el nódulo AV, la onda de activación eléctrica se pausa durante una décima de segundo aproximadamente, permitiendo que las aurículas se contraigan y vacíen su contenido de sangre en los ventrículos antes de producirse la propia contracción ventricular.

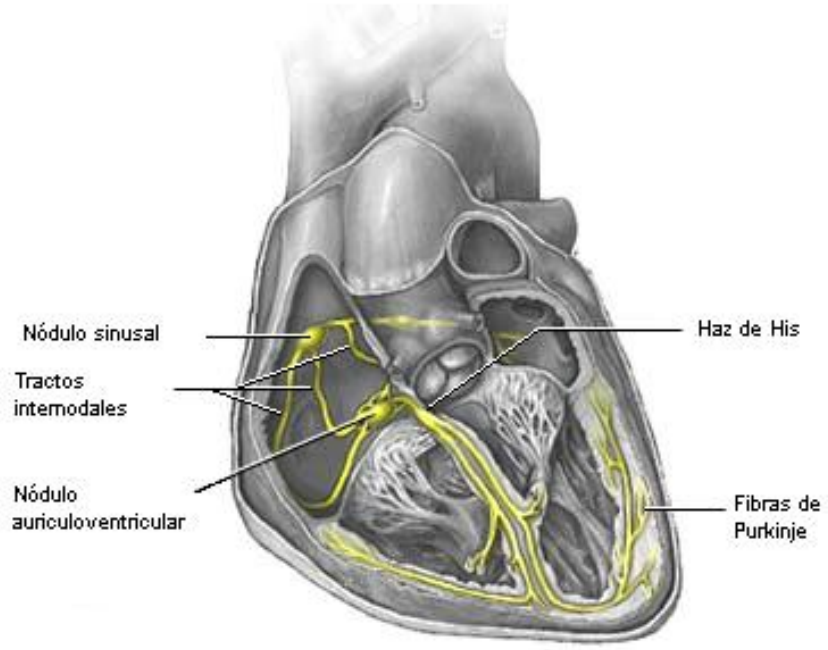

Figura 6.2 Sistema de conducción del corazón. Imagen disponible en: *http://www.electrocardiografia.es/a_electrica.html* [Consultado el 28-08-2019].

Después de atravesar el nódulo AV, el impulso cardíaco se propaga por el haz de His y sus ramas izquierda y derecha, finalmente distribuyéndose por las paredes de los ventrículos mediante las fibras de Purkinje, contrayéndose entonces ambos ventrículos, produciéndose la expulsión de sangre hacia los vasos sanguíneos del cuerpo.

6.3 ECG normal

En un ECG normal se puede distinguir, realizando la lectura de izquierda a derecha, la onda P, el segmento P-R, el complejo QRS, el segmento SR y la onda T.

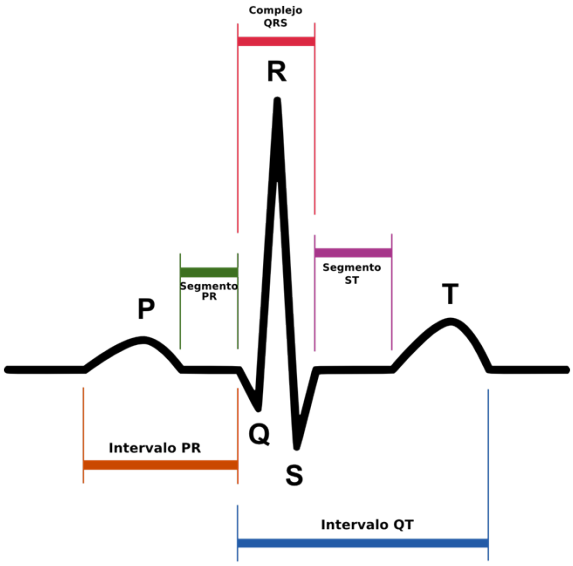

Figura 6.3 Ondas ECG. Imagen disponible en: *http://www.ub.edu/LabFisio/index.php?option=com_content&view=article&id=29&Itemid=148* [Consultado el 29-08-2019].

- La **onda P** representa el momento en el que las aurículas se están contrayendo y enviando sangre hacia los ventrículos. (Despolarización auricular)
- Durante el **segmento P-R**, las aurículas terminan de vaciarse y se produce una desaceleración en la transmisión de la corriente eléctrica, antes de contraerse los ventrículos. (El impulso viaja por el nodo AV y se propaga por el haz de His).
- Durante el **complejo QRS**, los ventrículos se contraen y expulsan la sangre. (Despolarización ventricular)
- La **onda T** representa el momento en el que el corazón se encuentra en un período de relajación una vez que se ha expulsado la sangre de los ventrículos. (Repolarización ventricular).

6.4 Interfaz gráfica

La interfaz gráfica que se ha implementado permite al usuario utilizar el electrocardiógrafo de una manera muy sencilla. En este caso, al igual que con el espirómetro, también se almacena la prueba en el archivo del usuario y sólo se puede ver el electrocardiograma del día seleccionado.

A continuación se muestran varias imágenes con el uso de la interfaz gráfica referente al ECG.

Al iniciar la interfaz gráfica del usuario, si no se conecta el aparato, aparece lo que se puede observar en la Figura 6.4. En este caso, habría que conectarlo al ordenador y volver a iniciar la interfaz.

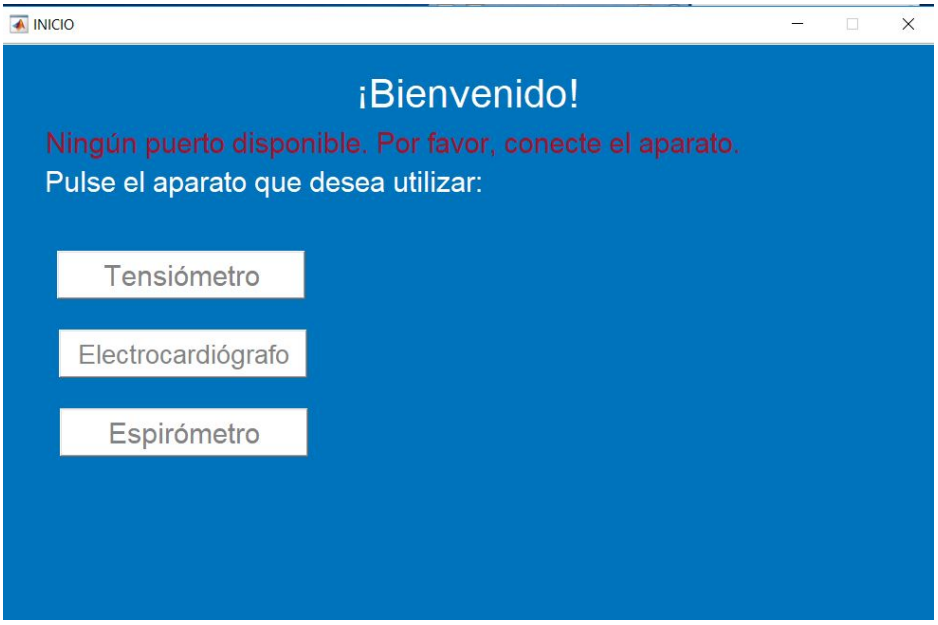

Figura 6.4 GUI Inicio.

Una vez conectado el aparato, Figura 6.5, si todo está correcto aparece el puerto disponible en la lista de arriba a la izquierda, y una vez seleccionado, ya es posible acceder a la interfaz del aparato deseado.

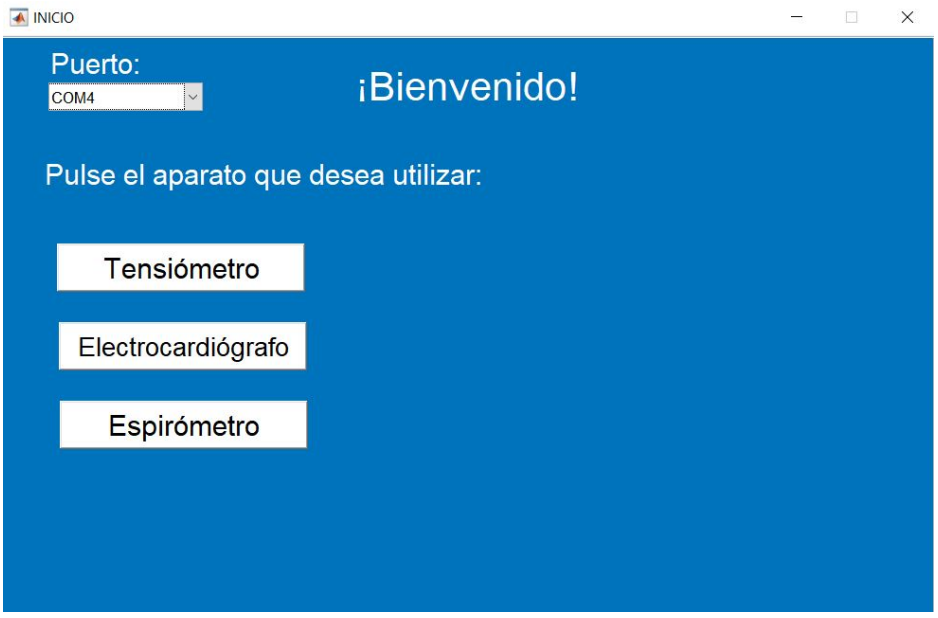

Figura 6.5 GUI Inicio.

Para el caso del ECG, en primer lugar aparece lo que se puede observar en la Figura 6.6. Una vez introducido el usuario (escribir el nombre y darle a Intro) y conectado, hay tres posibilidades según lo que quiera el usuario: Realizar una prueba, calibrar el aparato , o ver el electrocardiograma del día seleccionado. Cabe destacar que la calibración consiste en realizar una prueba sin guardar el resultado en el archivo de usuario para calibrar el aparato, lo que es recomendable antes de realizar el primer electrocardiograma.

Figura 6.6 GUI ECG.

Supongamos que se quiere realizar un electrocardiograma. El usuario debe tumbarse y colocarse los electrodos tal y como aparece en la Figura 6.7. Posteriormente, debe darle al botón de **Realizar ECG** y permanecer tumbado, quieto y callado durante unos minutos hasta la finalización de la prueba.

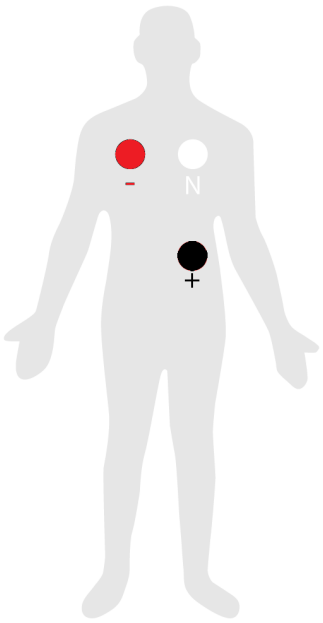

Figura 6.7 GUI ECG.

Si se quiere ver el resultado de la prueba, se le da al botón de **Diagnóstico** que aparece en la Figura 6.8.

Figura 6.8 GUI ECG.

En la Figura 6.9 se puede ver el diagnóstico de la prueba realizada. Como se puede observar, el resultado de la prueba es normal, pues no se ve ninguna anomalía.

Figura 6.9 GUI ECG.

Otro caso como ejemplo es el de la Figura 6.10, en el que se presenta una taquicardia.

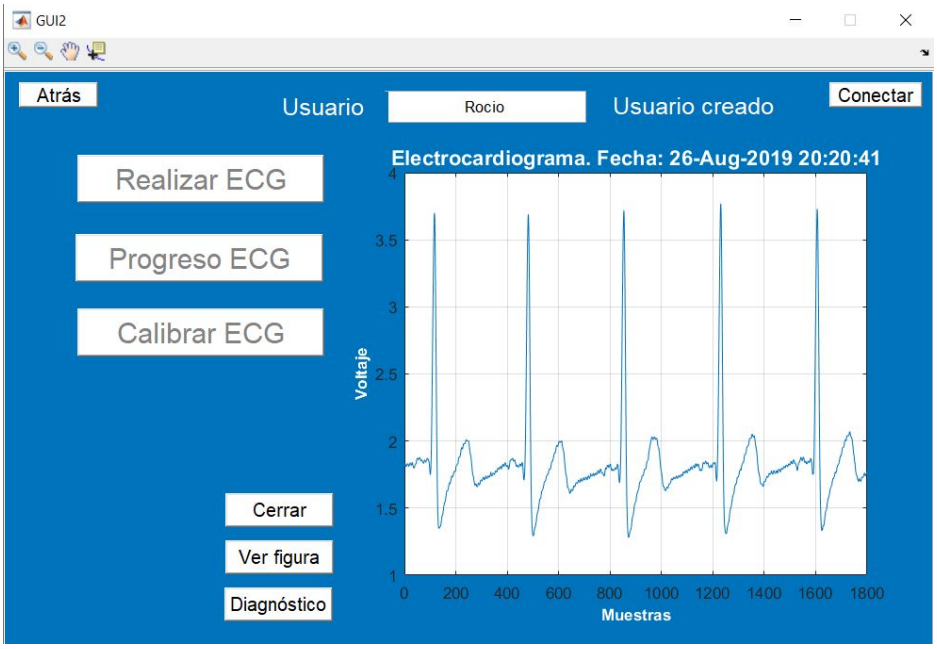

Figura 6.10 GUI ECG.

Figura 6.11 GUI ECG.

