Pulsímetro manual y múltiple mediante un Arduino Uno

Edward Duarte, Andrés Ramírez, Nicolas Pedraza, David Orozco
Ingeniería Electrónica, Departamento de Ingeniería Pontificia Universidad Javeriana, Bogotá D.C, Colombia
edwardduarte@javeriana.edu.co
camilo_ramirez@javeriana.edu.co
pedraza_n@javeriana.edu.co
davidorozco@javeriana.edu.co

Abstract— The objective to carry out this project is to create a system which analyzes and delivers the heart pulse of our patient or subject to be treated. For its realization it was decided to carry it out through ARDUINO (it is a printed board with the necessary components for the microcontroller to work and its communication with a computer through serial communication).

The construction of the system was carried out by means of a simulator and then the implementation was carried out in order to have a guide with the adequate operation, for a better solution of the project. The code was also made and tested in both the simulator and the ADUINO environment.

To obtain the result of beats per minute, the patient must press a push-button (button) and, referring to this, the ARDUINO will print the result by means of the LED screen.

XIII. Introducción

El objetivo para realizar de este proyecto es crear un sistema el cual analice y entregue el pulso cardiaco a cualquier persona que lo necesite o quiera conocer su pulso. Para su realización se decidió llevarlo a cabo por medio de ARDUINO (es una placa impresa con los componentes necesarios para que funcione el microcontrolador y su comunicación con un ordenador a través de la comunicación serial).

Se realizo la construcción del sistema por medio de un simulador y luego se llevó acabo la implementación para así tener una guía con el funcionamiento adecuado, para una mejor solución del proyecto. También, se realizó el código y se probó tanto en el entorno del simulador como en el del ADUINO. Es necesario aclarar que las pulsaciones por minuto son mostradas en una pantalla de 7 segmentos y cuatro dígitos por medio del refresco dinámico.
Para el obtener el resultado de pulsaciones por minutos el paciente deberá presionar un pulsador (botón) y referente a ello el ARDUINO imprimirá el resultado por medio de la pantalla led.

XIV. Directrices para la preparación de los manuscritos

La elaboración de este proyecto fue asesorada e instruida por el ingeniero y docente de la Pontificia Universidad Javeriana, Juan Carlos Giraldo, ingeniero electrónico, profesor de planta de la Pontificia Universidad Javeriana de Bogotá, del departamento de ingeniería electrónica, facultad de ingeniería

XV. Objetivo general

Realizar por medio de ARDUINO un sistema que nos devuelva el pulso cardiaco que presenta el usuario.

XVI. Objetivos específicos

- Codificar en el entorno ARDUINO en lenguaje relacionado a C++ con adaptaciones a Arduino.
- Devolver el dato más preciso a el pulso cardiaco del usuario.
- Realizar manualmente la construcción correcta del sistema.
- Comparar los resultados de nuestro sistema con un pulsímetro comercial, con el objetivo de medir y comparar la precisión.

XVII. Algunos errores comunes

El uso de resistencias de muy baja denominación puede causar un cortocircuito en el sistema de manera que podemos no solo dañar la pantalla led si no también nuestra placa Arduino, de manera que es pertinente el uso de resistencias entre 1k a 10k ohmios de manera que pase corriente y se observe el led con una luminiscencia adecuada.

XVIII. CONTEXTO

El lenguaje ARDUINO permite la creación de distintos tipos de microordenadores, cada uno distinto del otro debido a la personalización y libertad de hardware que tiene, ya que cada usuario puede personalizar su propia placa para darle el uso que quiera.

A continuación, se presenta:

- Estado del arte
- Marco Teórico
- Diagrama de flujo(anexo).
- Diagrama de estados
- Descripción código.
- Funcionamiento notas en el código.
- Conexiones.
- Conclusiones participantes

XIX. ESTADO DEL ARTE

Actualmente se ha podido evidenciar en muchos países el problema de monitoreo en pacientes con frecuencias cardiacas altas, esto debido a la limitación de dispositivos para poder llevar un control en los pacientes. Es por esta razón que en el articulo "Analysis of Heart Rate and Body Temperature from the Wireless Monitoring System Using Arduino" publicado en 2019, por eBOOKS se plantea una solución para poder tener un control adecuado y poder abarcar más personas. La solución planteada en este artículo es por medio de Arduino crear varios monitores vía bluetooth de manera que se pueda tener el control del paciente y temperatura corporal en el lugar que se encuentre. [1]

XX. MARCO TEÓRICO

ARDUINO: Arduino es una plataforma de creación de electrónica de código abierto, la cual está basada en hardware y software libre, flexible y fácil de utilizar para los creadores y desarrolladores. Esta plataforma permite crear diferentes tipos de microordenadores de una sola placa a los que la comunidad de creadores puede darles diferentes tipos de uso [2].

Pulsador: es un dispositivo que permite desviar o interrumpir el curso de una corriente eléctrica [3].

Frecuencia o pulso cardiacos: es la frecuencia cardíaca, o sea el número de veces que el corazón late en un minute [4].

Transistor: tipo de dispositivo electrónico semiconductor, capaz de modificar una señal eléctrica de salida como respuesta a una de entrada, sirviendo como amplificador, conmutador, oscilador o rectificador de la misma [5].

LED: El acrónimo inglés LED (Light Emitting Diode) que en castellano significa literalmente "Diodo Emisor de Luz'' es un elemento capaz de recibir una corriente eléctrica moderada y emitir una radiación electromagnética transformada en luz. Coloquialmente es conocido como Diodo Luminoso [6].

