[](https://www.google.com.mx/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&cad=rja&uact=8&ved=0ahUKEwiLht24m_DVAhUnllQKHd8SDDsQjRwIBw&url=https://www.uv.mx/instru/&psig=AFQjCNEbtXJyh1GAjqyUcX1-S8Ze4FcsfQ&ust=1503675912973634)[](https://www.google.com.mx/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&cad=rja&uact=8&ved=0ahUKEwiy4q2Im_DVAhVkwVQKHZb7AmIQjRwIBw&url=https://www.uv.mx/veracruz/nutricion/logos/&psig=AFQjCNH8WGtW5-1aH_KRPjBg2ARPnYR93Q&ust=1503675809245006)

Tópicos avanzados ii  
Sistemas embebidos

L01. Desarrollo de un sistema embebido simple

\*Digital timer  
\* RPM meter  
\*Digital code lock

Integrantes:

Quirino Cortes Akira Belem

Cancela Mora Jessica

Fecha de entrega: 22 de agosto de 2019

**Objetivo**

Revisar cada parte de los códigos del Digital timer, RPM Meter y Based digital code lock analizando también el diagrama eléctrico de cada uno con el fin de determinar su funcionamiento y los procesos que lleva a cabo cada sistema. Entender cada parte que conforma el código nos permitirá realizar ciertas mejoras.

***Temporizador (Digital timer)***

**¿Qué se implementó?**

Se implementará un temporizador digital que usa 4 dígitos de siete segmentos de cátodo común. Consta de controles para aumentar o disminuir minutos y segundos, además de un control para reiniciar el temporizador.

**Entradas.**

Las entradas están definidas por pulsadores que establecen cuanto aumentan los minutos y segundos. Por cada pulsación que se haga se definió en el código un ajuste de Pull-ups, poniéndolas en alto cuando alguna sea detectada.

**Salidas**

En el arduino se tienen ocho salidas que vienen de la parte digital tres de ellas entran al conversor serie-paralelo (74HC595), una en el reloj (SH\_CP), otra en el dato (DS) y la ultima en la captura (ST\_CP). Una cuarta salida del arduino va al buzzer y otras cuatro son las líneas que controlan los cátodos de los 7 segmentos. Las salidas del conversor son mostradas en el desplegador de siete segmentos de 4 dígitos.

**Procesos**

Teniendo como entradas los pull-ups de minutos y segundos, en el arduino se realizará una comunicación serial con el conversor para manejar el arreglo de siete segmentos.  
Las líneas que controlan los cátodos del siete segmentos van de la D2 a la D5; para el conversor, el data y el latch son la D6 y D7 respectivamente y para el clock PB0.  
Al ser transferidos los datos en forma serial al conversor este los transferirá de forma paralela al desplegador de siete segmentos de 4 dígitos para así mostrar los dígitos correspondientes a las pulsaciones que se hayan realizado. Así, dependiendo cuantas pulsaciones se hayan hecho, ya sea en el pulsador de minutos o segundos, se comenzará una cuenta regresiva a partir de lo que se dé como inicio en el temporizador. Al llegar a los cero minutos y cero segundos el buzzer deberá sonar alertando de que el tiempo que se ingresó con los pull-ups ha terminado.