6.5 Diagnóstico

Para el diagnóstico de la interfaz gráfica se ha realizado una estadística a partir de 6 pacientes, a partir de la cual se ha llegado a la siguiente conclusión:

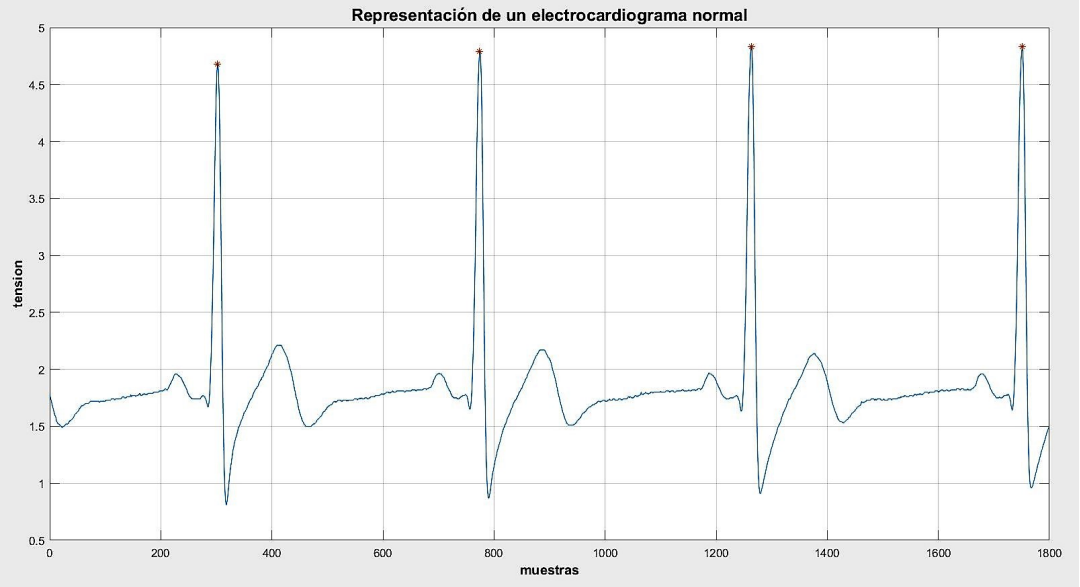

Figura 6.12 ECG normal.

- **Arritmia-Bradicardia:** Si un intervalo R-R dura más de 500 muestras.
- **Arritmia-Taquicardia:** Si un intervalo R-R dura menos de 400 muestras.
- **Arritmia-Ritmo irregular:** Si un intervalo R-R difiere de la media de muestras de todos los intervalos R-R en más de 40 muestras.

Así, en la Figura 6.13 se puede observar a simple vista que el usuario tiene taquicardia, pues en la figura anterior se presentan 4 complejos QRS, mientras que en este se presentan 5. Este es el mismo ejemplo que se puso en la Figura 6.10, cuyo diagnóstico resultó ser una taquicardia.

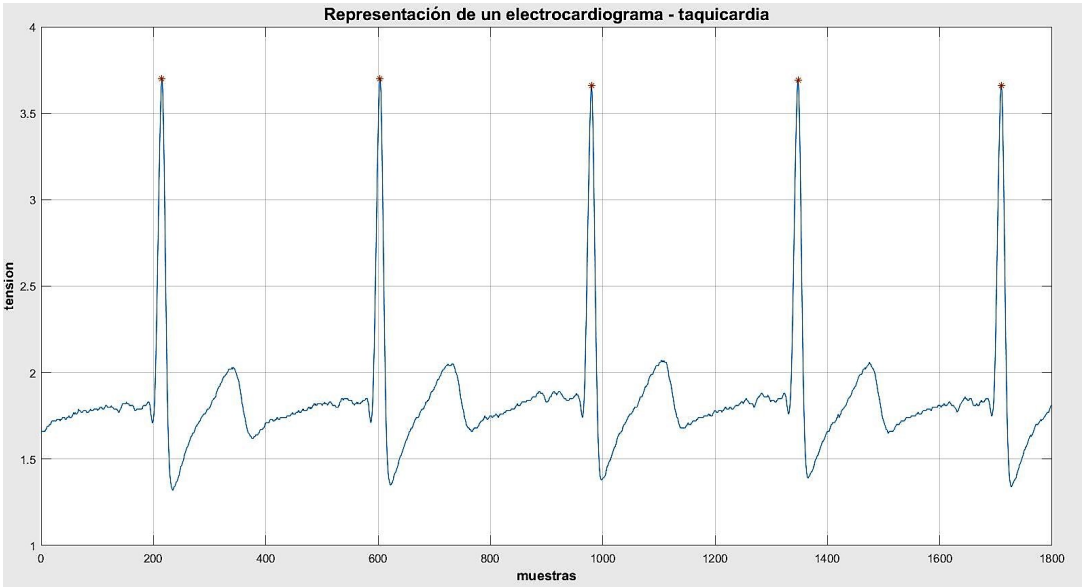

Figura 6.13 ECG taquicardia.

El código en el que se ha implementado el diagnóstico en Matlab es el siguiente:

Código 6.1 Programación diagnóstico ECG.

```
    load(filename,'progresoDato');

    %Busca las pruebas del tensiómetro
    pruebas = getByName(progresoDato, 'ECG');

    %Representa lo indicado en la GUI
    ECG = pruebas{c,1};


%PRUEBA R


%Para obtener los valores con los que comparar, se ha hecho la media
%de varias pruebas

[x1,x2] = size(ECG);
x = [0:(x1-1)];
maximos=[];
basal=mean(ECG);
vector=ECG-basal;
R=[];
aux1=0;
aux2=0;
a=0;

for k=7:length(vector)-6
    if vector(k)>1 %Para R
        %Se comprueba si es mayor o igual en un entorno de 6 muestras. De esta
            forma se busca el máximo
        if (vector(k)>vector(k-1) || vector(k)==vector(k-1)) && (vector(k)>
            vector(k+1) || vector(k)==vector(k+1)) && (vector(k)>vector(k+2) ||
            vector(k)==vector(k+2)) && (vector(k)>vector(k-2) || vector(k)==
            vector(k-2)) && (vector(k)>vector(k+3) || vector(k)==vector(k+3)) &&
             (vector(k)>vector(k-3) || vector(k)==vector(k-3)) && (vector(k)>
            vector(k+4) || vector(k)==vector(k+4)) && (vector(k)>vector(k-4) ||
            vector(k)==vector(k-4)) && (vector(k)>vector(k+5) || vector(k)==
            vector(k+5)) && (vector(k)>vector(k-5) || vector(k)==vector(k-5)) &&
             (vector(k)>vector(k+6) || vector(k)==vector(k+6)) && (vector(k)>
            vector(k-6) || vector(k)==vector(k-6))
           if vector(k)>aux1
               aux1=vector(k);
               aux2=k;
           end
        end
    end
    a=a+1;
    if(a==50) %Se comprueban máximos en un entorno de 50 muestras
        if(aux1~=0)
            maximos(end+1)=aux1;
        end
        if(aux2~=0)
```

```
            R=[R;aux2];
        end
        a=0;
        aux1=0;
        aux2=0;
    end
end

%plot(x,ECG,x(R),ECG(R),'*');



%-------------DIAGNOSTICO---------------%

% [p1,p2]=size(P);
[r1,r2]=size(R);
resta=[];
% restap=[];

for i=2:r1
 rr=R(i)-R(i-1);
 resta=[resta;rr];
end

% for i=2:p1
%  pr=P(i)-P(i-1)
%  restap=[restap;pr]
% end



for i=1:size(resta)
    if (abs(mean(resta)-resta(i))>40)
        diagnostico='Arritmia: Ritmo irregular';
    elseif (resta(i)>500)
        diagnostico='Arritmia: Bradicardia';
    elseif (resta(i)<400)
        diagnostico='Arritmia: Taquicardia';
    else
        diagnostico='ECG normal';
    end
end
```

7 Impresión 3D

Para proteger el escudo de MySignals y la placa de Arduino, se ha impreso una caja en 3D. A continuación se explicará el procedimiento.

7.1 Diseño

Se ha utilizado SolidWorks 2014 para el diseño de la caja protectora.

El plano referente a la parte inferior de la caja se puede ver en la Figura 7.1.

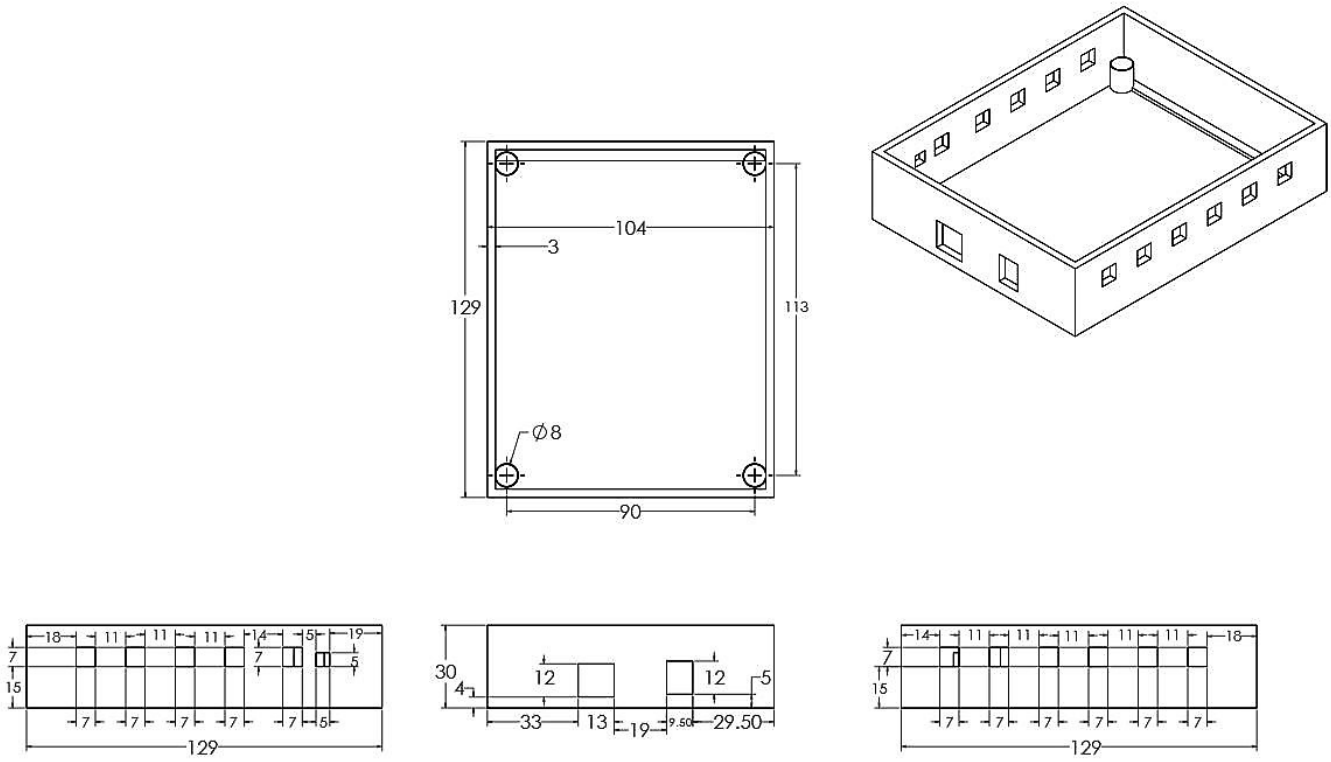

Figura 7.1 Planos de la caja protectora.

Y para la tapadera de la caja:

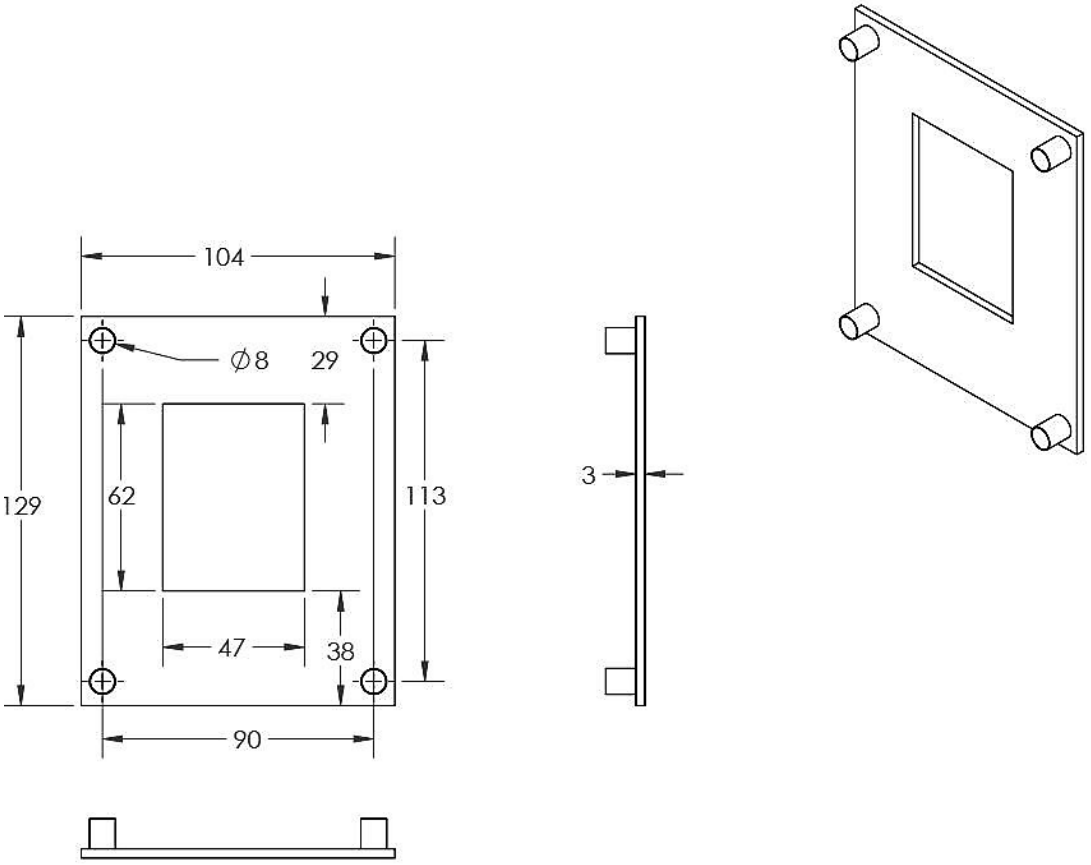

Figura 7.2 Planos de la tapa de la caja protectora.

7.2 Vista 3D de la caja protectora

Quedando la vista 3D de la siguiente forma:

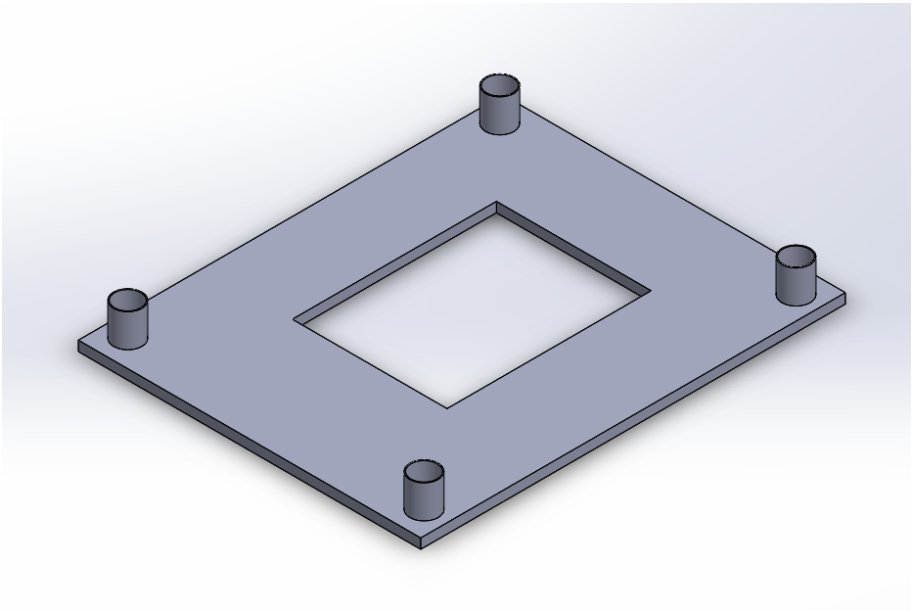

Figura 7.3 Tapadera de la caja protectora. Vista 3D..

