Universidad de Puerto Rico

Recinto Universitario de Mayagüez

Operación: Pancho

Manual de Instrucciones

Autores: Hernán Miranda, Karina Martínez, María Jiménez, Lianne Sánchez

Mentora: Nayda Santiago

Tabla de Contenidos: **Pagina:**

1. Sinopsis 3
2. Introducción 4
3. Objetivos 5
4. Instrucciones del Juego 5
5. Materiales 6-7
6. Tabla de Conexiones 8
7. Metodología
    a. Hardware 9-13
    b. Software 14-17
8. Preguntas frecuentes 18
9. Informe 14

Sinopsis

Debido a los avances en las áreas de Ciencias, Ingeniería, Tecnología y Matemáticas (STEM, por sus siglas en inglés), un grupo de estudiantes se unió para crear un programa de alcance llamado *Tech. Carnival*. El objetivo del *Tech. Carvinal* es motivar a los estudiantes de bajos recursos a cursar estudios postsecundarios en las áreas de STEM, específicamente ingeniera eléctrica e ingeniería de computadoras. Este programa reta a los estudiantes a crear unos módulos diseñados con sistemas embebidos, para que los estudiantes puedan armarlo por su cuenta y de esta manera proveerles conocimientos básicos de circuitos y programación para que lo puedan utilizar en un futuro. De esta manera se busca incrementar el número de estudiantes que apliquen a la universidad y darles la motivación a ejercer carreras en las áreas de STEM.

Introducción:

El programa de alcance The Tech. Carnival está dirigido a orientar estudiantes de escuelas públicas sobre las carreras de ingeniería eléctrica y computadoras. La mayoría de las escuelas en Puerto Rico son públicas, sin embargo, los estudiantes de estas escuelas representan la minoría en alcanzar estudios profesionales de ingeniera en la Universidad de Puerto Rico Recinto de Mayagüez [1]. The Tech. Carnival enseñará por medio de módulos interactivos, manuales y talleres, tanto a estudiantes como a maestros. Estos talleres serán dentro y fuera de horas de clase. De esta manera deseamos motivar a los estudiantes a obtener una carrera en la ingeniería mostrándole los conceptos básicos de circuitos y programación. Luego de cada taller, el estudiante podrá evaluar su experiencia y el conocimiento aprendido antes y después de cada taller.* El equipo de trabajo de Tech. Carnival tomará los resultados para mejorar y medir cuan efectivo son los talleres de orientación.

Objetivos:

- Ensamblar módulo de manera que se puedan implementar todos los componentes eléctricos.
- Construir el circuito y hacer las conexiones pertinentes al Arduino UNO.
- Aprender la lógica del juego e implementar un código para cada función del circuito.
- Entender las variaciones de cada componente del circuito utilizando la plataforma

I. ¿Qué es el Operation Pancho?

El Operation Pancho es un juego que consiste en "operar" a un paciente, removiendo los órganos de su cuerpo sin tocar los bordes en los agujeros, ya que están electrificados. Si el jugador toca el borde con la pinza, pierde el juego ya que las pinzas envían una señal digital al Arduino de que el circuito se ha cerrado iluminando la nariz de Pancho y haciendo un sonido de alerta. El reto consiste en remover la mayor cantidad de órganos, en menos de 60 segundos.

II. Dimensiones:

Se puede utilizar materiales reciclables como cartón o algún tipo de plástico. Usted debe hacer orificios de acuerdo al tamaño de la caja que utilice. Por ejemplo:

III.