Microcontrolador: es un circuito integrado que es el componente principal de una aplicación embebida. Es como una pequeña computadora que incluye sistemas para controlar elementos de entrada/salida. También incluye a un procesador y por supuesto memoria que puede guardar el programa y sus variables (flash y RAM) [7].

Pines PWM: (modulación por ancho o de pulso) es un tipo de señal de voltaje utilizada para enviar información o para modificar la cantidad de energía que se envía a una carga [8].

Resistencia: es una medida de la oposición al flujo de corriente en un circuito eléctrico. Se mide en ohmios, que se simbolizan con la letra griega omega (Ω). Se denominaron ohmios en honor a Georg Simon Ohm (1784-1854), un físico alemán que estudio la relación entre voltaje corriente y resistencia.[9]

Corriente: es la velocidad a la que un flujo de electrones pasa por un punto de un circuito eléctrico completo. En otras palabras, corriente=flujo.[10]

Voltaje: es la magnitud que da cuenta de la diferencia en el potencial eléctrico entre dos puntos determinados. También llamado diferencia de potencial o tensión eléctrica. [11]

XXI. Diagrama de flujos

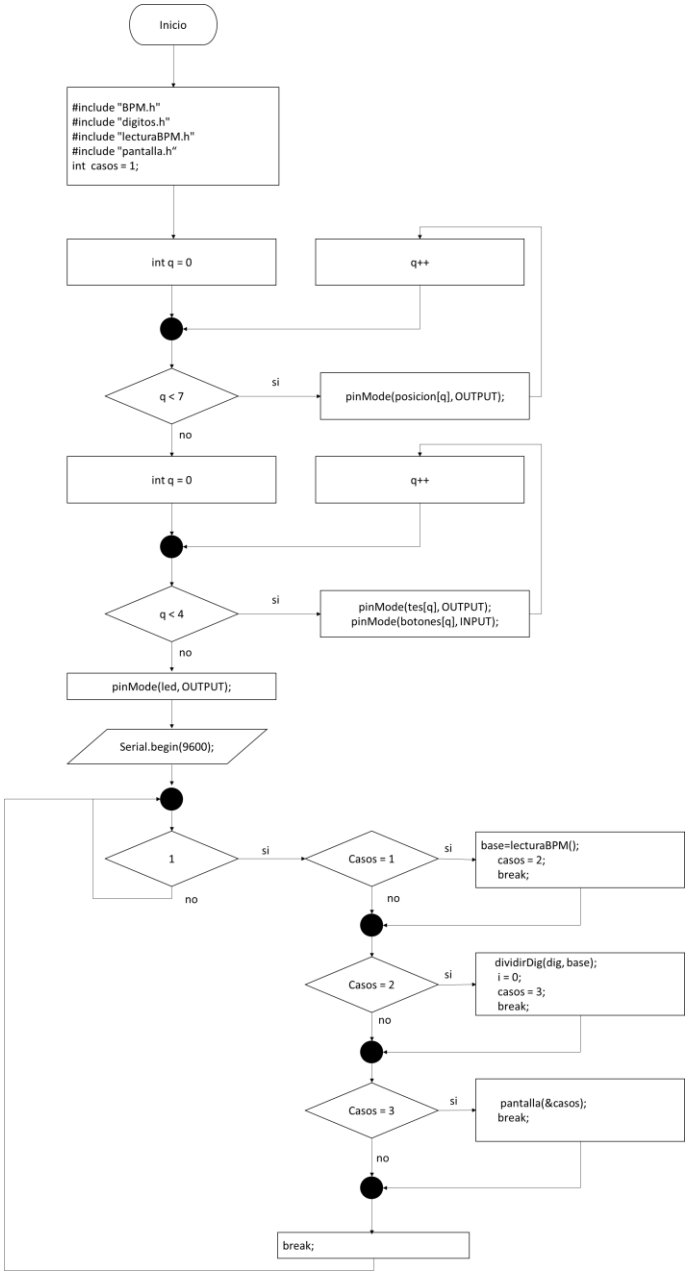


Fig. 1. Diagrama de flujo

XXII. Comportamiento de los elementos

Era fundamental para el proyecto determinar el comportamiento y la naturaleza del display 7 segmentos, para ello y debido a que no se tenía la datasheet del elemento se utilizó un multímetro para probar y determinar si era cátodo o ánodo común.

Para ello se pone el modo de continuidad en el multímetro, este permitirá encender los segmentos del display cuando se conecta uno al positivo y otro al negativo, el orden lo determina específicamente el tipo de display, si es ánodo común al mantener constante el negativo y desplazar el positivo cambiara los segmentos de cada digito, mientras si es cátodo común solamente cambiara los segmento de un único digito, con esto en mente se experimento con cada uno de los pines y se genero un listado de los pines que activan cada uno de los segmentos.

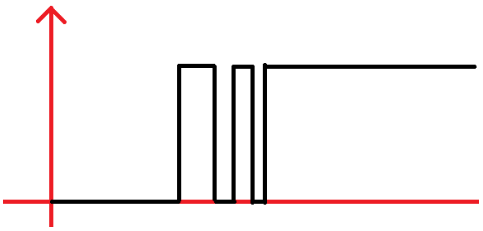
Debido a que era un cátodo común se utilizaron transistores NPN para su funcionamiento.