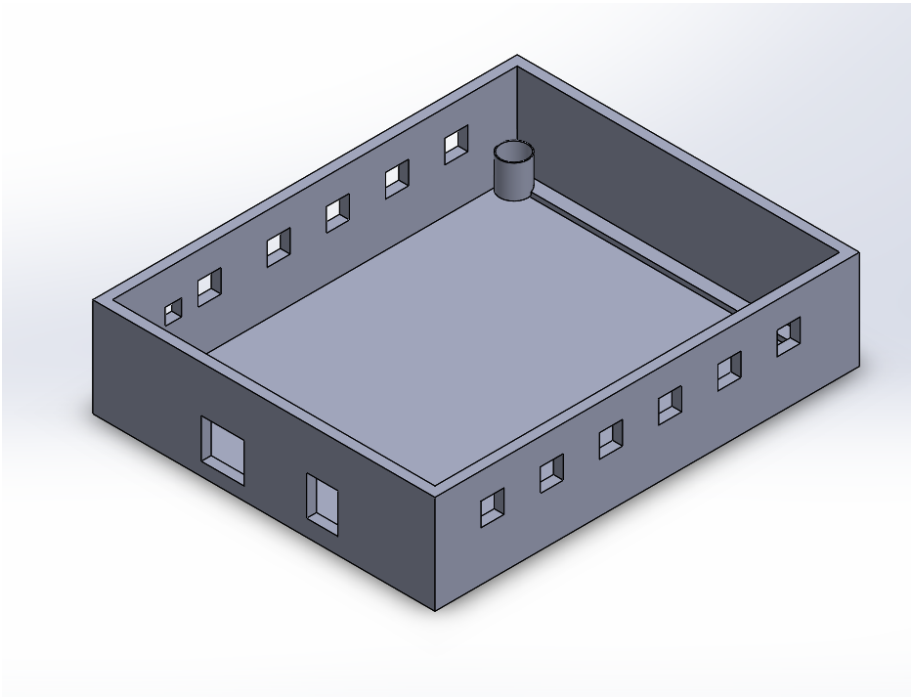

Figura 7.4 Caja protectora. Vista 3D..

8 Conclusiones líneas futuras

Finalmente, en este capítulo se expondrán las conclusiones de todo el trabajo realizado, así como las líneas futuras y propuestas de mejora.

8.1 Conclusiones

El objetivo que se pretendía alcanzar, que es en primer lugar hacer funcionar los dispositivos, y posteriormente crear la interfaz gráfica con Matlab, se ha cumplido satisfactoriamente.

Además, la inclusión del diagnóstico en la interfaz gráfica no se contemplaba al principio, pero podría captar más la atención del usuario, pues por ejemplo hoy en día todo el mundo tiene un tensiómetro en casa, pero hay quienes no saben interpretar el resultado.

Como conclusión, el resultado de este proyecto ha sido el esperado. Y personalmente, me siento bastante feliz porque he contribuído en el avance de la bioingeniería, ya que en un futuro bastante cercano todos podremos detectar enfermedades desde casa.

8.2 Líneas futuras

A pesar de haber conseguido el objetivo de este proyecto, pienso que se podría mejorar bastante:

- Diseñando otro tipo de escudo como Mysignals que permitiera realizar otro tipo de pruebas.
- Mejorando el diagnóstico del electrocardiograma incluyendo Machine Learning. Así, se podría obtener un diagnóstico bastante preciso teniendo en cuenta todas las posibles anomalías cardíacas.

Incluyendo mejoras en los diagnósticos y ampliando el número de pruebas que se pueden realizar en casa, se podrá llegar en un futuro a saber qué enfermedad tiene cada paciente incluso antes de ir al hospital. De esta forma el especialista ganaría bastante tiempo, pues sólo tendría que verificar que el diagnóstico es correcto para aplicar el tratamiento correspondiente.

Apéndice A
Código Arduino

```
#include <EEPROM.h>
#include <MySignals.h>
#include "Wire.h"
#include "SPI.h"

String text;
int ki;
int estado;
int k;
int velocidad;
int n;
int m;
int i;
int j;
int l;
int v[5];
int Conectado;
float ECG;
int medida;
int fev;
int fev1;
int fev2;
int fev3;
int fev4;
int fev5;

void setup()
{
  Serial.begin(9600);
  EEPROM.get(0,k);
//  EEPROM.get(1,fev);
  leer();
    if (ki==10){
     Serial.print("Hi");
     Serial.flush();
     Serial.end();
     Serial.begin(19200);
     k=1; // Se pone a 1 para leer a la velocidad del tensiómetro
     EEPROM.put(0,k);
  }
```

```
    if (ki==20){
     Serial.print("Hi");
     Serial.flush();
     Serial.end();
     Serial.begin(115200);
     k=2; // Se pone a 2 para leer a la velocidad del ECG
     EEPROM.put(0,k);
  }


     if (ki==30){
     Serial.print("Hi");
     Serial.flush();
     Serial.end();
     Serial.begin(9600);
     k=3; // Se pone a 3 para leer a la velocidad del espirómetro
     EEPROM.put(0,k);
  }

}



void loop(){

if (k==1)
{
  leerT();
}
else if (k==2)
{
  leerECG();
}
else if (k==3)
{
  leerSPR();
}


// Para el caso del tensiómetro

  if (ki==1){ //ki=1 sería para comprobar el estado del tensiómetro (Como un
      MUX)
    escribirT();}
 // Serial.print("Hi");}

  if (ki==11){ //ki=11 sería para comprobar si queremos leer o no.
     Serial.print("Hi");
  }


   if (ki==111){ //ki=111 para comprobar si queremos medir la tensión.
     Serial.print("Hi");
     Serial.flush();//limpiamos el serial
     medirT();
  }
```

```
// Para el caso del ECG


  if (ki==2){
    //Serial.print("Hi");
    ECGConectado();
    k=2;
  }


  if (ki==22){ //ki=22 sería para comprobar si queremos leer o no (ECG).
     Serial.print("Hi");
  }


   if (ki==222){ //ki=222 para comprobar si queremos realizar ECG.
     Serial.print("Hi");
     Serial.flush();//limpiamos el serial
     RealizarECG();
  }


// Para el caso del espirómetro

  if (ki==3){
  //  Serial.print("Hi");
    SPRConectado();
    k=3;
  }

   if (ki==33){
   //  Serial.print("Hi");
     Serial.flush();//limpiamos el serial
     ComprobarSPR();
  }

   if (ki==333){ //ki=333 para comprobar si queremos realizar espirometría
     Serial.print("Hi");
     Serial.flush();//limpiamos el serial
     RealizarSPR();
  }

   if (ki==3333){ //ki=3333 para comprobar si queremos escribir SPR
     Serial.flush();//limpiamos el serial
     EscribirSPR();
  }

  if (ki==3330){ //Para comprobar si queremos borrar las medidas del SPR
   MySignals.initSensorUART();
   MySignals.enableSensorUART(SPIROMETER);

   while (MySignals.getStatusSpiro() == 0)
   {
     delay(100);
```

```
   }
   MySignals.deleteSpiroMeasures();
   Serial.flush();
  }

}




void calibrar()
{

    MySignals.begin();
    float ECG = MySignals.getCalibratedECG(5,0,0,DATA);
  //  Serial.print("C");
    Serial.println(ECG);
    delay(1); // Esperar 1 milisegundo
}



void medirECG()
{
  float ECG = MySignals.getECG(VOLTAGE);
  Serial.println(ECG, 2); // 2 cifras decimales
  delay(1);
}


void leerT() {
    Serial.end();
    Serial.begin(19200);
    ki = 0;
    while(Serial.available()==0){}
    if (Serial.available()){
      text = Serial.readStringUntil('\n');
      ki = text.toInt();}
    Serial.flush();//limpiamos el serial.
}



void leerECG() {
    ki = 0;
    Serial.end();
    Serial.begin(115200);
    while(Serial.available()==0){}
    if (Serial.available()){
      text = Serial.readStringUntil('\n');
      ki = text.toInt();}
    Serial.flush();//limpiamos el serial.
}


void leerSPR() {
```

```
    ki = 0;
    Serial.end();
    Serial.begin(9600);
    while(Serial.available()==0){}
    if (Serial.available()){
      text = Serial.readStringUntil('\n');
      ki = text.toInt();}
    Serial.flush();//limpiamos el serial.
}



void leer() {
    ki = 0;
    while(Serial.available()==0){}
    if (Serial.available()){
      text = Serial.readStringUntil('\n');
      ki = text.toInt();}
    Serial.flush();//limpiamos el serial.
}



void ECGConectado(){

  // Si NO está conectado el ECG al escudo, al medir da valores menores a 3 V;
      y si está conectado, se ven valores mayores a este.
  // (COMPROBAR CON LOS PARCHES CONECTADOS SIN PONER EN EL PACIENTE)

  n=0;
  Conectado=0;
  Serial.end();
  Serial.begin(115200);
  MySignals.begin();
  while (n<1000)
  {
    ECG = MySignals.getECG(VOLTAGE);
   // Serial.println(ECG);
    delay(1); // Esperar 1 milisegundo
    n++;
    if (ECG>3 and ECG<4)
    {
      Conectado = 1;
      n=1000;
    }
  }

if (Conectado==1)
{
  Serial.print("Y");
}
else
{
  Serial.print("N");
}
}
```

```
void RealizarECG(){

  while(n<10000)
  {
    calibrar();
    n++;
  }

  n=0;

Serial.print("M");

  while(n<1800)
  {
    medirECG();
    n++;
  }

  n=0;

    Serial.print("X");
}




void escribirT(){
    MySignals.begin();
    MySignals.initSensorUART();
    MySignals.enableSensorUART(BLOODPRESSURE);
    estado = MySignals.getStatusBP();
    Serial.print(estado);
    delay(1000);
    Serial.flush();//limpiamos el serial.
    MySignals.disableSensorUART();
}




void medirT()
{
  MySignals.begin();

  MySignals.initSensorUART();
  MySignals.enableSensorUART(BLOODPRESSURE);

  tft.init();
  tft.setRotation(2);
  tft.fillScreen(ILI9341_BLACK);
  tft.setTextColor(ILI9341_WHITE, ILI9341_BLACK);


  tft.drawString("Blood pressure:", 0, 0, 2);
```

```
  while(MySignals.getStatusBP()==0){}

    if (MySignals.getStatusBP())
  {
    delay(1000);

    if (MySignals.getBloodPressure() == 1)
    {
      MySignals.disableMuxUART();
      //Serial.println();
      //Serial.print("Diastolic: ");
      Serial.println(MySignals.bloodPressureData.diastolic);
     // Serial.print("Systolic: ");
      Serial.println(MySignals.bloodPressureData.systolic);
    //  Serial.print("Pulse/min: ");
      Serial.println(MySignals.bloodPressureData.pulse);

    // Tensión y pulso por pantalla:
      tft.drawString("Diastolic:", 0, 15, 2);
      tft.drawNumber(MySignals.bloodPressureData.diastolic, 100, 15, 2);

      tft.drawString("Systolic:", 0, 30, 2);
      tft.drawNumber(MySignals.bloodPressureData.systolic, 100, 30, 2);

      tft.drawString("Pulse/min:", 0, 45, 2);
      tft.drawNumber(MySignals.bloodPressureData.pulse, 100, 45, 2);


      MySignals.enableMuxUART();}
  }
      Serial.flush();//limpiamos el serial.
      MySignals.disableSensorUART();
}




void SPRConectado(){

  MySignals.initSensorUART();
  MySignals.enableSensorUART(SPIROMETER);
  n=0;
  Conectado=0;
  Serial.end();
  Serial.begin(9600);
  MySignals.begin();
  while (n<10)
  {
    if (MySignals.getStatusSpiro() == 1)
    {
      Conectado=1;
      n=10;
    }
    n++;
    delay(100);
  }
```

```
if (Conectado==1)
{
  Serial.print("Y");
}
else
{
  Serial.print("N");
}

Serial.flush();//limpiamos el serial.
//MySignals.disableSensorUART();
}




void RealizarSPR(){
  n=0;
  m=0;
  medida=0;
  i=sizeof(int)+1;
  MySignals.begin();
//  MySignals.initSensorUART();
//  MySignals.enableSensorUART(SPIROMETER);
//  delay(1000);
//
//  MySignals.getSpirometer();
//  MySignals.disableMuxUART();
//  delay(1000);

MySignals.initSensorUART();
MySignals.enableSensorUART(SPIROMETER);
while(n<20)
{

  MySignals.enableMuxUART();
  delay(1000);

if (MySignals.getStatusSpiro() == 1)
 {

  MySignals.getSpirometer();
  MySignals.disableMuxUART();
  delay(1000);
//  Serial.print(MySignals.spir_measures);
 if (MySignals.spir_measures == 1)
  {
    medida=1;
    EEPROM.put(15,medida);
    fev=MySignals.spirometerData[0].spir_fev;
    delay(500);
    fev2=MySignals.spirometerData[0].spir_fev;
    delay(200);
    fev3=MySignals.spirometerData[0].spir_fev;
    delay(200);
    if ((fev==fev2)&&(fev2==fev3))
```

```
    {
      if ((fev>100) && (fev<600))
      {
        EEPROM.put(i,fev);
//        Serial.println(i);
//        Serial.println(fev);
        i+= sizeof(int);
        m++;
      }
    }
  }
  if(m==5)
  {
    n=20;
  }
  n++;
}
//delay(1000);
EEPROM.put(15,medida);

}

MySignals.enableMuxUART();
MySignals.disableSensorUART();
delay(1000);

}


void ComprobarSPR(){
  Serial.flush();//limpiamos el serial.
  EEPROM.get(15,medida);
  delay(500);
  if (medida == 1)
  {
//    Serial.print(MySignals.spirometerData[0].spir_pef);
//    Serial.print(F("L/min "));
    Serial.print("NUM"); // Numero
  }


  else
  {
    Serial.print("EMP"); // Empty
  }
  Serial.flush();//limpiamos el serial.

}

void EscribirSPR(){
  Serial.begin(9600);
  Serial.flush();
  delay(500);
  EEPROM.get(3,fev1);
  delay(500);
  EEPROM.get(5,fev2);
  delay(500);
```

```
  EEPROM.get(7,fev3);
  delay(500);
  EEPROM.get(9,fev4);
  delay(500);
  EEPROM.get(11,fev5);
  delay(500);


  v[1]=fev1;
  v[2]=fev2;
  v[3]=fev3;
  v[4]=fev4;
  v[5]=fev5;

  // Se mira de los 5 valores cual está repetido 3 veces (más repetido)
  for(l=1;l<6;l++)
  {

    i=0;
    for (j=1;j<6;j++)
    {
      if (v[l]==v[j]){
      i++;}
      if (i==3)
      {
      fev=v[l];
      l=6;
      j=6;
      }
    }
  }


  Serial.print(fev);
  delay(100);
  Serial.print(fev);
  delay(100);
  Serial.print(fev);
}
```