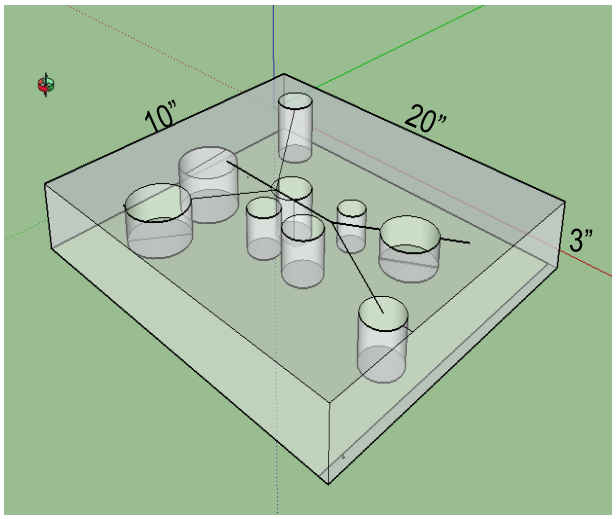
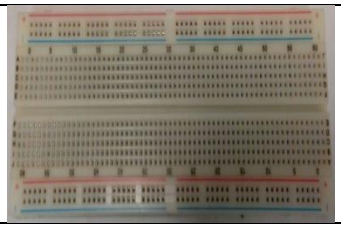
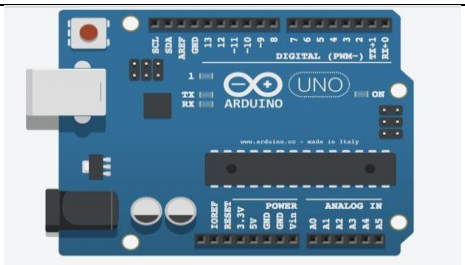
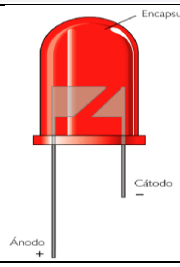


Figura 1 - Dimensiones

Materiales

Materiales	Descripción	Imagen
Cinta Conductiva	Une los bordes de los agujeros, y conduce electricidad de manera que cuando la pinza toque la cinta cierre el circuito.	
Pinzas	Utilizadas para sacar los órganos.	
Tablero de Circuitos	Para conectar los cables y formar el circuito.	
Cable de corriente	Este cable se conecta del Arduino a la Computadora.	
Arduino UNO	Microcontrolador	
Luces pequeñas LED	Utilizada para la nariz de Pancho, nos avisa cuando el borde de algún agujero hace contacto	
IC2 7-segment Display	Cronometro	

Piezo Element con cables eléctricos	Hace un sonido cuando la pinza hace contacto con los bordes del agujero.	
Botón de opresión	Para iniciar el juego	
Cables eléctricos	Conectan el circuito.	
Resistor 10 kOhm	Se utiliza para la LED, ya que disminuye el flujo de corriente y evita que la LED se sobrecargue y se queme.	
Cinta adhesiva negra de electricistas	Para unir los cables.	

a. Materiales eléctricos son accesibles en las tiendas Adafruit y Sparkfun.
Total: $93.50

IV. **Tabla de Conexiones:**

Puertos del Arduino	Aparatos Conectados
3.3V	Botón
5V	Pantalla *7-Segment*
GND	*7-Segment,* Bombilla LED, Bocina
A4	Pantalla *7-Segment*
A5	Pantalla *7-Segment*
10	Pinzas, Bombilla LED
11	Bocina (lado positivo +)
13	Botón momentáneo

IV. Instrucciones Paso a Paso

1. Primero conecte la bocina. El lado positivo al puerto 11 y el negativo no lo conecte. La bocina será nuestra alerta cuando el jugador toque los bordes. La bocina tendrá un signo de (+) indicando que ese es el lado positivo utilice un cable de color rojo para conectarlo.

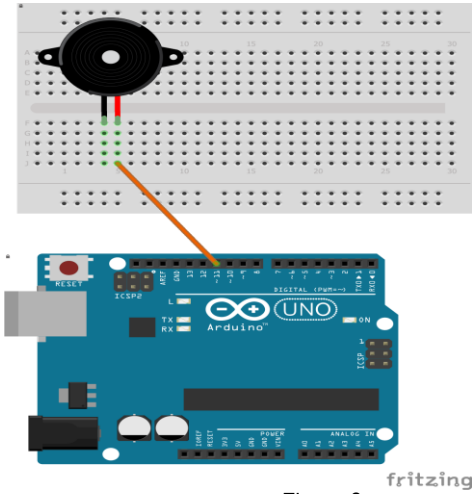


Figura 2

2. Ahora conectaremos la nariz de Pancho, para esto utilizaremos una LED, conecte con un cable a el ánodo de la LED en el puerto 10, también conecte las pinzas en la conexión a tierra (*ground)* del tablero de circuitos. El cátodo no lo conecte. El *ánodo es la parte más larga de la LED*, además es su lado positivo y el cátodo el negativo.