XXIII. Descripción código

LecturaBPM.h

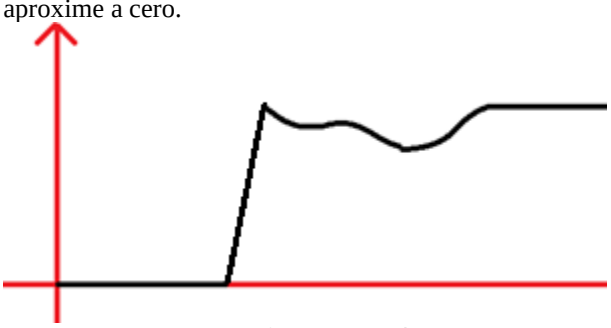
Esta es la función que se encarga de separar y ejecutar las lecturas de los 4 pulsadores, además se encarga del filtrado del rebote, para ello toca considerar que debido a que no se tenía un Arduino mega, se utilizó un Arduino uno, esto genero unos escases de pines digitales, por lo cual se utilizó pines analógicos.

Para el funcionamiento del código toca considerar la diferencia entre el rebote en un pin digital y un pin analógico, principalmente los pines digitales al solo poder tener 2 estados el rebote es expresa como mini pulsos después de ejecutar la acción.



Graf. 1. Señal Digital.

Mientras que en un pin analógico debido a que este puede tomar una gran variedad de valores diferentes, el rebote solamente fluctúa debajo del uno, esto genera que el rebote nunca se aproxime a cero.



Graf. 2. Señal Analógico

Se usa este efecto para usar el filtro, el código solamente tomara el pulso como valido cuando este tenga desde antes valores en cero, eso quiere decir que el solamente tomara el primer pulso de la señal e ignorara los demás.

Cuando la señal llegue a cero, permitirá tomar un nuevo pulso, esto genera que se puede seleccionar solamente los principios de los pulsos.

BPM.h

Esta función determina las pulsaciones por minuto después que la señal ya fue filtrada.

En primera instancia se tienen 3 variables, PrimeraPulsacionT, SegundaPulsacionT, intervaloPulsaciones.

El programa comienza tomando el tiempo cuando detecta el primer pulso, almacena el tiempo en la variable PrimeraPulsacionT, después detecta el segundo pulso y almacena el tiempo en SegundaPulsacionT, para proceder a calcular el tiempo entre los pulsos y almacenarlo en intervaloPulsaciones, para continuar tomando muestras se asigna el valor de SegundaPulsacionT en PrimeraPulsacionT y toma el siguiente pulso en SegundaPulsacionT, con ello permite continuamente tomar la diferencia de los pulsos. Para finalizar se realiza el calculo para determinar las pulsaciones por minuto y se retorna el valor.

dígitos.h

para determinar cada uno de los dígitos del valor de pulsaciones la función dígitos.h separa cada uno de estos en un arreglo de 4 elementos, para ello se guardará en el arreglo el cociente del número con 10 potencia a su posición inicializando en cero y el sustraendo de los valores previos producto con 10 potencia de su posición, también inicializada en cero.

```
dig[0] = base/1000;
dig[1] = (base/100)-(dig[0]*10);
dig[2] = (base/10)- (dig[0]*100) - (dig[1]*10);
dig[3] = base- (dig[0]*1000) - (dig[1]*100) - (dig[2]*10);
```

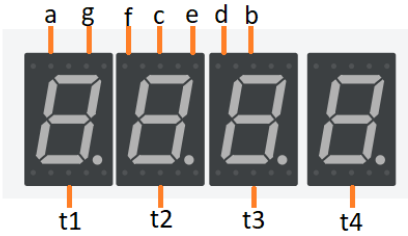
Imag. 1. código digitos.h

Pantalla.h:

Esta función es la encargada de la impresión en el display del número seleccionado.

Para el diseño del proyecto se utilizó un display de 7 segmentos, 4 dígitos con 24 pines de cátodo común, pero gran parte de ellos no serán utilizado debido a que no son de utilidad.

el display 7 segmentos tiene un total de 11 pines útiles.



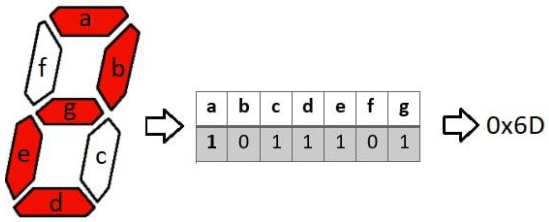
Imag. 2. Repartición pines para código pantalla.h

Los conectores superiores permiten la activación de los segmentos de un mismo digito, mientras que los inferiores permiten seleccionar cual digito se quiere activar, el inconveniente es que debido a que no se puede seleccionar que segmentos individuales se requiere activar para cada digito, es imposible mostrar valores diferentes con una señal continua.

Para solucionar el problema se realizará un efecto visual, cuando una fuente emisora de luz se enciende y se apaga a una frecuencia muy alta, el ojo humano no es capaz de percibir el cambio.

En base a ello se diseña el código para generar este efecto visual e imprimir diferentes números en el display de 7 segmentos.

En primera instancia se realiza un arreglo de 10 elementos que contengan los bits que se necesitan para formar el numero en el display, se almacenaran en forma de un numero hexadecimal y se utilizaran cuando se quiera conocer los segmentos de un número.



Imag. 3. Generación de los números en Hexa

Con esto en cuenta, la función realiza un ciclo por cada uno de los segmentos y de los dígitos, donde se evaluará si el valor extraído tiene ese segmento en su imagen, para ello se utiliza la función bitRead que permite obtener el valor de un bit en específico de cualquier valor, se selecciona los segmentos que cumplen con la condición y se activan cuando se va a imprimir el segmento.

Para una mejor visualización se utilizó el numero 0478 con un delay para observar el ciclo.

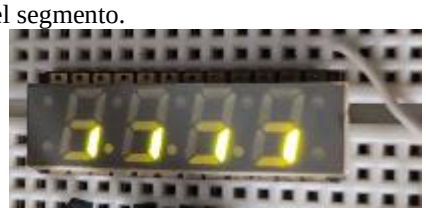
Iniciando el ciclo se comienza con el segmento a, el único digito que no lo contiene, es el número 4, por lo tanto, el segundo digito no estará activado.