Apéndice B
Código Matlab - GUI tensiómetro

* Para que funcione la interfaz son necesarios los archivos .fig donde se configura la parte gráfica.

```
function varargout = GUI1(varargin)
% GUI1 MATLAB code for GUI1.fig
%      GUI1, by itself, creates a new GUI1 or raises the existing
%      singleton*.
%
%      H = GUI1 returns the handle to a new GUI1 or the handle to
%      the existing singleton*.
%
%      GUI1('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in GUI1.M with the given input arguments.
%
%      GUI1('Property','Value',...) creates a new GUI1 or raises the
%      existing singleton*. Starting from the left, property value pairs are
%      applied to the GUI before GUI1_OpeningFcn gets called. An
%      unrecognized property name or invalid value makes property application
%      stop. All inputs are passed to GUI1_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help GUI1

% Last Modified by GUIDE v2.5 08-Apr-2019 17:59:13

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',    mfilename, ...
                   'gui_Singleton', gui_Singleton, ...
                   'gui_OpeningFcn', @GUI1_OpeningFcn, ...
                   'gui_OutputFcn', @GUI1_OutputFcn, ...
                   'gui_LayoutFcn', [] , ...
                   'gui_Callback', []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
```

```
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before GUI1 is made visible.
function GUI1_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject   handle to figure
% eventdata reserved - to be defined in a future version of MATLAB
% handles   structure with handles and user data (see GUIDATA)
% varargin  command line arguments to GUI1 (see VARARGIN)

% Choose default command line output for GUI1
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);



% UIWAIT makes GUI1 wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = GUI1_OutputFcn(hObject, eventdata, handles)
% varargout cell array for returning output args (see VARARGOUT);
% hObject   handle to figure
% eventdata reserved - to be defined in a future version of MATLAB
% handles   structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject   handle to pushbutton1 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles   structure with handles and user data (see GUIDATA)
nombre=get(handles.edit1,'string');
filename=strcat(nombre,'.mat');
set(hObject,'Enable','off')
set(handles.pushbutton2,'Enable','off')
set(handles.pushbutton5,'Enable','off')
pause(1);
set(handles.text7,'String','Iniciando...')
pause(1);
kp = strcat('11',10);
global puerto;
puertoserie(puerto,19200,kp);
set(handles.text7,'String','Midiendo...')
pause(1);
```

```
dato = MedirTensionGUI(filename,puerto);
valores=([’’,10,’Presión sistólica: ’,num2str(dato{1}.systolic),10, ’Presión
    diastólica: ’ ,num2str(dato{1}.diastolic),10, ’Pulso: ’,num2str(dato{1}.
    pulse)]);
set(handles.text4,’String’,valores);
set(handles.text4,’Visible’,’On’);
set(handles.pushbutton4,’Visible’,’On’);
set(handles.pushbutton11,’Visible’,’On’);
set(handles.text7,’String’,’Conectado’);
set(handles.pushbutton5,’Enable’,’on’);



% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject   handle to pushbutton2 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles   structure with handles and user data (see GUIDATA)

nombre = get(handles.edit1,’string’);
filename = strcat(nombre,’.mat’);
[v, pruebas] = ContadorFechas(filename);
[n,x]=size(v);
fechaInicial = pruebas{1,2};
t0 = pruebas{1,2}; %La fecha inicial
t  = pruebas{2,2}; %Segunda fecha intervalo
str = ’’;
if n>1
    for i=1:n
        t = datestr(addtodate(datenum(fechaInicial), v(i)*7, ’day’));
        t0 = datestr(addtodate(datenum(t), -7, ’day’));
        cad = strcat(datestr(t0),32,45,32,datestr(t));
        str = strvcat(str,cad);
    end
else
    cad = strcat(datestr(t0),32,45,32,datestr(t));
    str = strvcat(str,cad);
end

set(handles.popupmenu1,’String’,str);
set(handles.text5,’Visible’,’on’);
set(handles.popupmenu1,’Visible’,’on’);




function edit1_Callback(hObject, eventdata, handles)
% hObject   handle to edit1 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles   structure with handles and user data (see GUIDATA)

% Hints: get(hObject,’String’) returns contents of edit1 as text
%        str2double(get(hObject,’String’)) returns contents of edit1 as a
    double
nombre=get(hObject,’string’);
%Creamos el fichero si no existe
```

```
filename=strcat(nombre,'.mat');
existe = crearfichero(filename);
set(handles.text3,'String','Usuario creado');
set(handles.pushbutton9,'Visible','on');
set(handles.pushbutton9,'Enable','On');
set(handles.pushbutton4,'Visible','off');
set(handles.pushbutton11,'Visible','off');
set(handles.pushbutton6,'Visible','off');
set(handles.pushbutton7,'Visible','off');
cla reset; %Para cerrar la gráfica (quita la leyenda y los datos)
set(handles.axes3,'Visible','off');
set(handles.text4,'Visible','off');
set(handles.text5,'Visible','off');
set(handles.text6,'Visible','off');
set(handles.text7,'Visible','off');
set(handles.text8,'Visible','Off');
set(handles.popupmenu1,'Visible','off');


if existe == 1
    set(handles.pushbutton2,'Enable','on');
else
    set(handles.pushbutton2,'Enable','off');
end


% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject   handle to edit1 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles   empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject   handle to pushbutton4 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles   structure with handles and user data (see GUIDATA)

set(handles.text4,'String','');
set(handles.text4,'Visible','Off');
set(handles.text8,'String','');
set(handles.text8,'Visible','Off');
set(handles.pushbutton4,'Visible','Off');
set(handles.pushbutton11,'Visible','Off');
nombre = get(handles.edit1,'string');
filename=strcat(nombre,'.mat');
existe = crearfichero(filename);
load(filename,'progresoDato');
```

```
if existe == 1
    set(handles.pushbutton2,'Enable','on');
else
    set(handles.pushbutton2,'Enable','off');
end


% --- Executes on selection change in popupmenu1.
function popupmenu1_Callback(hObject, eventdata, handles)
% hObject   handle to popupmenu1 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles   structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns popupmenu1 contents
    as cell array
%       contents{get(hObject,'Value')} returns selected item from popupmenu1
c = get(hObject,'Value');
nombre = get(handles.edit1,'string');
filename = strcat(nombre,'.mat');
axes=(handles.axes3);
ProgresoTensionGUI(c,filename);
set(handles.popupmenu1,'String','');
set(handles.text5,'Visible','off');
set(handles.popupmenu1,'Visible','off');
set(handles.pushbutton6,'Visible','on');
set(handles.pushbutton7,'Visible','on');
set(handles.pushbutton2,'Enable','off');



% --- Executes during object creation, after setting all properties.
function popupmenu1_CreateFcn(hObject, eventdata, handles)
% hObject   handle to popupmenu1 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles   empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in pushbutton5.
function pushbutton5_Callback(hObject, eventdata, handles)
% hObject   handle to pushbutton5 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles   structure with handles and user data (see GUIDATA)
close all;
INICIO;


% --- Executes on button press in pushbutton6.
function pushbutton6_Callback(hObject, eventdata, handles)
% hObject   handle to pushbutton6 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)
set(handles.pushbutton7,'Visible','off');
set(handles.pushbutton6,'Visible','off');
cla reset; %Para cerrar la gráfica (quita la leyenda y los datos)
set(handles.axes3,'Visible','off');
set(handles.pushbutton1,'Enable','off');
set(handles.pushbutton2,'Enable','off');

% --- Executes on button press in pushbutton7.
function pushbutton7_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton7 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

c = get(handles.popupmenu1,'Value');
nombre = get(handles.edit1,'string');
filename = strcat(nombre,'.mat');
figure(2);
ProgresoTensionGUI2(c,filename);


% --- Executes on button press in pushbutton9.
function pushbutton9_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton9 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set(handles.pushbutton9,'Enable','Off');
set(handles.pushbutton4,'Visible','off');
set(handles.pushbutton11,'Visible','Off');
set(handles.pushbutton6,'Visible','off');
set(handles.pushbutton7,'Visible','off');
cla reset; %Para cerrar la gráfica (quita la leyenda y los datos)
set(handles.axes3,'Visible','off');
set(handles.text4,'Visible','off');
set(handles.text5,'Visible','off');
set(handles.text6,'Visible','off');
set(handles.text7,'Visible','off');
set(handles.popupmenu1,'Visible','off');
pause(1);

% %La primera vez es para indicarle que se trata del tensiómetro para cambiar
% %la velocidad del puerto serie

global puerto;
kp = strcat('10',10)
a = puertoserie(puerto,9600,kp);

% pause(5);

kp = strcat('1',10)
a = puertoserie(puerto,19200,kp);
    if a=='a1'
        set(handles.text7,'String','Conectado');
        set(handles.text7,'ForegroundColor',[0 1 0]);
        set(handles.pushbutton1,'Enable','on');
    end
    if a=='a0'
```

```
        set(handles.text7,'String','Desconectado');
        set(handles.text7,'ForegroundColor',[0.64 0.08 0.18]);
        set(handles.pushbutton9,'Enable','on');
    end
set(handles.text6,'Visible','on');
set(handles.text7,'Visible','on');


% --- Executes on button press in pushbutton11.
function pushbutton11_Callback(hObject, eventdata, handles)
% hObject   handle to pushbutton11 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles   structure with handles and user data (see GUIDATA)

nombre=get(handles.edit1,'string'); %Nombre de usuario
filename=strcat(nombre,'.mat');
load(filename,'progresoDato'); %Carga el fichero del usuario
pruebas = getByName(progresoDato, 'tension'); %Busca las pruebas del tensió
    metro
systolic = pruebas{end,1}.systolic %Coge la tensión sistólica de la última
    prueba
diastolic = pruebas{end,1}.diastolic %Coge la tensión diastólica de la última
    prueba
pulse = pruebas{end,1}.pulse %Coge el pulso de la última prueba

if ((systolic<80)||(diastolic<60))
    diagnostico='Hipotensión';
elseif ((systolic<120)&&(diastolic<80))
    diagnostico='Tensión normal';
elseif ((systolic<140)||(diastolic<90))
    diagnostico='Prehipertension';
elseif ((systolic<160)||(diastolic<100))
    diagnostico='Hipertensión grado 1';
elseif ((systolic<180)||(diastolic<110))
    diagnostico='Hipertensión grado 2';
else
    diagnostico='Crisis hipertensiva (EMERGENCIA)';
end


if (pulse>100)
    diagnostico = strvcat(diagnostico,'Taquicardia');
elseif (pulse<50)
    diagnostico = strvcat(diagnostico,'Braquicardia');
else
    diagnostico = strvcat(diagnostico,'Pulso normal');
end

texto=strvcat('El resultado es',diagnostico)
set(handles.text8,'String',texto);
set(handles.text8,'Visible','On');
```