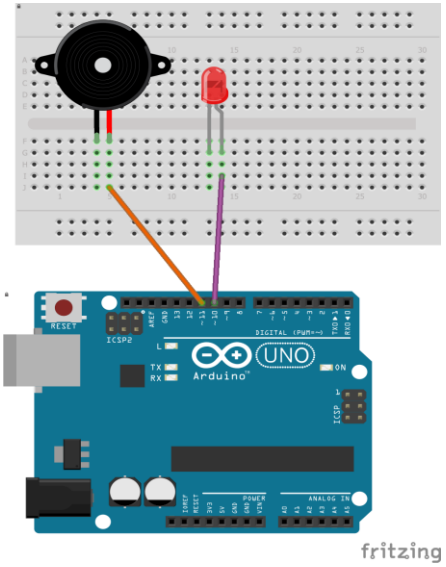


Figura 3

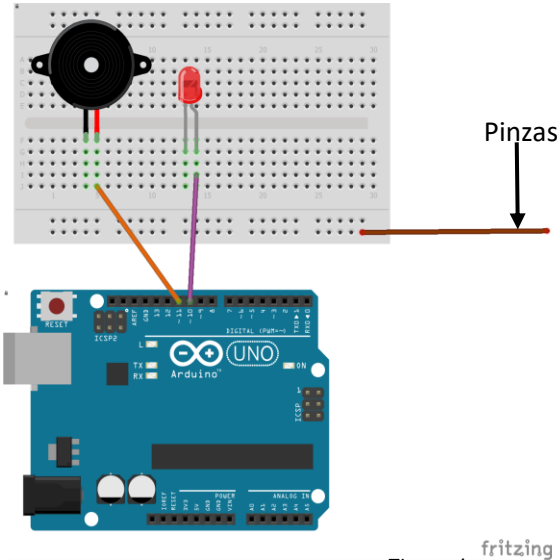


Figura 4

3. Conecte otro cable al puerto 13, en esta conexión se añadirá un resistor como se muestra en la imagen. Luego conecte un cable del botón el área designada a voltaje en el tablero. El botón puede ser conectado de cualquiera de sus partes, pero conecte un cable a la vez. La función de el botón es darle comienzo al juego luego de oprimirlo, pero eso lo veremos más adelante en el área de *Software*.

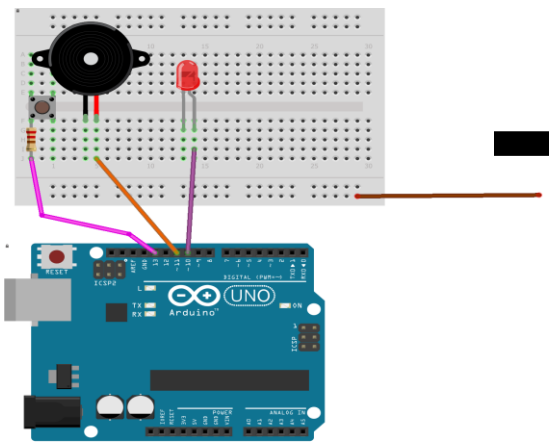


Figura 5

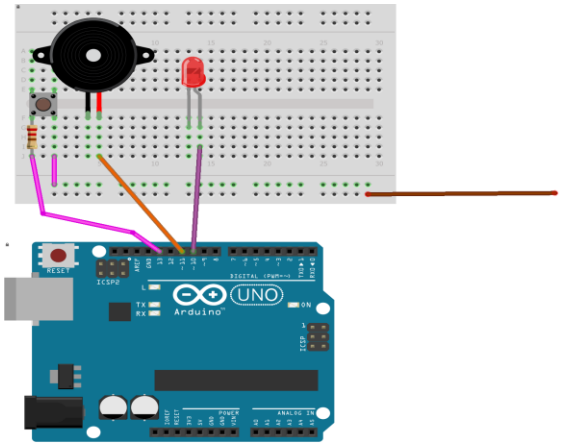


Figura 6

4. Ahora conectaremos nuestro cronometro, conecte el “7-Segment” de la siguiente manera. Haga una conexión desde el puerto C de la pantalla, al puerto A5 del Arduino.