Imag. 4. Display a frecuencia alta (segmentos a)

Después desactiva el segmento a y activa el b, y observamos que todos los elementos contienen el b, lo que lleva a que todos estén activos.

Imag. 5. Display a frecuencia alta (segmentos b)

El siguiente es el segmento c, que, como el b, todos los dígitos contienen el segmento.



Imag. 6. Display a frecuencia alta (segmentos c)

En el segmento d, el digito 7 y 4 no lo contienen, por lo tanto, solamente los extremos se iluminan

Imag. 7. Display a frecuencia alta (segmentos d)

Continuando con el segmento e, solamente los dígitos 0 y 8 lo contienen, lo que conlleva que se comporte de una manera similar al segmento d.

Imag. 8. Display a frecuencia alta (segmentos e)

El segmento f, el único digito que no lo contiene es el 7, por lo tanto, las decenas no estarán activas.

Imag. 9. Display a frecuencia alta (segmentos f)

Finalmente, en el segmento g solamente están activos el 4 y 8 lo que excluye los dígitos 0 y 7.

Imag. 10. Display a frecuencia alta (segmentos g)

Cuando se repite este ciclo con una frecuencia muy alta, no se percibe el cambio de los segmentos, generando que se perciba el número 0478 en el display.

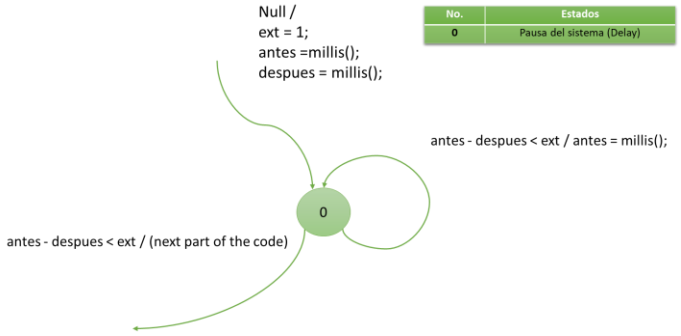
Imag. 11. Display a frecuencia alta

Utilizando esto se permite mostrar cualquier numero de 4 dígitos en el display.

<u>delayA.h:</u>

Esta función es la maquina de estados finitos que permite hacer un delay sin interrumpir el ciclo, en primera instancia se manejan 2 variables una almacenara el tiempo de inicio, y la otra el tiempo actual, cuando la diferencia del tiempo sea la estipulada se saldrá del ciclo y continuara con la operación.

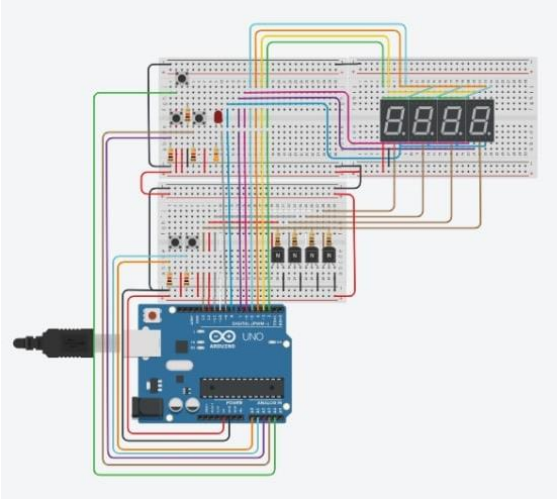
Además, se realizo el siguiente diagrama de estados finitos con el fin de entender a cabalidad el funcionamiento de la función delay.h, donde básicamente se hace uso de un while (1) y un switch case en referencia a los estados finitos como se resaltó anteriormente.



Imag. 12.1. Diagrama de estados finitos (delay)

XXIV. MONTAJE DEL PROYECTO

El resultado de la aplicación de los conceptos a nivel de hardware es el siguiente:



Imag. 12.2. Diseño en Tinkercad

Allí, se puede observar la solución planteada a detalle para el correcto funcionamiento del proyecto. Cabe destacar los 4 botones para 4 usuario simultáneos y uno para pasar por la información de las pulsaciones de cada uno de los usuarios en pantalla.

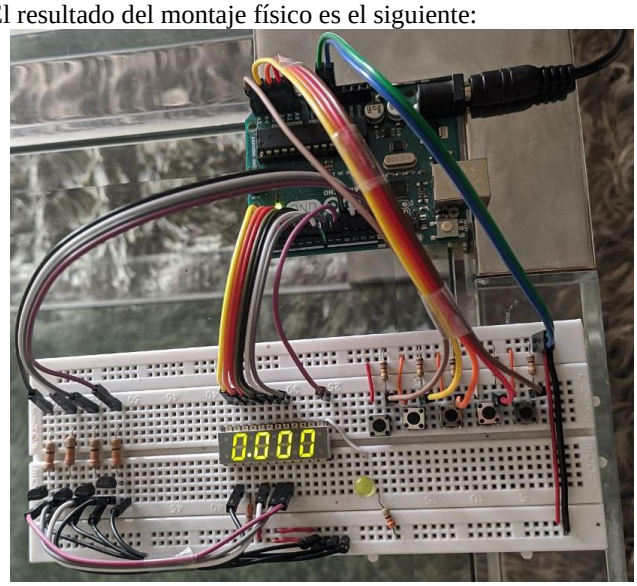
De esta manera, se procede a buscar la datasheet de la pantalla en cuestión y determinar si es de ánodo o cátodo común. Para esta práctica, fue imposible conseguiré la datasheet del producto. Por lo cual fue necesario hacer mediciones de pruebas con un multímetro en su función de continuidad y poder determinar las funciones de cada pin.