Apéndice C
Código Matlab - GUI espirómetro

* Para que funcione la interfaz son necesarios los archivos .fig donde se configura la parte gráfica.

```
function varargout = GUI3(varargin)
% GUI3 MATLAB code for GUI3.fig
%      GUI3, by itself, creates a new GUI3 or raises the existing
%      singleton*.
%
%      H = GUI3 returns the handle to a new GUI3 or the handle to
%      the existing singleton*.
%
%      GUI3('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in GUI3.M with the given input arguments.
%
%      GUI3('Property','Value',...) creates a new GUI3 or raises the
%      existing singleton*. Starting from the left, property value pairs are
%      applied to the GUI before GUI3_OpeningFcn gets called. An
%      unrecognized property name or invalid value makes property application
%      stop. All inputs are passed to GUI3_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help GUI3

% Last Modified by GUIDE v2.5 04-Apr-2019 22:12:54

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',    mfilename, ...
                   'gui_Singleton', gui_Singleton, ...
                   'gui_OpeningFcn', @GUI3_OpeningFcn, ...
                   'gui_OutputFcn', @GUI3_OutputFcn, ...
                   'gui_LayoutFcn', [] , ...
                   'gui_Callback', []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
```

```
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before GUI3 is made visible.
function GUI3_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject   handle to figure
% eventdata reserved - to be defined in a future version of MATLAB
% handles   structure with handles and user data (see GUIDATA)
% varargin  command line arguments to GUI3 (see VARARGIN)

% Choose default command line output for GUI3
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);



% UIWAIT makes GUI3 wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = GUI3_OutputFcn(hObject, eventdata, handles)
% varargout cell array for returning output args (see VARARGOUT);
% hObject   handle to figure
% eventdata reserved - to be defined in a future version of MATLAB
% handles   structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject   handle to pushbutton1 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles   structure with handles and user data (see GUIDATA)
nombre=get(handles.edit1,'string');
filename=strcat(nombre,'.mat');
set(hObject,'Enable','off')
set(handles.pushbutton2,'Enable','off')
set(handles.pushbutton17,'Enable','off')
set(handles.pushbutton5,'Enable','off')
pause(1);
set(handles.text7,'String','Leyendo...')
pause(1);

global puerto;
[dato,a]=SPRGUI(filename,puerto)
if (a=='NUM')
```

```
    valores=([''',10,'Fev1: ',num2str(dato{1}),' L',10]);
    set(handles.text4,'String',valores);
    set(handles.text4,'Visible','On');
    set(handles.pushbutton4,'Visible','On');
    set(handles.pushbutton11,'Visible','On');
    set(handles.text7,'String','Conectado');
    set(handles.pushbutton5,'Enable','on');
    set(handles.pushbutton17,'Enable','on');
else
    set(handles.text4,'String','No hay ningún dato disponible');
    set(handles.text4,'Visible','On');
    set(handles.pushbutton4,'Visible','On');
    set(handles.pushbutton11,'Visible','On');
    set(handles.text7,'String','Conectado');
    set(handles.pushbutton5,'Enable','on');
    set(handles.pushbutton17,'Enable','on');
end



% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject   handle to pushbutton2 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles   structure with handles and user data (see GUIDATA)

nombre = get(handles.edit1,'string');
filename = strcat(nombre,'.mat');
str = FechasSPR(filename);
set(handles.pushbutton1,'Enable','off');
set(handles.pushbutton2,'Enable','off');

set(handles.popupmenu1,'String',str);
set(handles.text5,'Visible','on');
set(handles.popupmenu1,'Visible','on');







function edit1_Callback(hObject, eventdata, handles)
% hObject   handle to edit1 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles   structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%       str2double(get(hObject,'String')) returns contents of edit1 as a
    double
nombre=get(hObject,'string');
%Creamos el fichero si no existe
filename=strcat(nombre,'.mat');
existe = crearfichero(filename);
set(handles.text3,'String','Usuario creado');
set(handles.pushbutton9,'Visible','on');
set(handles.pushbutton9,'Enable','On');
```

```
set(handles.pushbutton4,'Visible','off');
set(handles.pushbutton11,'Visible','off');
set(handles.pushbutton6,'Visible','off');
set(handles.pushbutton7,'Visible','off');
cla reset; %Para cerrar la gráfica (quita la leyenda y los datos)
set(handles.axes3,'Visible','off');
set(handles.text4,'Visible','off');
set(handles.text5,'Visible','off');
set(handles.text6,'Visible','off');
set(handles.text7,'Visible','off');
set(handles.popupmenu1,'Visible','off');


if existe == 1
    set(handles.pushbutton2,'Enable','on');
else
    set(handles.pushbutton2,'Enable','off');
end


% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject   handle to edit1 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles   empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject   handle to pushbutton4 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles   structure with handles and user data (see GUIDATA)

set(handles.text4,'String','');
set(handles.text4,'Visible','Off');
set(handles.text9,'Visible','Off');
set(handles.text10,'Visible','Off');
set(handles.text15,'String','');
set(handles.text15,'Visible','Off');
set(handles.text13,'Visible','Off');
set(handles.edit3,'Visible','Off');
set(handles.edit5,'Visible','Off');
set(handles.pushbutton16,'Visible','Off');
set(handles.radiobutton3,'Visible','Off');
set(handles.radiobutton6,'Visible','Off');

set(handles.pushbutton4,'Visible','Off');
set(handles.pushbutton11,'Visible','off');
nombre = get(handles.edit1,'string');
filename=strcat(nombre,'.mat');
```

```
existe = crearfichero(filename);
load(filename,'progresoDato');

if existe == 1
    set(handles.pushbutton2,'Enable','on');
else
    set(handles.pushbutton2,'Enable','off');
end


% --- Executes on selection change in popupmenu1.
function popupmenu1_Callback(hObject, eventdata, handles)
% hObject   handle to popupmenu1 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles   structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns popupmenu1 contents
    as cell array
%        contents{get(hObject,'Value')} returns selected item from popupmenu1
c = get(hObject,'Value');
nombre = get(handles.edit1,'string');
filename = strcat(nombre,'.mat');
load(filename,'progresoDato');
pruebas = getByName(progresoDato, 'SPR')
SPR = pruebas{c,1}
valores=(['',10,'Fev1: ',num2str(SPR),' L',10]);
set(handles.text4,'String',valores);
set(handles.text4,'Visible','On');
set(handles.pushbutton4,'Visible','On');
set(handles.pushbutton11,'Visible','On');
set(handles.text7,'String','Conectado');
set(handles.pushbutton5,'Enable','on');

set(handles.popupmenu1,'String','');
set(handles.text5,'Visible','off');
set(handles.popupmenu1,'Visible','off');
set(handles.pushbutton2,'Enable','off');



% --- Executes during object creation, after setting all properties.
function popupmenu1_CreateFcn(hObject, eventdata, handles)
% hObject   handle to popupmenu1 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles   empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in pushbutton5.
function pushbutton5_Callback(hObject, eventdata, handles)
% hObject   handle to pushbutton5 (see GCBO)
```

```
% eventdata reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close all;
INICIO;


% --- Executes on button press in pushbutton6.
function pushbutton6_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton6 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set(handles.pushbutton7,'Visible','off');
set(handles.pushbutton6,'Visible','off');
cla reset; %Para cerrar la gráfica (quita la leyenda y los datos)
set(handles.axes3,'Visible','off');
set(handles.pushbutton1,'Enable','off');
set(handles.pushbutton2,'Enable','off');

% --- Executes on button press in pushbutton7.
function pushbutton7_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton7 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

c = get(handles.popupmenu1,'Value');
nombre = get(handles.edit1,'string');
filename = strcat(nombre,'.mat');
figure(2);
ProgresoTensionGUI2(c,filename);


% --- Executes on button press in pushbutton9.
function pushbutton9_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton9 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set(handles.pushbutton9,'Enable','Off');
set(handles.pushbutton4,'Visible','off');
set(handles.pushbutton11,'Visible','off');
set(handles.pushbutton6,'Visible','off');
set(handles.pushbutton7,'Visible','off');
cla reset; %Para cerrar la gráfica (quita la leyenda y los datos)
set(handles.axes3,'Visible','off');
set(handles.text4,'Visible','off');
set(handles.text5,'Visible','off');
set(handles.text6,'Visible','off');
set(handles.text7,'Visible','off');
set(handles.popupmenu1,'Visible','off');
pause(1);

% %La primera vez es para indicarle que se trata del tensiómetro para cambiar
% %la velocidad del puerto serie

global puerto;
kp = strcat('30',10)
a = puertoserie(puerto,9600,kp);
```

```
% pause(5);

%kp = strcat('3',10)
a = ConectSPR(puerto,9600);
    if a=='Y'
        set(handles.text7,'String','Conectado');
        set(handles.text7,'ForegroundColor',[0 1 0]);
        set(handles.pushbutton1,'Enable','on');
        set(handles.pushbutton17,'Enable','on');
    end
    if a=='N'
        set(handles.text7,'String','Desconectado');
        set(handles.text7,'ForegroundColor',[0.64 0.08 0.18]);
        set(handles.pushbutton9,'Enable','on');
    end
set(handles.text6,'Visible','on');
set(handles.text7,'Visible','on');


% --- Executes on button press in pushbutton11.
function pushbutton11_Callback(hObject, eventdata, handles)
% hObject   handle to pushbutton11 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles   structure with handles and user data (see GUIDATA)

set(handles.text9,'Visible','On');
set(handles.text10,'Visible','On');
set(handles.text13,'Visible','On');
set(handles.edit3,'Visible','On');
set(handles.edit5,'Visible','On');
set(handles.pushbutton16,'Visible','On');
set(handles.uibuttongroup4,'Visible','On');
set(handles.radiobutton3,'Visible','On');
set(handles.radiobutton6,'Visible','On');




function edit3_Callback(hObject, eventdata, handles)
% hObject   handle to edit3 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles   structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit3 as text
%        str2double(get(hObject,'String')) returns contents of edit3 as a
    double
edad=str2double(get(handles.edit5,'String'))
altura=str2double(get(handles.edit3,'String'))
if((isnan(edad)==0) && (isnan(altura)==0) && (not(mod(edad,1))) && (not(mod(
    altura,1))))
    set(handles.pushbutton16,'Enable','on');
else
    set(handles.pushbutton16,'Enable','off');
end
```

```
% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject   handle to edit3 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles   empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit5_Callback(hObject, eventdata, handles)
% hObject   handle to edit5 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles   structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit5 as text
%        str2double(get(hObject,'String')) returns contents of edit5 as a
    double

edad=str2double(get(handles.edit5,'String'))
altura=str2double(get(handles.edit3,'String'))
%%not(mod(x,1)) sale 1 si x es numero entero, y 0 si es numero decimal
if((isnan(edad)==0) && (isnan(altura)==0) && (not(mod(edad,1))) && (not(mod(
    altura,1))))
    set(handles.pushbutton16,'Enable','on');
else
    set(handles.pushbutton16,'Enable','off');
end


% --- Executes during object creation, after setting all properties.
function edit5_CreateFcn(hObject, eventdata, handles)
% hObject   handle to edit5 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles   empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in pushbutton16.
function pushbutton16_Callback(hObject, eventdata, handles)
% hObject   handle to pushbutton16 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles   structure with handles and user data (see GUIDATA)

valor=get(handles.text4,'String') %Valor de FEV1 obtenido en str
[i,j]=size(valor)
```

```
Fev=str2double(valor(2,9:(j-2))) %Se pasa a numero el dato
edad=str2double(get(handles.edit5,'String'))
altura=str2double(get(handles.edit3,'String'))
hombre=get(handles.radiobutton3,'Value');
mujer=get(handles.radiobutton6,'Value');
if (mujer==1)
    Fvc=diagnosticoSPR(edad,altura,'M')
else
    Fvc=diagnosticoSPR(edad,altura,'H')
end
resultado=Fev/Fvc*100
if (resultado>80)
    diagnostico='Diagnostico normal';
elseif (resultado>50)
    diagnostico='Diagnostico leve';
elseif (resultado>30)
    diagnostico='Diagnostico moderado';
else
    diagnostico='Diagnostico grave (pulmonar obstructivo)';
end

texto=['El resultado es',10,num2str(resultado),' %',10,diagnostico])
set(handles.text15,'String',texto);
set(handles.text15,'Visible','On');


% --- Executes on button press in pushbutton17.
function pushbutton17_Callback(hObject, eventdata, handles)
% hObject   handle to pushbutton17 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles   structure with handles and user data (see GUIDATA)
set(hObject,'Enable','off')
set(handles.pushbutton5,'Enable','off')
set(handles.pushbutton1,'Enable','off')
pause(1);

ke = strcat('3330',10);
global puerto;
a = puertoserie(puerto,9600,ke);
set(handles.pushbutton1,'Enable','on')
set(handles.pushbutton5,'Enable','on')
```