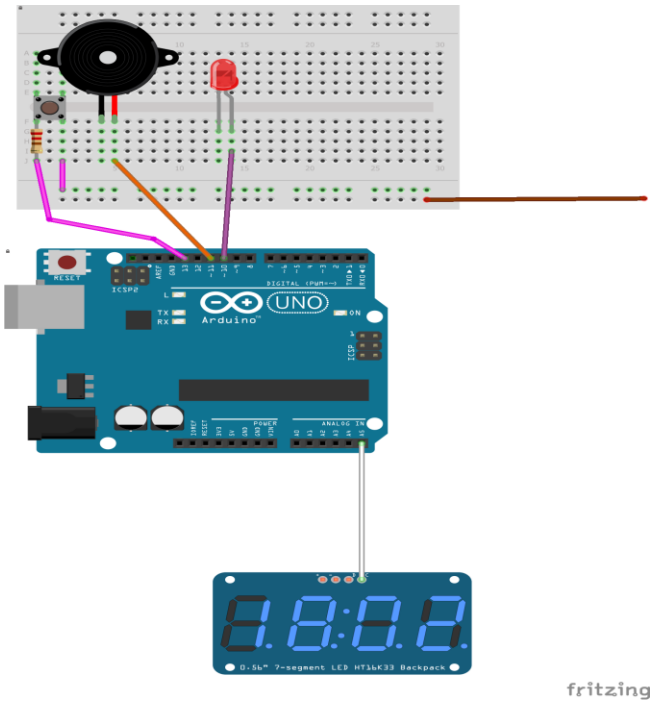


Figura 7

Luego conecte el puerto D de la pantalla al puerto A4 del Arduino, como se muestra a continuación:

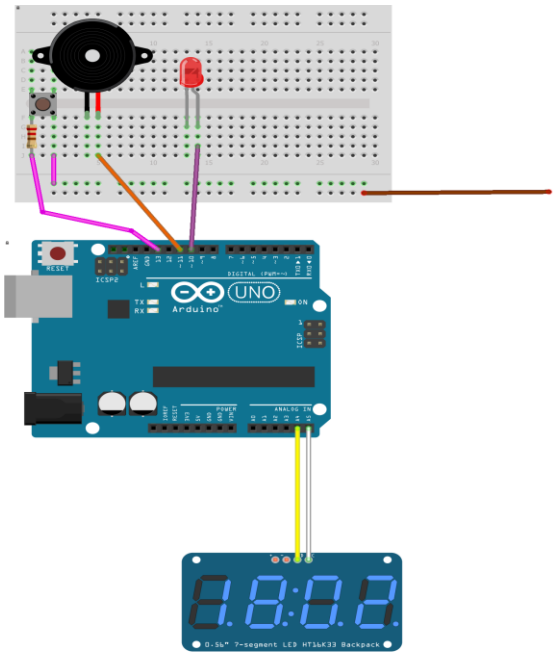


Figura 8

Ahora conecte los cables de voltaje (rojo) y *ground,* el cable rojo va de los 5V de la pantalla, a el puerto de 5V del Arduino. El cable negro, va desde el puerto de *ground* de la pantalla, a la conexión a tierra del Arduino.