De esta manera se desarrollo la siguiente tabla que explica como están distribuidos los pines para esta pantalla.

	1	2	3	4	5	6	7	8	9	10	11	12
1	-	a	g	f	c	e	d	b		:		.
2	-	a...g	.	a...g	a...g		a...g	a...g	:		.	.

Imag. 12.3. Tablas de pines en pantalla.

El resultado del montaje físico es el siguiente:



Imag. 13. Sistema realizado a mano en Arduino

De esta manera se obtiene el producto con el que se pueden encontrar los resultados a continuación.

XXV. Resultados

Para este proyecto primeramente se debe conocer si el display es de cátodo común o ánodo para así realizar todo el proceso respecto a esto, de manera que mediante un multímetro se procede a realizar un análisis mediante las terminales del display. Luego de realizarse dicho proceso se reconoce que el display es de cátodo común (Imagen 1) ya que los leds se encuentran unidos a la terminal negativa.

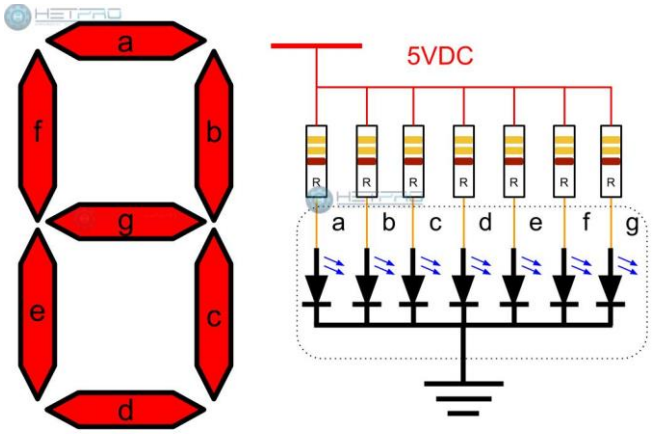
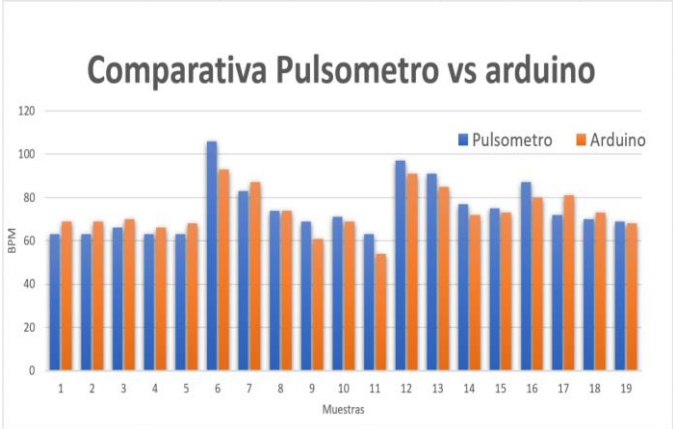


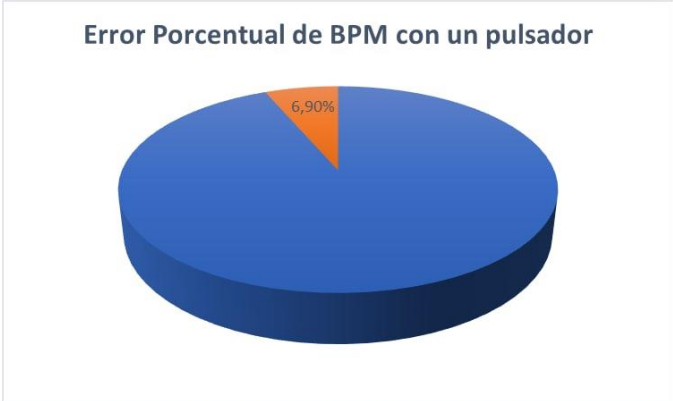
Imagen 14. Cátodo común.

Luego, se decide realizar varias pruebas con el pulsómetro creado mediante Arduino, de manera que sus datos obtenidos sean comparados al mismo tiempo con un pulsómetro comercial así para conocer qué tan acercado están los valores del dispositivo Arduino, de esta manera se puede evidenciar en el grafico 1 esta comparativa. Cabe aclarar que para una mejor precisión de las pulsaciones que arroja el proyecto se hizo uso de un fonendoscopio para disminuir el margen de error aleatorio.



Graf. 3. Comparativa pulsómetro vs Arduino.

De dicha grafica se puede analizar que los valores comparados con los valores del pulsómetro no presentan una gran diferencia, de manera que se puede calcular que tan alejados estamos de estos valores, para ello se realiza un análisis del error porcentual el cual se observa en el grafico 2.



Graf. 4. Error porcentual con un Pulsador.

De esta manera se observa que los datos obtenidos por el Arduino solo discrepan en un 6.9% a los de un pulsómetro comercial.

Además, para una mejor calidad en el diseño, se realizó una comparativa con diferentes frecuencias para percibir los números en el display, de manera que se pudiera observar de manera clara los números y con la luminiscencia adecuada. Para ello se cambió la frecuencia generando así que en algunos intentos los números no se alcanzaran ni a detallar, como en la imagen 2.