Apéndice D
Código Matlab - GUI ECG

* Para que funcione la interfaz son necesarios los archivos .fig donde se configura la parte gráfica.

```
function varargout = GUI2(varargin)
% GUI2 MATLAB code for GUI2.fig
%      GUI2, by itself, creates a new GUI2 or raises the existing
%      singleton*.
%
%      H = GUI2 returns the handle to a new GUI2 or the handle to
%      the existing singleton*.
%
%      GUI2('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in GUI2.M with the given input arguments.
%
%      GUI2('Property','Value',...) creates a new GUI2 or raises the
%      existing singleton*. Starting from the left, property value pairs are
%      applied to the GUI before GUI2_OpeningFcn gets called. An
%      unrecognized property name or invalid value makes property application
%      stop. All inputs are passed to GUI2_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help GUI2

% Last Modified by GUIDE v2.5 29-Aug-2019 17:33:24

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',    mfilename, ...
                   'gui_Singleton', gui_Singleton, ...
                   'gui_OpeningFcn', @GUI2_OpeningFcn, ...
                   'gui_OutputFcn', @GUI2_OutputFcn, ...
                   'gui_LayoutFcn', [] , ...
                   'gui_Callback', []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
```

```
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before GUI2 is made visible.
function GUI2_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject   handle to figure
% eventdata reserved - to be defined in a future version of MATLAB
% handles   structure with handles and user data (see GUIDATA)
% varargin  command line arguments to GUI2 (see VARARGIN)

% Choose default command line output for GUI2
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes GUI2 wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = GUI2_OutputFcn(hObject, eventdata, handles)
% varargout cell array for returning output args (see VARARGOUT);
% hObject   handle to figure
% eventdata reserved - to be defined in a future version of MATLAB
% handles   structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject   handle to pushbutton1 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles   structure with handles and user data (see GUIDATA)
nombre=get(handles.edit1,'string');
filename=strcat(nombre,'.mat');
set(hObject,'Enable','off')
set(handles.pushbutton2,'Enable','off')
set(handles.pushbutton5,'Enable','off')
set(handles.pushbutton12,'Enable','off')
pause(1);
set(handles.text7,'String','Iniciando...')
pause(1);
kp = strcat('22',10);
global puerto;
puertoserie(puerto,115200,kp);
set(handles.text7,'String','Midiendo...')
pause(1);
```

```
dato = ECGGUI(filename,puerto);
set(handles.pushbutton2,'Enable','on');
set(handles.pushbutton12,'Enable','on');
valores = dato{1};
[x1,x2] = size(valores);
x = [0:(x1-1)];
axes=(handles.axes3);
plot(x,valores);
grid;
title(['Electrocardiograma. Fecha: ',datestr(dato{2})],'FontSize',13,'
    FontWeight','bold','Color','w');
xlabel('Muestras','FontSize',10,'FontWeight','bold','Color','w');
ylabel('Voltaje','FontSize',10,'FontWeight','bold','Color','w');
set(handles.pushbutton10,'Visible','on');
set(handles.pushbutton14,'Visible','on');
set(handles.pushbutton9,'Visible','on');
set(handles.pushbutton1,'Enable','off');
set(handles.pushbutton2,'Enable','off');
set(handles.pushbutton12,'Enable','off');
set(handles.text7,'String','Conectado');
set(handles.pushbutton5,'Enable','on');




% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject   handle to pushbutton2 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles   structure with handles and user data (see GUIDATA)

nombre = get(handles.edit1,'string');
filename = strcat(nombre,'.mat');
str = FechasECG(filename);
set(handles.pushbutton1,'Enable','off');
set(handles.pushbutton2,'Enable','off');
set(handles.pushbutton12,'Enable','off');

set(handles.popupmenu1,'String',str);
set(handles.text5,'Visible','on');
set(handles.popupmenu1,'Visible','on');




function edit1_Callback(hObject, eventdata, handles)
% hObject   handle to edit1 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles   structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as a
    double
nombre=get(hObject,'string');
%Creamos el fichero si no existe
```

```
filename=strcat(nombre,'.mat');
existe = crearficheroECG(filename);
set(handles.text3,'String','Usuario creado');
%set(handles.pushbutton1,'Enable','on');
set(handles.pushbutton11,'Visible','on');
set(handles.pushbutton11,'Enable','On');
set(handles.pushbutton9,'Visible','off');
set(handles.pushbutton10,'Visible','off');
set(handles.pushbutton14,'Visible','off');
set(handles.pushbutton6,'Visible','off');
set(handles.pushbutton15,'Visible','off');
set(handles.axes3,'Visible','off');
set(handles.text5,'Visible','off');
set(handles.text6,'Visible','off');
set(handles.text8,'Visible','off');
set(handles.text7,'Visible','off');
set(handles.pushbutton16,'Visible','off');
set(handles.popupmenu1,'Visible','off');



if existe == 1
    set(handles.pushbutton2,'Enable','on');
else
    set(handles.pushbutton2,'Enable','off');
end


% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject   handle to edit1 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles   empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



% --- Executes on selection change in popupmenu1.
function popupmenu1_Callback(hObject, eventdata, handles)
% hObject   handle to popupmenu1 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles   structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns popupmenu1 contents
    as cell array
%        contents{get(hObject,'Value')} returns selected item from popupmenu1
c = get(hObject,'Value');
nombre = get(handles.edit1,'string');
filename = strcat(nombre,'.mat');
%figure(2);
axes=(handles.axes3);
```

```
ProgresoECGGUI(c,filename);
set(handles.text5,'Visible','off');
set(handles.popupmenu1,'String','');
set(handles.text5,'Visible','off');
set(handles.popupmenu1,'Visible','off');
set(handles.pushbutton6,'Visible','on');
set(handles.pushbutton15,'Visible','on');
set(handles.pushbutton9,'Visible','on');



% --- Executes during object creation, after setting all properties.
function popupmenu1_CreateFcn(hObject, eventdata, handles)
% hObject   handle to popupmenu1 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles   empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in pushbutton5.
function pushbutton5_Callback(hObject, eventdata, handles)
% hObject   handle to pushbutton5 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles   structure with handles and user data (see GUIDATA)
close all;
INICIO;


% --- Executes on button press in pushbutton6.
function pushbutton6_Callback(hObject, eventdata, handles)
% hObject   handle to pushbutton6 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles   structure with handles and user data (see GUIDATA)
c = get(handles.popupmenu1,'Value');
nombre = get(handles.edit1,'string');
filename = strcat(nombre,'.mat');
figure(2);
ProgresoECGGUI2(c,filename);



% --- Executes on button press in pushbutton9.
function pushbutton9_Callback(hObject, eventdata, handles)
% hObject   handle to pushbutton9 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles   structure with handles and user data (see GUIDATA)
set(handles.pushbutton9,'Visible','off');
set(handles.pushbutton10,'Visible','off');
set(handles.pushbutton14,'Visible','off');
set(handles.pushbutton6,'Visible','off');
set(handles.pushbutton15,'Visible','off');
```

```
set(handles.axes3,'Visible','off');
set(handles.pushbutton1,'Enable','off');
set(handles.pushbutton2,'Enable','off');
set(handles.pushbutton12,'Enable','off');


% --- Executes on button press in pushbutton10.
function pushbutton10_Callback(hObject, eventdata, handles)
% hObject   handle to pushbutton10 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles   structure with handles and user data (see GUIDATA)

nombre = get(handles.edit1,'string');
filename = strcat(nombre,'.mat');
load(filename,'progresoDato');
pruebas = getByName(progresoDato, 'ECG');
[c, d] = size(pruebas)
figure(2);
ProgresoECGGUI2(c,filename);


% --- Executes on button press in pushbutton11.
function pushbutton11_Callback(hObject, eventdata, handles)
% hObject   handle to pushbutton11 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles   structure with handles and user data (see GUIDATA)
set(handles.pushbutton11,'Enable','Off');
set(handles.pushbutton9,'Visible','off');
set(handles.pushbutton10,'Visible','off');
set(handles.pushbutton14,'Visible','off');
set(handles.axes3,'Visible','off');
set(handles.text5,'Visible','off');
set(handles.text6,'Visible','off');
set(handles.text7,'Visible','off');
set(handles.popupmenu1,'Visible','off');
pause(1);


% %La primera vez es para indicarle que se trata del ECG para cambiar
% %la velocidad del puerto serie
kp = strcat('20',10)
global puerto;
a = puertoserie(puerto,9600,kp);


kp = strcat('2',10);
a = puertoserie(puerto,115200,kp);

    if a=='Y'
        set(handles.text7,'String','Conectado');
        set(handles.text7,'ForegroundColor',[0 1 0]);
        set(handles.pushbutton1,'Enable','on');
        set(handles.pushbutton12,'Enable','on');
    end
    if a=='N'
        set(handles.text7,'String','Desconectado');
        set(handles.text7,'ForegroundColor',[0.64 0.08 0.18]);
```

```
        set(handles.pushbutton11,'Enable','on');
    end
set(handles.text6,'Visible','on');
set(handles.text7,'Visible','on');


% --- Executes on button press in pushbutton12.
function pushbutton12_Callback(hObject, eventdata, handles)
% hObject   handle to pushbutton12 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles   structure with handles and user data (see GUIDATA)

set(hObject,'Enable','off')
set(handles.pushbutton2,'Enable','off')
set(handles.pushbutton5,'Enable','off')
set(handles.pushbutton1,'Enable','off')
pause(1);
set(handles.text7,'String','Iniciando...')
pause(1);
kp = strcat('22',10);
global puerto;
puertoserie(puerto,115200,kp);
set(handles.text7,'String','Midiendo...')
pause(1);

dato = ECGcalibrar(puerto);
set(handles.pushbutton2,'Enable','on');
set(handles.pushbutton1,'Enable','on');
valores = dato{1};
[x1,x2] = size(valores);
x = [0:(x1-1)];
axes=(handles.axes3);
plot(x,valores);
grid;
title(['Electrocardiograma. Fecha: ',datestr(dato{2})],'FontSize',13,'
    FontWeight','bold','Color','w');
xlabel('Muestras','FontSize',10,'FontWeight','bold','Color','w');
ylabel('Voltaje','FontSize',10,'FontWeight','bold','Color','w');
set(handles.pushbutton9,'Visible','on');
set(handles.pushbutton1,'Enable','off');
set(handles.pushbutton2,'Enable','off');
set(handles.pushbutton12,'Enable','off');
set(handles.text7,'String','Conectado');
set(handles.pushbutton5,'Enable','on');


% --- Executes on button press in pushbutton14.
function pushbutton14_Callback(hObject, eventdata, handles)
% hObject   handle to pushbutton14 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles   structure with handles and user data (see GUIDATA)
nombre = get(handles.edit1,'string');
filename = strcat(nombre,'.mat');
load(filename,'progresoDato');
pruebas = getByName(progresoDato, 'ECG');
[c, d] = size(pruebas)
diagnostico=DiagnosticoECG(c,filename);
```

```
set(handles.pushbutton9,'Visible','off');
set(handles.pushbutton10,'Visible','off');
set(handles.pushbutton14,'Visible','off');
set(handles.pushbutton6,'Visible','off');
set(handles.pushbutton15,'Visible','off');
set(handles.axes3,'Visible','off');
set(handles.pushbutton1,'Enable','off');
set(handles.pushbutton2,'Enable','off');
set(handles.pushbutton12,'Enable','off');
valores=['',10,diagnostico,10];
set(handles.text8,'String',valores);
set(handles.text8,'Visible','On');
set(handles.pushbutton16,'Visible','on');


% --- Executes on button press in pushbutton15.
function pushbutton15_Callback(hObject, eventdata, handles)
% hObject   handle to pushbutton15 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles   structure with handles and user data (see GUIDATA)
c = get(handles.popupmenu1,'Value');
nombre = get(handles.edit1,'string');
filename = strcat(nombre,'.mat');
diagnostico=DiagnosticoECG(c,filename);
set(handles.pushbutton9,'Visible','off');
set(handles.pushbutton10,'Visible','off');
set(handles.pushbutton14,'Visible','off');
set(handles.pushbutton6,'Visible','off');
set(handles.pushbutton15,'Visible','off');
set(handles.axes3,'Visible','off');
set(handles.pushbutton1,'Enable','off');
set(handles.pushbutton2,'Enable','off');
set(handles.pushbutton12,'Enable','off');
valores=['',10,diagnostico,10];
set(handles.text8,'String',valores);
set(handles.text8,'Visible','On');
set(handles.pushbutton16,'Visible','on');


% --- Executes on button press in pushbutton16.
function pushbutton16_Callback(hObject, eventdata, handles)
% hObject   handle to pushbutton16 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles   structure with handles and user data (see GUIDATA)
set(handles.text8,'String','');
set(handles.text8,'Visible','Off');
set(handles.pushbutton16,'Visible','off');
```

Apéndice E
Código Matlab - GUI INICIO

* Para que funcione la interfaz son necesarios los archivos .fig donde se configura la parte gráfica.

```
function varargout = INICIO(varargin)
% INICIO MATLAB code for INICIO.fig
%      INICIO, by itself, creates a new INICIO or raises the existing
%      singleton*.
%
%      H = INICIO returns the handle to a new INICIO or the handle to
%      the existing singleton*.
%
%      INICIO('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in INICIO.M with the given input arguments.
%
%      INICIO('Property','Value',...) creates a new INICIO or raises the
%      existing singleton*. Starting from the left, property value pairs are
%      applied to the GUI before INICIO_OpeningFcn gets called. An
%      unrecognized property name or invalid value makes property application
%      stop. All inputs are passed to INICIO_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help INICIO

% Last Modified by GUIDE v2.5 19-Mar-2019 18:35:18

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',    mfilename, ...
                   'gui_Singleton', gui_Singleton, ...
                   'gui_OpeningFcn', @INICIO_OpeningFcn, ...
                   'gui_OutputFcn', @INICIO_OutputFcn, ...
                   'gui_LayoutFcn', [] , ...
                   'gui_Callback', []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
```

```
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before INICIO is made visible.
function INICIO_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject   handle to figure
% eventdata reserved - to be defined in a future version of MATLAB
% handles   structure with handles and user data (see GUIDATA)
% varargin  command line arguments to INICIO (see VARARGIN)

% Choose default command line output for INICIO
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

global puertos;
puertos = seriallist; %Variable global para acceder a los puertos desde todas
    las funciones
if isempty(puertos)==0
    set(handles.popupmenu2,'String',puertos);
    set(handles.text8,'Visible','on');
    set(handles.popupmenu2,'Visible','on');
else
    set(handles.text9,'Visible','on');
end




% UIWAIT makes INICIO wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = INICIO_OutputFcn(hObject, eventdata, handles)
% varargout cell array for returning output args (see VARARGOUT);
% hObject   handle to figure
% eventdata reserved - to be defined in a future version of MATLAB
% handles   structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in pushbutton5.
function pushbutton5_Callback(hObject, eventdata, handles)
% hObject   handle to pushbutton5 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles   structure with handles and user data (see GUIDATA)
```

```
close all;
GUI1;

% --- Executes on button press in pushbutton6.
function pushbutton6_Callback(hObject, eventdata, handles)
% hObject   handle to pushbutton6 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles   structure with handles and user data (see GUIDATA)
close all;
GUI2;


% --- Executes on selection change in popupmenu2.
function popupmenu2_Callback(hObject, eventdata, handles)
% hObject   handle to popupmenu2 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles   structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns popupmenu2 contents
    as cell array
%       contents{get(hObject,'Value')} returns selected item from popupmenu2

global puertos; %Debemos poner la variable global en cada función
global puerto;
c = get(hObject,'Value');
puerto=puertos(c);
set(handles.pushbutton5,'Enable','on');
set(handles.pushbutton6,'Enable','on');
set(handles.pushbutton7,'Enable','on');


% --- Executes during object creation, after setting all properties.
function popupmenu2_CreateFcn(hObject, eventdata, handles)
% hObject   handle to popupmenu2 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles   empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in pushbutton7.
function pushbutton7_Callback(hObject, eventdata, handles)
% hObject   handle to pushbutton7 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles   structure with handles and user data (see GUIDATA)
close all;
GUI3;
```

Apéndice F
Código Matlab - Funciones

Aquí se pondrán las funciones que se utilizan en la interfaz.

Código F.1 Función ConectSPR.

```
function a=ConectSPR(puertoA,velocidad)
n=0;
%close all;
clc;
valor=0;

%Borra datos que se encuentren previos y vuelve a declarar el puerto y la
%velocidad de transmisión
delete(instrfind({'port'},{puertoA}));
puerto=serial(puertoA);
puerto.BaudRate=velocidad;

fopen(puerto);%abre el puerto a utilizar

a=[];
ke = strcat('3',10);


while(n==0) %isempty(a)==1 n==0
 fprintf(puerto, '%s', ke);
 %a=fread(PS,1,'char')
 a=fscanf(puerto,'%d',3);
 if (isempty(a)==0)
     if ((a=='Y') | (a=='N'))
         n=1;
     end
 end
end



%cierra y borra el puerto utilizado, borra todas las variables utilizadas
delete(instrfind({'port'},{puertoA}));
fclose(serial(puertoA));
delete(serial(puertoA));
```

```
end
```

Código F.2 Función ContadorFechas.

```
function [v, pruebas] = ContadorFechas( filename )

load(filename,'progresoDato');
pruebas = getByName(progresoDato, 'tension'); %Busca las pruebas del tensió
    metro
[filas, columnas] = size(pruebas);

i = 1; %Contador de pruebas
m = 1; %Contador de semanas con datos
fechaFinal = pruebas{filas,2};
t0 = pruebas{1,2};%Inicializamos la variable con la fecha inicial
t = datestr(addtodate(datenum(t0), 7, 'day'));
v=[1]; %Vector de semanas en los que se ha realizado la prueba

while(t<fechaFinal)
        fecha = pruebas{i,2};
        if ((isbetween(fecha,t0,t))==0)
            while(t<fecha) %Actualizamos t0,t y m hasta alcanzar la fecha
                t0 = datestr(t);
                t = datestr(addtodate(datenum(t), 7, 'day'));
                m=m+1;
            end
                if (v(end)~=m)
                    v = [v;m];
                end
        end
    i = i+1;
end


end
```

Código F.3 Función CrearFichero.

```
function existe = crearfichero( filename )

if exist(filename, 'file')
     load(filename,'progresoDato');
     pruebas = getByName(progresoDato, 'tension');

     [i,j]=size(pruebas);
     if i>1
         existe = 1;
     else
         existe = 0;
     end
else
     progresoDato = [];
```

```
     save(filename,'progresoDato');
     existe = 0;
end


end
```