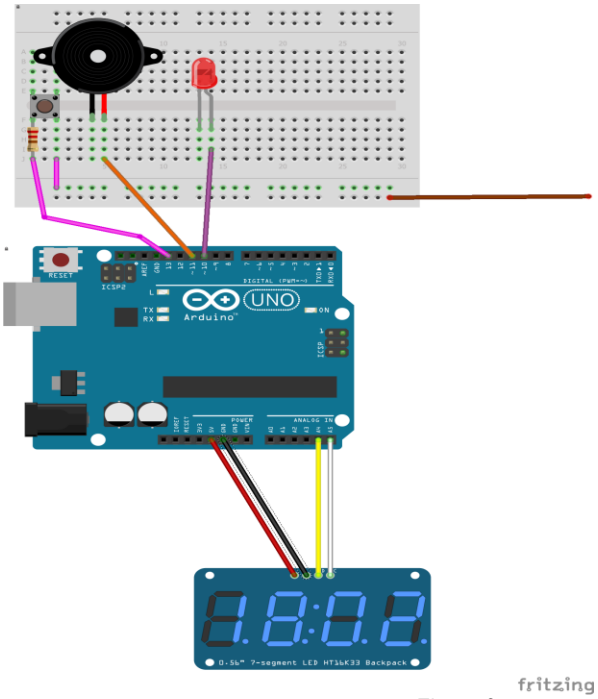


Figura 9

5. Ahora haremos todas las conexiones a tierra. Conecte el cátodo de la LED, a la conexión a tierra del tablero, de igual manera, haga la conexión para la bocina.

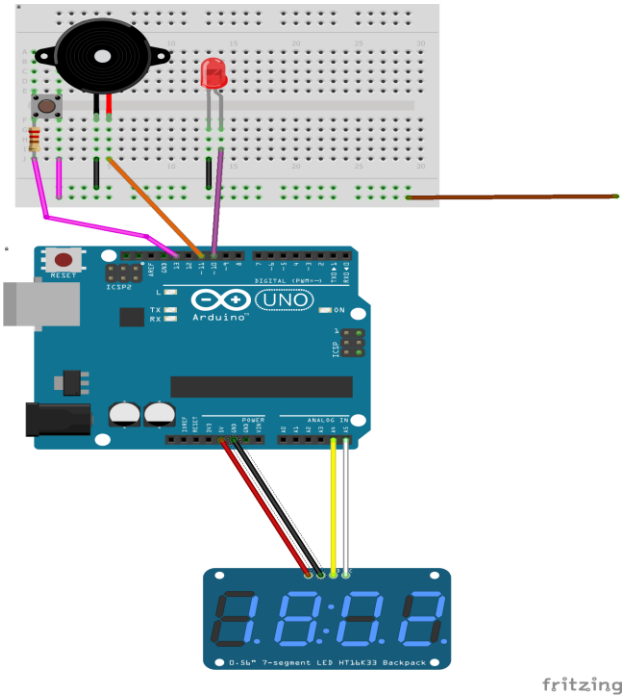


Figura 10

6. Ahora conecte el volteje desde el Arduino, conecte un cable desde el puerto 3.3V del Arduino a el tablero de circuitos, y la conexión a tierra (cable negro).

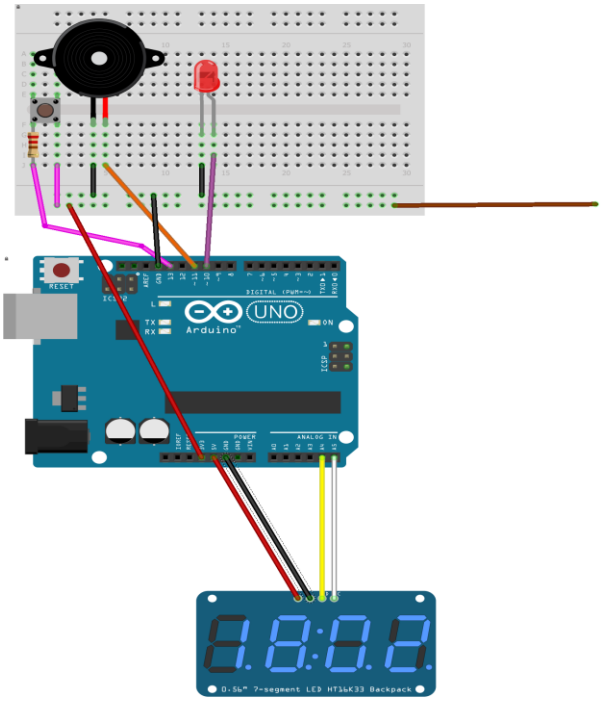


Figura 11 – Circuito Final

7. Divida la tapa de la caja en dos partes: superior e inferior y en la parte superior, dibuje a Pancho, luego haga los agujeros para los órganos como se ilustra en la fingura.