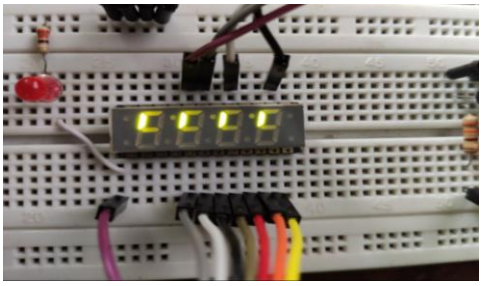


Imagen 14. Frecuencia baja.

Luego de varios intentos se llegó a un resultado muy satisfactorio en el cual la luminiscencia del display era la esperada y su frecuencia permitía percibir los números de manera correcta, como se muestra en la imagen 3.

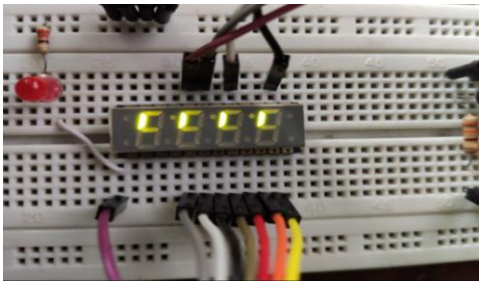


Imagen 3. Frecuencia alta.

Por ello se puede ver que entre más alta la frecuencia mejor se puede percibir el número.

XXVI. Análisis

Al empezar el proyecto se encontró con el primer problema a solucionar, este consistía en conocer si el display de 7 segmentos con el cual se va a trabajar es de ánodo común (Terminal Positiva) o cátodo común (Terminal Negativa) esto para así emparejarlo con los transistores correctos para un correcto funcionamiento del sistema, así como sus respectivas conexiones. Al ser cátodo común el display se optó por trabajar con transistores npn. Con los transistores y el display 7 segmentos ya bien conectados, se procede a poner los pulsadores para asi plantear como tomar los pulsos del usuario, de manera que toca decidir entre tomar señales digitales o analógicas del Arduino dadas por el pulsador. De esta manera se cae en cuenta que es más efectivo el trabajo con señales analógicas, ya que en las digitales el efecto rebote nos causa varios pulsos, que a la hora de analizarlos en el código nos da errores, por el contrario, al usar la señal analógica esta su rebote no llega a cero de manera que al analizarla en el código solo tomamos cuando este pulso acaba por lo cual con esta señal no se genera ningún error y es más efectiva. Luego de obtener estos dos resultados, se procede a codificarlo todo para que se vea reflejado estos análisis en el respectivo display, es por ello por lo que en el código se tuvo en cuenta la inclusión de un delay el cual se generó por medio de estados finitos, esto para dar un tiempo de análisis en los pulsos.

XXVII. Conclusión

- El uso del Arduino uno fue una opción viable pero un poco limitada debido a su cantidad de puertos, el uso de un Arduino mega nos ayudaría a expandir el sistema y mejorarlo.
- El uso de frecuencias altas permite una mejor visualización de los números en el display, de igual manera el control de la luminiscencia para que el led no se funda es equivalente a resistencias altas.
- Debido a que el propio pulso se lo está realizando el usuario se genera una incertidumbre y por lo cual un error, aun así, el error que se presenta en nuestro pulsómetro mediante Arduino en bajo comparado con los datos de un pulsímetro comercial.
- El uso de señales analógicas facilito el trabajo debido a que el efecto rebote no llega a 0, esto permitiendo analizar que la señal se acaba solo cuando llega a 0, por lo cual no genera varios pulsos que interfieran con la toma de datos.

XXVIII. Código del proyecto

Este proyecto es de código abierto y cualquier persona puede acceder a él. A continuación, se adjunta el enlace que contiene la totalidad de códigos.

https://livejaverianaedu-my.sharepoint.com/:u:/g/personal/edwardduarte_javeriana_edu_co/Ec2hCCiRKqdBg2ffQHccCIABdXEijG--yoFM7oM64RJvXA?e=N3vfNW

XXIX. REFERENCIAS

[1] I. "Analysis of Heart Rate and Body Temperature from the Wireless Monitoring System Using Arduino", eBOOKS, 2021. [Online]. Available: https://www.researchgate.net/publication/337116465_Analysis_of_Heart_Rate_and_Body_Temperature_from_the_Wireless_Monitoring_System_Using_Arduino [Accessed: 28- May- 2021].

[2] "¿Que es la programacion Arduino y para que sirve?, BeJOB, 2021 [Online]. Available: https://www.bejob.com/que-es-la-programacion-con-arduino-y-para-que-sirve/ [Accessed: 28- May- 2021].

[3] I. "¿Qué son y para que sirven los pulsadores eléctricos?", SyZ COMINSA,2021. [Online]. Available: https://syzcominsa.pe/blog/que-son-y-para-que-sirven-los-pulsadores-electricos#:~:text=Un%20interruptor%20o%20pulsador%20el%C3%A9ctrico,curso%20de%20una%20corriente%20el%C3%A9ctrica. [Accessed: 28- May- 2021].

[4] "¿Cómo tomarte el pulso? ", MAYO CLINIC, 2021. [Online]. Available: https://www.mayoclinic.org/es-es/how-to-take-pulse/art-20482581#:~:text=El%20pulso%20es%20la%20frecuencia,arteria%20car%C3%B3tida%20en%20el%20cuello. [Accessed: 28- May- 2021].

[5] "Concepto de Transistor", concepto *de*, 2021. [Online]. Available: https://concepto.de/transistor/ [Accessed: 28- May- 2021].

[6] "LUCES LED ", Helloauto, 2021. [Online]. Available: https://helloauto.com/glosario/luces-led [Accessed: 28- May- 2021].

[7] "Microcontrolador ¿Qué es y para que sirve?",Hetpro, 2021 [Online].Available:https://hetpro-store.com/TUTORIALES/microcontrolador/ [Accessed: 28- May- 2021].