Código F.4 Función DiagnosticoECG.

```
function diagnostico=DiagnosticoECG(c,filename)

    load(filename,'progresoDato');

    %Busca las pruebas del tensiómetro
    pruebas = getByName(progresoDato, 'ECG');

    %Representa lo indicado en la GUI
    ECG = pruebas{c,1};


%PRUEBA R


%Para obtener los valores con los que comparar, se ha hecho la media
%de varias pruebas

[x1,x2] = size(ECG);
x = [0:(x1-1)];
maximos=[];
basal=mean(ECG);
vector=ECG-basal;
R=[];
aux1=0;
aux2=0;
a=0;

for k=7:length(vector)-6
    if vector(k)>1 %Para R
        %Se comprueba si es mayor o igual en un entorno de 6 muestras. De esta
            forma se busca el máximo
        if (vector(k)>vector(k-1) || vector(k)==vector(k-1)) && (vector(k)>
            vector(k+1) || vector(k)==vector(k+1)) && (vector(k)>vector(k+2) ||
            vector(k)==vector(k+2)) && (vector(k)>vector(k-2) || vector(k)==
            vector(k-2)) && (vector(k)>vector(k+3) || vector(k)==vector(k+3)) &&
             (vector(k)>vector(k-3) || vector(k)==vector(k-3)) && (vector(k)>
            vector(k+4) || vector(k)==vector(k+4)) && (vector(k)>vector(k-4) ||
            vector(k)==vector(k-4)) && (vector(k)>vector(k+5) || vector(k)==
            vector(k+5)) && (vector(k)>vector(k-5) || vector(k)==vector(k-5)) &&
             (vector(k)>vector(k+6) || vector(k)==vector(k+6)) && (vector(k)>
            vector(k-6) || vector(k)==vector(k-6))
          if vector(k)>aux1
              aux1=vector(k);
              aux2=k;
          end
        end
```

```
    end
    a=a+1;
    if(a==50) %Se comprueban máximos en un entorno de 50 muestras
        if(aux1~=0)
            maximos(end+1)=aux1;
        end
        if(aux2~=0)
            R=[R;aux2];
        end
        a=0;
        aux1=0;
        aux2=0;
    end
end

%plot(x,ECG,x(R),ECG(R),'*');



%-------------DIAGNOSTICO---------------%

% [p1,p2]=size(P);
[r1,r2]=size(R);
resta=[];
% restap=[];

for i=2:r1
 rr=R(i)-R(i-1);
 resta=[resta;rr];
end

% for i=2:p1
%  pr=P(i)-P(i-1)
%  restap=[restap;pr]
% end



for i=1:size(resta)
    if (abs(mean(resta)-resta(i))>40)
        diagnostico='Arritmia: Ritmo irregular';
    elseif (resta(i)>500)
        diagnostico='Arritmia: Bradicardia';
    elseif (resta(i)<400)
        diagnostico='Arritmia: Taquicardia';
    else
        diagnostico='ECG normal';
    end
end


end
```

Código F.5 Función diagnosticoSPR.

```
function resultado=diagnosticoSPR(edad,altura,genero)
%UNTITLED3 Summary of this function goes here
%   Detailed explanation goes here
M=  [2.24, 2.40, 2.57, 2.73, 2.90, 3.06, 3.23, 3.39, 3.56, 3.72, 3.89;
     2.42, 2.59, 2.75, 2.92, 3.08, 3.25, 3.41, 3.58, 3.74, 3.91, 4.07;
     2.60, 2.77, 2.93, 3.10, 3.26, 3.43, 3.59, 3.76, 3.92, 4.09, 4.25;
     2.79, 2.95, 3.12, 3.28, 3.45, 3.61, 3.78, 3.94, 4.11, 4.27, 4.44;
     2.97, 3.14, 3.30, 3.47, 3.63, 3.80, 3.96, 4.13, 4.29, 4.46, 4.62;
     3.15, 3.34, 3.52, 3.71, 3.89, 4.08, 4.26, 4.45, 4.63, 4.82, 5.00;
     3.04, 3.23, 3.41, 3.60, 3.78, 3.97, 4.15, 4.34, 4.52, 4.71, 4.89;
     2.93, 3.12, 3.30, 3.49, 3.67, 3.86, 4.04, 4.23, 4.41, 4.60, 4.78;
     2.71, 2.90, 3.08, 3.27, 3.45, 3.64, 3.82, 4.01, 4.19, 4.38, 4.56;
     2.49, 2.68, 2.86, 3.05, 3.23, 3.42, 3.60, 3.79, 3.97, 4.16, 4.34;
     2.27, 2.46, 2.64, 2.83, 3.01, 3.20, 3.38, 3.57, 3.75, 3.94, 4.12;
     2.05, 2.24, 2.42, 2.61, 2.79, 2.98, 3.16, 3.35, 3.53, 3.72, 3.90;
     1.83, 2.02, 2.20, 2.39, 2.57, 2.76, 2.94, 3.13, 3.31, 3.50, 3.68];


 H= [2.52, 2.77, 3.02, 3.27, 3.52, 3.77, 4.02, 4.27, 4.52, 4.77, 5.02;
     2.68, 2.93, 3.18, 3.43, 3.68, 3.93, 4.18, 4.43, 4.68, 4.93, 5.18;
     2.83, 3.08, 3.33, 3.58, 3.83, 4.08, 4.33, 4.58, 4.83, 5.08, 5.33;
     2.99, 3.24, 3.49, 3.74, 3.99, 4.24, 4.49, 4.74, 4.99, 5.24, 5.49;
     3.15, 3.40, 3.65, 3.90, 4.15, 4.40, 4.65, 4.90, 5.15, 5.40, 5.65;
     3.30, 3.55, 3.80, 4.05, 4.30, 4.55, 4.80, 5.05, 5.30, 5.55, 5.80;
     3.24, 3.57, 3.89, 4.22, 4.54, 4.87, 5.19, 5.52, 5.84, 6.17, 6.49;
     3.10, 3.42, 3.75, 4.07, 4.40, 4.72, 5.05, 5.37, 5.70, 6.02, 6.35;
     2.81, 3.13, 3.46, 3.78, 4.11, 4.43, 4.76, 5.08, 5.41, 5.73, 6.06;
     2.52, 2.84, 3.17, 3.49, 3.82, 4.14, 4.47, 4.79, 5.12, 5.44, 5.77;
     2.23, 2.55, 2.88, 3.20, 3.53, 3.85, 4.18, 4.50, 4.83, 5.15, 5.48;
     1.94, 2.26, 2.59, 2.91, 3.24, 3.56, 3.89, 4.21, 4.54, 4.86, 5.19;
     1.65, 1.97, 2.30, 2.62, 2.95, 3.27, 3.60, 3.92, 4.25, 4.57, 4.90];


if (edad<11)
       i=1;
elseif (edad<13)
       i=2;
elseif (edad<15)
       i=3;
elseif (edad<17)
       i=4;
elseif (edad<19)
       i=5;
elseif (edad<23)
       i=6;
elseif (edad<28)
       i=7;
elseif (edad<35)
       i=8;
elseif (edad<45)
       i=9;
elseif (edad<55)
       i=10;
elseif (edad<65)
       i=11;
elseif (edad<75)
       i=12;
```

```
else
       i=13;
end



if (altura<148)
       j=1;
elseif (altura<153)
       j=2;
elseif (altura<158)
       j=3;
elseif (altura<163)
       j=4;
elseif (altura<168)
       j=5;
elseif (altura<173)
       j=6;
elseif (altura<178)
       j=7;
elseif (altura<183)
       j=8;
elseif (altura<188)
       j=9;
elseif (altura<193)
       j=10;
else
       j=11;
end

if (genero=='M')
    resultado=M(i,j);
else
    resultado=H(i,j);
end


end
```

Código F.6 Función ECG.

```
function dato=ECG(puertoA,velocidad)

%close all;
clc;
valores = []; %Declara vector en el cual se van a guardar los valores

%Borra datos que se encuentren previos y vuelve a declarar el puerto y la
%velocidad de transmisión
delete(instrfind({'port'},{puertoA}));
puerto=serial(puertoA);
puerto.BaudRate=velocidad;

fopen(puerto);%abre el puerto a utilizar
```

```
valor = 0;
numero = [];
i = 0;


a=[];
ke = strcat('222',10);

while(isempty(a)==1) %isempty(a)==1
 fprintf(puerto, '%s', ke);
 %a=fread(PS,1,'char')
 a=fscanf(puerto,'%d',2)
end


while (valor~='M')
valor = fread(puerto,1,'char');
end


while (valor~='X')
    for i=1:6
        valor = fread(puerto,1,'char');
        if (i<5)
            numero=[numero,valor];
        end
    end
    if (size(numero) == [1 4])
        cad = strcat(numero(1),numero(2),numero(3),numero(4)); %Unimos los
            caracteres para formar el float
    end
    valores = [valores; str2num(cad)]; %Lo transformamos en numeros
    numero = [];

end


dato = {valores, datetime('now'),'ECG'};



%cierra y borra el puerto utilizado, borra todas las variables utilizadas
delete(instrfind({'port'},{puertoA}));
fclose(serial(puertoA));
delete(serial(puertoA));
end
```

Código F.7 Función ECGcalibrar.

```
function dato = ECGcalibrar(puerto)
%UNTITLED5 Summary of this function goes here
%   Detailed explanation goes here
```

```
dato = ECG(puerto,115200);

end
```

Código F.8 Función ECGGUI.

```
function dato = ECGGUI(filename,puerto)
%UNTITLED5 Summary of this function goes here
%   Detailed explanation goes here

load(filename,'progresoDato');

dato = ECG(puerto,115200);

progresoDato = [progresoDato; dato];

save(filename,'progresoDato');

end
```

Código F.9 Función FechasECG.

```
function str = FechasECG( filename )

load(filename,'progresoDato');
pruebas = getByName(progresoDato, 'ECG'); %Busca las pruebas del tensiómetro
[filas, columnas] = size(pruebas);
str = '';


for i=1:filas
    fecha = pruebas{i,2};
    cad = datestr(fecha);
    str = strvcat(str,cad);
end


end
```

Código F.10 Función FechasSPR.

```
function str = FechasSPR( filename )

load(filename,'progresoDato');
pruebas = getByName(progresoDato, 'SPR'); %Busca las pruebas del tensiómetro
[filas, columnas] = size(pruebas);
str = '';


for i=1:filas
    fecha = pruebas{i,2};
```

```
    cad = datestr(fecha);
    str = strvcat(str,cad);
end


end
```

Código F.11 Función getByName.

```
function datos = getByName(progreso, name)

[i, j] = ind2sub(size(progreso), find(strcmp(progreso, name)));

if isempty(i)
    datos = [];
else
    datos = reshape({progreso{i,:}}, [length(i), length({progreso{i,:}})/length
        (i)]);
end

end
```

Código F.12 Función graficaTension.

```
function graficaTension(progreso, vectorValores)

systolic=[];
diastolic=[];
pulse=[];
time=[];

for i = vectorValores
    systolic = [systolic; progreso{i,1}.systolic];
    diastolic = [diastolic; progreso{i,1}.diastolic];
    pulse = [pulse; progreso{i,1}.pulse];
    time = [time; progreso{i,2}];
end

%Añadimos líneas para indicar los valores normales máximos
plot([time(1),time(end)],[120,120],'r'); hold on;
plot([time(1),time(end)],[80,80]),'b'; hold on;
plot([time(1),time(end)],[100,100],'g'); hold on;
plot(time,systolic,'r*--'); hold on;
plot(time,diastolic,'b*--'); hold on;
plot(time,pulse,'g*--'); hold on;
grid;

title({'Representación de la tensión sistólica, diastólica y pulso'},'FontSize'
    ,13,'FontWeight','bold','Color','w');
xlabel('Fecha','FontSize',10,'FontWeight','bold','Color','w');
ylabel('mmHg / pulsaciones(min)','FontSize',10,'FontWeight','bold','Color','w')
    ;
legend('sistólica','diastólica','pulso','Location', 'Best');
```

```
end
```

Código F.13 Función MedirTensionGUI.

```
function dato = MedirTensionGUI( filename,puerto)
%UNTITLED5 Summary of this function goes here
%   Detailed explanation goes here

load(filename,'progresoDato');

dato = tensiometro(puerto,19200);

progresoDato = [progresoDato; dato];

save(filename,'progresoDato');

end
```

Código F.14 Función ProgresoECGGUI.

```
function ProgresoECGGUI(c,filename)

    load(filename,'progresoDato');

    %Busca las pruebas del tensiómetro
    pruebas = getByName(progresoDato, 'ECG');

    %Representa lo indicado en la GUI
    ECG = pruebas{c,1};
    [x1,x2] = size(ECG);
    x = [0:(x1-1)];
    plot(x,ECG);

    grid;
    title(['Electrocardiograma. Fecha: ',datestr(pruebas{c,2})],'FontSize',13,'
        FontWeight','bold','Color','w');
    xlabel('Muestras','FontSize',10,'FontWeight','bold','Color','w');
    ylabel('Voltaje','FontSize',10,'FontWeight','bold','Color','w');



end
```

Código F.15 Función ProgresoTensionGUI.

```
function ProgresoTensionGUI(c,filename)

    load(filename,'progresoDato');

    %Busca las pruebas del tensiómetro
```

```
    pruebas = getByName(progresoDato, 'tension');
    [i, j] = size(pruebas);
    n=1;
    v=[];
    [w, pruebas] = ContadorFechas(filename);

    %c es el intervalo que queremos, debemos buscar esas fechas en pruebas:

    fechaInicial = pruebas{1,2};%Inicializamos la variable con la fecha inicial

    %t0 va a ser la primera fecha del rango, y t la final
    %por eso hay que actualizar ambas según el valor de c

    t = datestr(addtodate(datenum(fechaInicial), w(c)*7, 'day'));
    t0 = datestr(addtodate(datenum(t), -7, 'day'));


    for m = 1:i
        fecha = pruebas{m,2};
        if isbetween(fecha,t0,t);
            v = [v,m];
        end
    end

    %Representa lo indicado en la GUI
    graficaTension(pruebas,v);


end
```