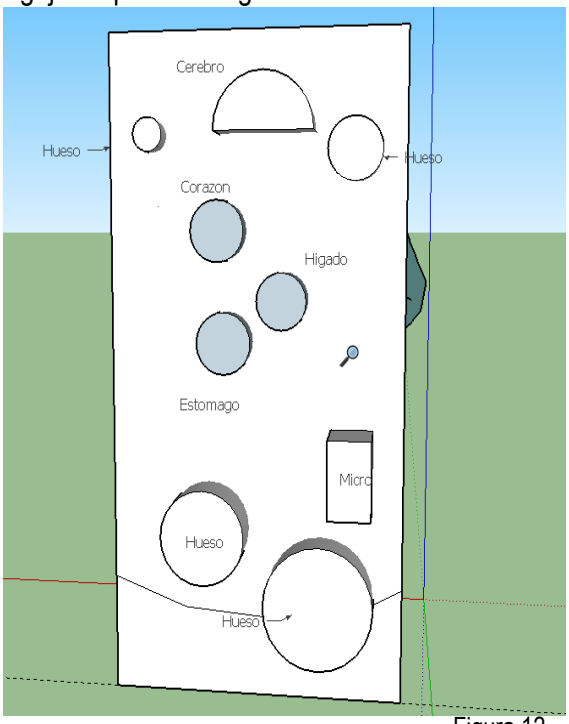


Figura 12

Nota: Pegue la cinta conductiva de cobre alrededor de los agujeros del dibujo. Asegúrese de que este bien pegado a todo el borde y de que todos se conecten entre sí. Este será nuestro sensor que detectará cada vez que el jugador toque un borde.

8. En la caja, haga los espacios para colocar cada uno de los órganos y huesos.

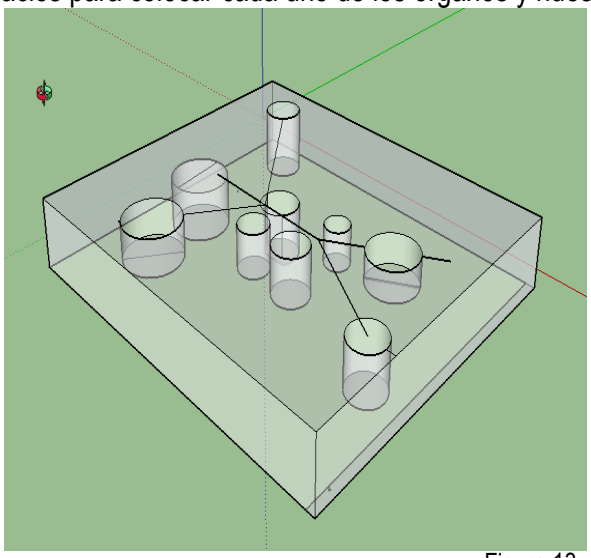


Figura 13

9. De no presentar ningún problema, conecte el Arduino a una fuente de energía (por ejemplo: computadora, batería de 9V).

V. Software

OBJETIVO:

Nuestro objetivo es programar a Pancho y sus componentes. Se quiere que cuando se toque el botón empiece a correr un tiempo establecido en este caso 1 minuto. Y en este tiempo se quiere que cuando las pinzas toquen el metal, la luz se encienda y el *"piezzo"* toque un tono, por un tiempo específico y que el tiempo pare y pase el turno al próximo jugador.

¡Recordemos!

Lo primero que se establece cuando se abre el compilador de Arduino, es programar las dos funciones principales del programa. La primera función es *"set up()"* donde se establece las condiciones iniciales como por ejemplo los modos de uso de un los puertos. La segunda función principal que debe tener el programa es *"loop()"*. Dentro de esta función se programa lo que se quiere que corra en el micro controlador indefinidamente, luego de haber terminado la primera función, hasta que no esté conectado a una fuente. Ninguna de las funciones devuelve algún valor así que se le coloca *"void"* afrente de la función. Las instrucciones que se encuentren dentro de estos símbolos *"{ }"* son las que se ejecutaran cuando la función sea llamada.

```
void setup(){
}
void loop(){
}
```

Es importante recordar que un puerto puede tener dos modos, uno encendido que se programa con *HIGH* y uno apagado que se programa *LOW*. Más adelante utilizaremos esto, para programar la funcionalidad de cada puerto.