[8] "Salidas PWM Arduino", *Programo Ergo sum* 2021. [Online]. Available: https://www.programoergosum.com/cursos-online/arduino/255-salidas-analogicas-pwm-con-arduino/salidas-analogicas-pwm#:~:text=Salidas%20PWM%20en%20Arduino,se%20env%C3%ADa%20a%20una%20carga. [Accessed: 28- May- 2021].

[9] "¿Que es la Resistencia?", *fluke*, 2021. [Online]. Available: https://www.fluke.com/es-co/informacion/blog/electrica/que-es-la-resistencia [Accessed: 28- May- 2021].

[10] "¿Que es la Corriente?", *fluke*, 2021. [Online]. Available: https://www.fluke.com/es-co/informacion/blog/electrica/que-es-la-corriente [Accessed: 28- May- 2021].

[11] "Concepto de Voltaje", *concepto.de*, 2021. [Online]. Available: https://concepto.de/voltaje/ [Accessed: 28- May- 2021].

XXX. ANEXOS

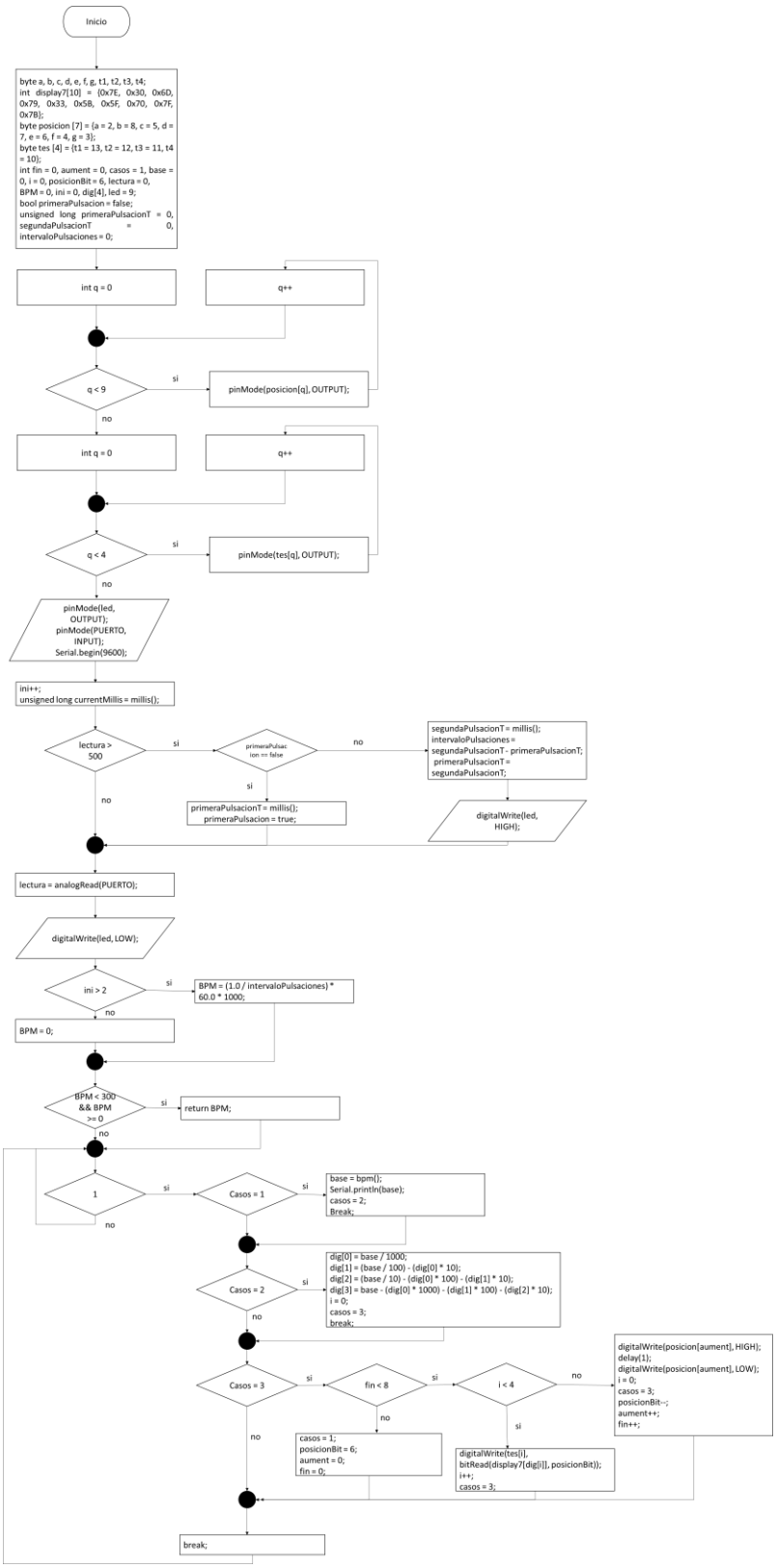

Inicio
byte a, b, c, d, e, f, g, t1, t2, t3, t4;
int display7[10] = {0x7E, 0x30, 0x6D, 0x79, 0x33, 0x5B, 0x5F, 0x70, 0x7F, 0x7B};
byte posicion [7] = {a = 2, b = 8, c = 5, d = 7, e = 6, f = 4, g = 3};
byte tes [4] = {t1 = 13, t2 = 12, t3 = 11, t4 = 10};
int fin = 0, aument = 0, casos = 1, base = 0, i = 0, posicionBit = 6, lectura = 0,
BPM = 0, ini = 0, dig[4], led = 9;
bool primeraPulsacion = false;
unsigned long primeraPulsacionT = 0,
segundaPulsacionT = 0,
intervaloPulsaciones = 0;
int q = 0
q++
q < 9
si
pinMode(posicion[q], OUTPUT);
no