Código F.16 Función SPR.

```
function [dato,a]=SPR(puertoA,velocidad)

n=0;
m=0;
%close all;
clc;
valor=0;

%Borra datos que se encuentren previos y vuelve a declarar el puerto y la
%velocidad de transmisión
delete(instrfind({'port'},{puertoA}));
puerto=serial(puertoA);
puerto.BaudRate=velocidad;

fopen(puerto);%abre el puerto a utilizar

a=[];
ke = strcat('333',10);
ki = strcat('3333',10);

while(n==0) %isempty(a)==1

  fprintf(puerto, '%s', ke);
```

```
  a=fscanf(puerto,'%d',11)
  [x,y]=size(a);

 if (y==3)
    if (a=='EMP')
       n=1;
    end
    if (a=='NUM')
       n=1;
    end
 end
end

n=0;

while(n==0) %isempty(a)==1
  fprintf(puerto, '%s', ki);
  valor1=fscanf(puerto,'%d',3)
  valor2=fscanf(puerto,'%d',3)
  valor3=fscanf(puerto,'%d',3)
  if ((isempty(valor1)==0) && (isempty(valor2)==0) && (isempty(valor3)==0))
      if ((valor1==valor2) && (valor2==valor3))
          n=1;
          valor=valor1;
      end
  end
  m=m+1;
  %Añadimos la siguiente comprobación por si el dato guardado en la EEPROM
  %no es válido
  if (m==10)
      valor=0;
      a='ERR';
      n=1;
  end
end


if valor>600
    valor=valor/1000;
else
    valor=valor/100;
end


dato = {valor, datetime('now'),'SPR'};

%cierra y borra el puerto utilizado, borra todas las variables utilizadas
delete(instrfind({'port'},{puertoA}));
fclose(serial(puertoA));
delete(serial(puertoA));

end
```

Código F.17 Función tensiometro.

```
function dato=tensiometro(puertoA,velocidad)

%close all;
clc;
tension=struct('systolic', 0, 'diastolic', 0, 'pulse', 0); %Declara variable en
     la cual se van a guardar los valores

%Borra datos que se encuentren previos y vuelve a declarar el puerto y la
%velocidad de transmisión
delete(instrfind({'port'},{puertoA}));
puerto=serial(puertoA);
puerto.BaudRate=velocidad;

fopen(puerto);%abre el puerto a utilizar


%presión diastólica (mínima/baja)

diastolic=[0];

a=[];

ke = strcat('111',10);

while(isempty(a)==1) %isempty(a)==1
 fprintf(puerto, '%s', ke);
 %a=fread(PS,1,'char')
 a=fscanf(puerto,'%d',2)
end

while (diastolic(end) ~= 'i') % i que sale antes del valor válido
    diastolic=[diastolic; fread(puerto, 1, 'char')];
end

diastolic=[diastolic; fread(puerto, 1, 'char')];

if diastolic(end)=='1' % Si es igual a 1
    diastolic=[diastolic; fread(puerto, 1, 'char')];
    diastolic=[diastolic; fread(puerto, 1, 'char')];
    disp(['La presión diastólica es 1', diastolic(length(diastolic)-1),
        diastolic(end)]);
    tension.diastolic=['1',diastolic(length(diastolic)-1), diastolic(end)];
else
    diastolic=[diastolic; fread(puerto, 1, 'char')];
    disp(['La presión diastólica es ', diastolic(length(diastolic)-1),
        diastolic(end)]);
    tension.diastolic=['0',diastolic(length(diastolic)-1), diastolic(end)];
end


%presión sistólica (máxima/alta)
systolic=[0];


while (systolic(end) ~= 10) % nueva linea
    systolic=[systolic; fread(puerto, 1, 'char')];
```

```
end

systolic=[systolic; fread(puerto, 1, 'char')];

if systolic(end)=='1' % Si es igual a 1
    systolic=[systolic; fread(puerto, 1, 'char')];
    systolic=[systolic; fread(puerto, 1, 'char')];
    disp(['La presión sistólica es 1', systolic(length(systolic)-1), systolic(
        end)]);
    tension.systolic=['1',systolic(length(systolic)-1), systolic(end)];
else
    systolic=[systolic; fread(puerto, 1, 'char')];
    disp(['La presión sistólica es ', systolic(length(systolic)-1), systolic(
        end)]);
    tension.systolic=['0',systolic(length(systolic)-1), systolic(end)];
end


%pulso
pulse=[0];

while (pulse(end) ~= 10) % nueva línea
    pulse=[pulse; fread(puerto, 1, 'char')];
end

pulse=[pulse; fread(puerto, 1, 'char')];

if pulse(end)=='1' % Si es igual a 1
    pulse=[pulse; fread(puerto, 1, 'char')];
    pulse=[pulse; fread(puerto, 1, 'char')];
    disp(['El pulso es 1', pulse(length(pulse)-1), pulse(end)]);
    tension.pulse=['1', pulse(length(pulse)-1), pulse(end)];
else
    pulse=[pulse; fread(puerto, 1, 'char')];
    disp(['El pulso es ', pulse(length(pulse)-1), pulse(end)]);
    tension.pulse=['0',pulse(length(pulse)-1), pulse(end)];
end


%Pasamos los valores de la cadena a valores numericos
tension.diastolic=str2num(tension.diastolic);
tension.systolic=str2num(tension.systolic);
tension.pulse=str2num(tension.pulse);

%Creamos un cell con los parámetros y la fecha/hora
dato = {tension, datetime('now'),'tension'};


%cierra y borra el puerto utilizado, borra todas las variables utilizadas
delete(instrfind({'port'},{puertoA}));
fclose(serial(puertoA));
delete(serial(puertoA));
end
```

Índice de Figuras

1.1 Willem Kolff, diseñador del primer equipo para diálisis 1

2.1 Arduino Uno 3
2.2 MySignals HW 4

3.1 Incluir librerías en Arduino 7
3.2 Incluir librerías en Arduino 8
3.3 Incluir librerías en Arduino 8
3.4 Incluir librerías en Arduino 9
3.5 Notificación en Arduino 9
3.6 Escudo Mysignals Arduino 10
3.7 Conexionado de Sensores 10
3.8 Conexión pulsioxímetro 11
3.9 Conexión tensiómetro 11
3.10 Ejemplo tensiómetro 12
3.11 Comprobación puerto 13
3.12 Subir ejemplo 14
3.13 Ejecución ejemplo 15
3.14 Monitor Serie ejemplo 16
3.15 Colocación correcta del tensiómetro. Imagen disponible en: https://www.cooking-hacks.com/mysignals-hw-ehealth-medical-biometric-iot-platform-arduino-tutorial/ [Consultado el 22-08-2019] 16
3.16 Resultados tensiómetro. Imagen disponible en: https://www.cooking-hacks.com/mysignals-hw-ehealth-medical-biometric-iot-platform-arduino-tutorial/ [Consultado el 22-08-2019] 17

4.1 Tensiómetro. Imagen disponible en: https://www.cooking-hacks.com/mysignals-hw-ehealth-medical-biometric-iot-platform-arduino-tutorial/ [Consultado el 19-08-2019] 19
4.2 Funcionamiento del Tensiómetro. Imagen disponible en: https://mitensiometro.com/como-funciona-un-tensiometro/ [Consultado el 19-08-2019] 20
4.3 Grados de presión arterial. Imagen disponible en: https://www.comprartensiometro.net/tension-arterial/cuales-son-los-valores-normales-de-tension-arterial/ [Consultado el 19-08-2019] 20
4.4 GUI Inicio 21
4.5 GUI Inicio 22
4.6 GUI Tensiómetro 22
4.7 GUI Tensiómetro 23
4.8 GUI Tensiómetro 23
4.9 Posición correcta del tensiómetro. Imagen disponible en: https://www.cooking-hacks.com/mysignals-hw-ehealth-medical-biometric-iot-platform-arduino-tutorial/ [Consultado el 20-08-2019] 24
4.10 GUI Tensiómetro 24
4.11 GUI Tensiómetro 25
4.12 GUI Tensiómetro 25
4.13 GUI Tensiómetro 26

5.1 Espirómetro. Imagen disponible en: https://www.cooking-hacks.com/mysignals-hw-ehealth-medical-biometric-iot-platform-arduino-tutorial/ [Consultado el 20-08-2019] 29
5.2 Fev1 en hombres. Imagen disponible en: https://www.ugr.es/ jhuertas/EvaluacionFisiologica/Espirometria/volteoricostablas.htm [Consultado el 20-08-2019] 30
5.3 Fev1 en mujeres. Imagen disponible en: https://www.ugr.es/ jhuertas/EvaluacionFisiologica/Espirometria/volteoricostablas.htm [Consultado el 20-08-2019] 31
5.4 Primer paso espirómetro 32
5.5 Colocación correcta del espirómetro 32
5.6 Colocación correcta del espirómetro 33
5.7 GUI Inicio 33
5.8 GUI Inicio 34
5.9 GUI Inicio 34
5.10 GUI Inicio 35
5.11 FVC en hombres. Imagen disponible en: https://www.ugr.es/ jhuertas/EvaluacionFisiologica/Espirometria/volteoricostablas.htm [Consultado el 21-08-2019] 36
5.12 FVC en mujeres. Imagen disponible en: https://www.ugr.es/ jhuertas/EvaluacionFisiologica/Espirometria/volteoricostablas.htm [Consultado el 20-08-2019] 36

6.1 Electrocardiógrafo. Imagen disponible en: https://www.cooking-hacks.com/mysignals-hw-ehealth-medical-biometric-iot-platform-arduino-tutorial/ [Consultado el 24-08-2019] 41
6.2 Sistema de conducción del corazón. Imagen disponible en: *http:// www.electrocardiografia.es/ a_electrica.html* [Consultado el 28-08-2019] 42
6.3 Ondas ECG. Imagen disponible en: *http:// www.ub.edu/ LabFisio/ index.php?option=com_content&view=article&id=29&Itemid=148* [Consultado el 29-08-2019] 43
6.4 GUI Inicio 44
6.5 GUI Inicio 44
6.6 GUI ECG 45
6.7 GUI ECG 45
6.8 GUI ECG 46
6.9 GUI ECG 46
6.10 GUI ECG 47
6.11 GUI ECG 47
6.12 ECG normal 48
6.13 ECG taquicardia 48

7.1 Planos de la caja protectora 51
7.2 Planos de la tapa de la caja protectora 52
7.3 Tapadera de la caja protectora. Vista 3D. 53
7.4 Caja protectora. Vista 3D. 53

Índice de Códigos

3.1	Programación resultado tensión por TFT	17
4.1	Programación diagnóstico tensión	26
5.1	Programación diagnóstico espirómetro	37
5.2	Programación diagnóstico espirómetro. Tablas	38
6.1	Programación diagnóstico ECG	49
F.1	Función ConectSPR	97
F.2	Función ContadorFechas	98
F.3	Función CrearFichero	98
F.4	Función DiagnosticoECG	99
F.5	Función diagnosticoSPR	100
F.6	Función ECG	102
F.7	Función ECGcalibrar	103
F.8	Función ECGGUI	104
F.9	Función FechasECG	104
F.10	Función FechasSPR	104
F.11	Función getByName	105
F.12	Función graficaTension	105
F.13	Función MedirTensionGUI	106
F.14	Función ProgresoECGGUI	106
F.15	Función ProgresoTensionGUI	106
F.16	Función SPR	107
F.17	Función tensiometro	108

Bibliografía

[1] *Arduino - wikipedia, la enciclopedia libre*, https://es.wikipedia.org/wiki/Arduino, (Consultado en agosto, 2019).

[2] *Arduino uno - wikipedia, la enciclopedia libre*, https://es.wikipedia.org/wiki/Arduino_Uno, (Consultado en agosto, 2019).

[3] *Arduino uno r3, tutorial especificaciones electrónicas y programación.*, https://www.infootec.net/arduino/, (Consultado en agosto, 2019).

[4] *¿cómo funciona un tensiómetro? explicación | mitensiómetro.com*, https://mitensiometro.com/como-funciona-un-tensiometro/, (Consultado en agosto, 2019).

[5] *Cómo se hace la espirometría - pruebas médicas*, https://www.webconsultas.com/pruebas-medicas/como-se-hace-la-espirometria-13118, (Consultado en agosto, 2019).

[6] *El espirómetro | soplasopla*, https://soplasopla.com/2013/05/05/el-espirometro/, (Consultado en agosto, 2019).

[7] *Electrocardiógrafo - wikipedia, la enciclopedia libre*, https://es.wikipedia.org/wiki/Electrocardi%C3%B3grafo, (Consultado en agosto, 2019).

[8] *Esfigmomanómetro - wikipedia, la enciclopedia libre*, https://es.wikipedia.org/wiki/Esfigmoman%C3%B3metro, (Consultado en agosto, 2019).

[9] *fbbva_librocorazon_cap4.pdf*, https://www.fbbva.es/microsites/salud_cardio/mult/fbbva_libroCorazon_cap4.pdf, (Consultado en agosto, 2019).

[10] *IntroduccÍon*, http://www.ptolomeo.unam.mx:8080/xmlui/bitstream/handle/132.248.52.100/633/A6.pdf?sequence=6, (Consultado en agosto, 2019).

[11] *Mysignals hw – medical and ehealth development platform*, https://www.cooking-hacks.com/mysignals-hw-ehealth-medical-biometric-arduino-shield, (Consultado en agosto, 2019).

[12] *(pdf) evolución de la bioingeniería y la nanotecnología: desde la biofísica hasta la convergencia científico-tecnológica (nbic)*, https://www.researchgate.net/publication/320087425_Evolucion_de_la_bioingenieria_y_la_nanotecnologia_desde_la_biofisica_hasta_la_convergencia_cientifico-tecnologica_nbic, (Consultado en agosto, 2019).

[13] *Pruebas funcionales respiratorias*, https://www.clinicasubiza.com/es-es/enfermedades/pruebasdiagn%C3%B3sticas/pruebasfuncionalesrespiratorias.aspx, (Consultado en agosto, 2019).

[14] *Qué es un espirómetro y para qué sirve - el blog de girodmedical*, https://www.girodmedical.es/blog_es/espirometro-para-que-sirve/, (Consultado en agosto, 2019).

[15] *Ritmo cardiaco, análisis*, https://www.my-ekg.com/como-leer-ekg/ritmo-cardiaco.html, (Consultado en agosto, 2019).

[16] *Shields para arduino | aprendiendo arduino*, https://aprendiendoarduino.wordpress.com/2015/03/23/shields-para-arduino/, (Consultado en agosto, 2019).

[17] *Sistema de conducción cardiaco*, https://www.my-ekg.com/bases/sistema-conduccion.html, (Consultado en agosto, 2019).

[18] *Sonidos de korotkov - wikipedia, la enciclopedia libre*, https://es.wikipedia.org/wiki/Sonidos_de_Korotkov, (Consultado en agosto, 2019).

[19] MuyMedico, *Minimanual de ecg ilustrado*, 2019.