Para programar este juego de Operation, tenemos que importar varias librerías. Para Arduino las librerías se incluyen colocando un *"#include Nombredela librería.h"*. Para este programa tiene que incluir la librería *"<Wire.h> "*, *"Adafruit_LEDBackpack.h" , "Adafruit_GFX.h"* . Estas librerías son utilizadas para la comunicación entre el Arduino y el 7-segment.

```
#include <Wire.h>
#include "Adafruit_LEDBackpack.h"
#include "Adafruit_GFX.h"
```

1

¡Creemos variables!

Luego de que hayamos incluido las librerías necesarias, podemos iniciar variables.

Recuerda las **variables** son simplemente nombre de elementos guardados en memoria. Para que este programa funcione se tiene que inicializar tres variables de tipo *"int"* fuera de las funciones principales. La primera variable sera una que guarde el número de segundos que hay en un minuto, esto nos servirá para enseñar los segundos en la pantalla del 7-segment. La segunda variable, representa el estado inicial de un botón que empezara el conteo para que cada jugador tenga un turno. Para este caso se inicia a *"0"*, que significa *LOW*. En caso de el botón significa que el circuito no está cerrado. Cuando el circuito cierre, entonces esto obtendrá el valor de un *"1" o HIGH.* Se creará una tercera variable que representará cada vez que las pinzas completan el circuito, esto es el mismo comportamiento de un botón. El formato para crear variables es *"TIPO NOMBRE = VALOR".* Luego como último se inicializa una matriz con la librería que se incluyó en el programa. Lo que está ocurriendo es que se está creando una matriz de tipo *"Adafruit_7segment()".*

```
int t=60;
int gameOn=0;
int organs=0;
Adafruit_7segment matrix=Adafruit_7segment();
```

¡Configuremos la funcionalidad!

En la primera parte hay unos comandos necesarios para que el "7-segment" funcione. Luego se inicializa la matriz a una dirección específica que ya vino integrada. Esto se hace con el comando *"matrix.begin(0x70);"* ese es la dirección especifica de nuestro *"7-segment"*.

Para configurar los puertos del micro controlador existe un comando específico el mismo es *"pinMode(#de_Puerto, INPUT/OUTPUT"*. Uno coloca *INPUT* si quiere que el puerto lea información de entrada. Y se coloca *OUTPUT* si se quiere ejecutar un comando a través de ese puerto. En nuestro caso necesitamos dos puertos para entrada, estos serán utilizados para el botón y para las pinzas cuando cierren el circuito. Tendremos un tercer puerto dedicado a salida, que es el puerto donde se ejecuta el sonido de el *"piezzo".*

```
void setup () {
  #ifndef __AVR_ATtiny85__
   #endif
   matrix.begin(0x70);


  pinMode(13,INPUT); //input de boton
  pinMode(10,INPUT); //input de pinzas
  pinMode(11,OUTPUT); //output piezzo
}
```

2

¡Creemos la función infinita!

Lo primero que se quiere es imprimir en la pieza del 7-segment el tiempo establecido en este caso 60 que son los segundos que tiene el jugador para sacar órganos de nuestro amigo pancho. para hacer esto existe un comando de las librerías incluidas que nos permite hacer esto. El comando es *"matrix.print(valor, base_del_valor)"* , en este caso la base de nuestro número es decimal. Y el comando que aparece al lado *"matrix.writeDisplay();"*.

**Gotitas del saber: Cuando decimos que un número es decimal, cuando es de base diez. Esto son los números que utilizan en sus clases de matemáticas desde elemental.

Luego se quiere obtener el valor del puerto de entrada de las pinzas para ver si el jugador perdió su turno. Existe un comando en C que lee el puerto que se le coloque como parámetro y así lo guardas en la misma variable y así ella tendrá el valor actual del puerto. Este comando es *"digitalRead(#PIN);"*, si lo quieres guardar en la variable, se coloca la variable y el signo de asignación *"="*.

**Gotitas del saber: En programación "=" este símbolo no significa "igualdad", significa asignación (Por ejemplo; Yo asigné el trabajo a Julio. Esta premisa significa que el trabajo es de Julio.) En programación igualdad se expresa como "==".