int q = 0
q++
q < 4
si
pinMode(tes[q], OUTPUT);
no
pinMode(led, OUTPUT);
pinMode(PUERTO, INPUT);
Serial.begin(9600);
ini++;
unsigned long currentMillis = millis();
lectura > 500
si
primeraPulsacion == false
no
segundaPulsacionT = millis();
intervaloPulsaciones = segundaPulsacionT - primeraPulsacionT;
primeraPulsacionT = segundaPulsacionT;
si
primeraPulsacionT = millis();
primeraPulsacion = true;
digitalWrite(led, HIGH);
no
lectura = analogRead(PUERTO);
digitalWrite(led, LOW);
ini > 2
si
BPM = (1.0 / intervaloPulsaciones) * 60.0 * 1000;
no
BPM = 0;
BPM < 300 && BPM >= 0
si
return BPM;
no
1
si
Casos = 1
si
base = bpm();
Serial.println(base);
casos = 2;
Break;
no
Casos = 2
si
dig[0] = base / 1000;
dig[1] = (base / 100) - (dig[0] * 10);
dig[2] = (base / 10) - (dig[0] * 100) - (dig[1] * 10);
dig[3] = base - (dig[0] * 1000) - (dig[1] * 100) - (dig[2] * 10);
i = 0;
casos = 3;
break;
no
Casos = 3
si
fin < 8
si
i < 4
no
digitalWrite(posicion[aument], HIGH);
delay(1);
digitalWrite(posicion[aument], LOW);
i = 0;
casos = 3;
posicionBit--;
aument++;
fin++;
no
casos = 1;
posicionBit = 6;
aument = 0;
fin = 0;
si
digitalWrite(tes[i], bitRead(display7[dig[i]], posicionBit));
i++;
casos = 3;
no
break;

Fig. 1. Diagrama de flujo sin funciones

XXXI. DIAGRAMA DE ESTADOS FINITOS

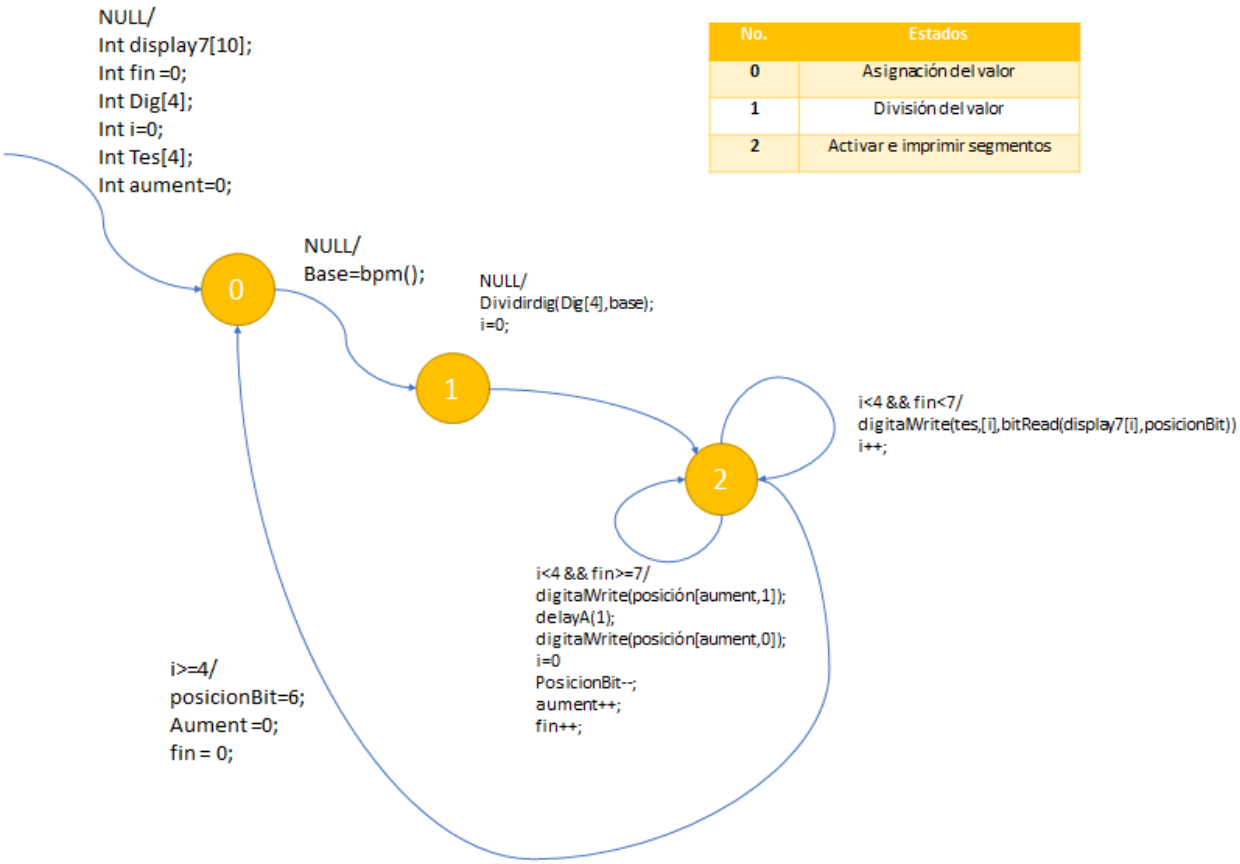


Fig. 3. Diagrama de estados finitos.