Luego de leer el valor de entrada queremos verificar si esta en HIGH para que el *"piezzo"* alarme y notifique al jugador que no puede continuar jugando porque perdió. Para esto se hace una comparación entre la variable y el valor *HIGH*, si esto se cumple entonces se utilizará la siguiente función *"tone(#de_Puerto, Frecuencia, segundos)"*.

```
void(*resetFunc) (void)=0;//función que nos
//ayudará luego

void loop() {

  matrix.print(t,DEC);
  matrix.writeDisplay();

  organs = digitalRead(10);

  if(organs==HIGH){

tone(11,200,100);

    }
```

Luego se quiere verificar si el botón está en el estado *HIGH* para comenzar el reloj y comenzar el juego. Se hace un *"digital Read"* como se hizo anteriormente, pero esta vez deben usar el puerto del botón en este caso 13, y se guardará bajo el el nombre *"gameOn"*. Luego se necesita hacer una comparación de dos variables. Queremos comparar que la variable del tiempo este en 60 y que el botón haya sido apretado, esto se verifica comparándolo con un *HIGH* y comparando la variable con 60.

```
gameOn= digitalRead(13);

if(gameOn==HIGH && t==60){
```

**Gotitas del saber: En programación cuando quieres programar un "y" el símbolo que se utiliza para representar la "y" es "&&".

Si estas condiciones se cumplen, queremos hacer una iteración de un programa que dure hasta que el tiempo sea 0. Esto se hace con un *"while loop"*, con un comando *"while(CONDICION) { INSTRUCCIONES } "*.

¡Programemos dentro del loop while!

Dentro del loop se quiere volver a verificar si las pinzas están tocando las paredes, se hace lo que se hizo al principio del programa en el paso 4, con una diferencia. La diferencia será, que se le añadirá un comando *"resetFunc()"*, este comando llevará al Arduino reinicie . Se quiere reiniciar para que comience de nuevo la función original del Arduino.

Antes de este comando se le añada un *"delay(#MILI-SEGUNDOS)"* para que de tiempo de escuchar el tono y que vuelva a comenzar la función loop, este comando le da pausa al programa por los mili-segundos especificados.

Al salir de este chequeo se quiere imprimir el tiempo como en el paso 3, en el 7-segment y se le quitara un valor a la variable t, en este caso esta es la variable que contiene el número de segundos. Luego de terminar, este loop seguirá ocurriendo por 60 segundos, hasta que t llegue a 0. Muy importante es finalizar cada linea, con un punto y coma. Y cerrar los debidos corchetes.

```
  while (t>=0){



//Código antes visto parte 4
    organs = digitalRead(10);
    if(organs==HIGH){
      tone(11,200,1000);
      delay(1000);
      resetFunc();
    }
//Código antes visto parte 3
    matrix.print(t,DEC);
    matrix.writeDisplay();
    delay(1000);
    t--;
  }
    t=60;
 }
}
```

Preguntas Frecuentes:

1. ¿Por qué mi Operation no enciende?

Verifique que todas las conexiones estén correctas. De estar correctas, verifique la batería, asegúrese de que tiene aún energía. De no funcionar se puede comunicar con nosotros.

2. La bombilla encendía, pero ya no. ¿Qué sucedió?

Lo más seguro su LED recibió más voltaje (o corriente) del necesario y se ha quemado, verifique que su resistor tenga los ohmios correctos y que este bien conectado. Reemplace la bombilla.

3. La pinza no hace nada, ¿Qué pasa?

La pinza no siempre es de materiales conductores, si la pinza es plástica, no va conducir electricidad. Coloque dos pedazos de cinta adhesiva de cobre y péguela en las puntas de la pinza. A cada extremo conecte un cable. Verifique su conexión.

Evaluación:

1. ¿Cuánto voltaje se necesita para que el juego corra?

2. ¿Por qué se utiliza la cinta de cobre en los bordes del interior de Pancho y en las pinzas?

3. ¿Qué resistores se utilizan para el 7-segment y cuál es su función?

Respuestas:

1. 5V
2. Esta sirve de conductor de corriente.
3. 10kOhm porque requiere un voltaje menor al de la fuente.

Reference:

[1] Departamento de Educación de Puerto Rico, http://www.de.gobierno.pr/, buscado el 10 de septiembre del 2016.