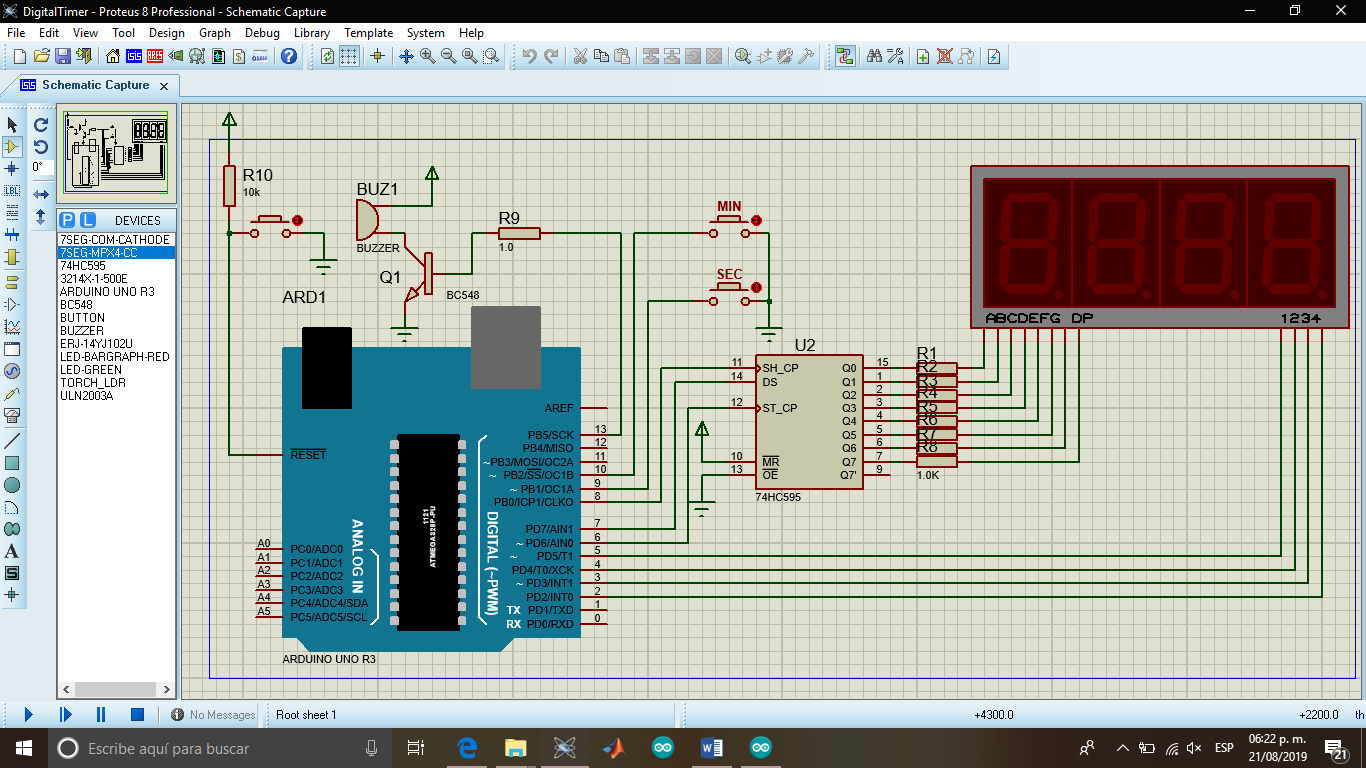
**Características.**

Mediante los pull-ups se realizarán las pulsaciones para ingresar algún tiempo, luego comenzará una cuenta regresiva. Esto se puede observar en el desplegador de siete segmentos de 4 dígitos. Una vez trascurrido el tiempo el buzzer sonará alertando que el tiempo ha terminado.

**Lista de partes.**

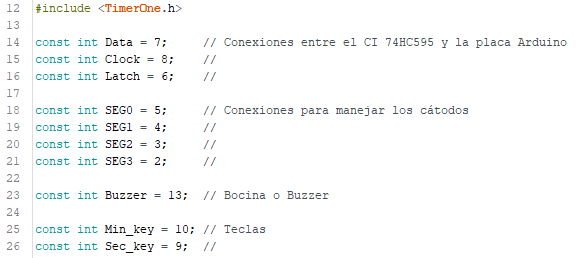
1- 7 segmentos con 4 dígitos.  
1- Conversor serie-paralelo 74HC595  
1- Arduino uno  
1- Transistor NPN BC548  
3- Pulsadores  
1- Buzzer  
9- Resistencias de 1KΩ  
1- Resistencia de 10KΩ

**Diagrama eléctrico**

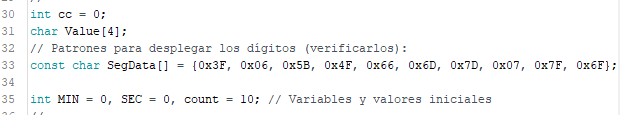


**Listado de código**

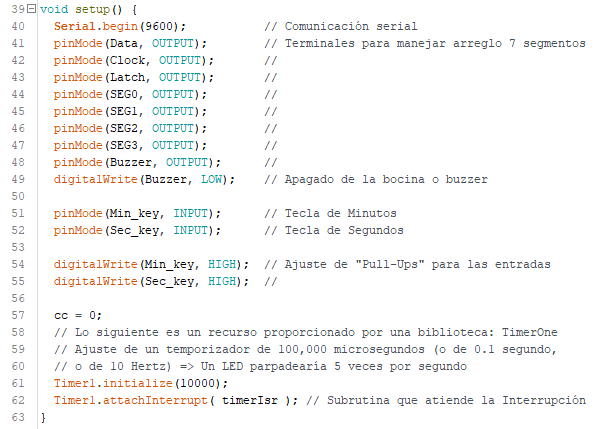
1. En la primera parte del código se determina la biblioteca que se requiere, así como las conexiones que se necesitan para realizar la comunicación entre el arduino y el 74HC595, también con el buzzer y las teclas. Cómo se puede observar en el diagrama eléctrico, las líneas de conexión tienen el número que se determina en el código.

****

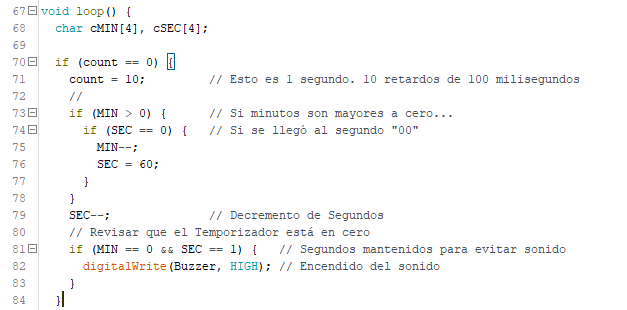
1. En la segunda parte del código se declaran variables, tales como minutos, segundos y un contador las cuales cuentan con un valor inicial de 0.



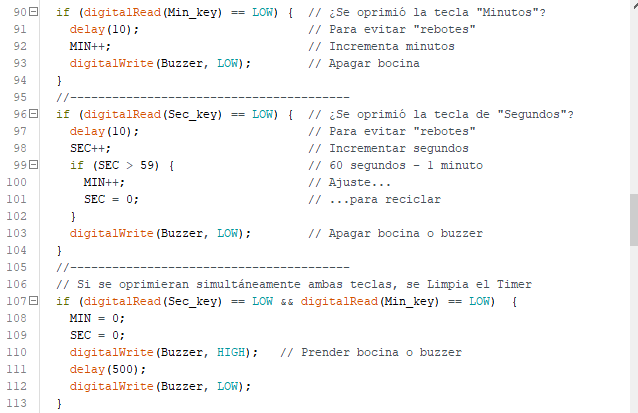
1. En el void setup se realizan los ajusten generales, tales como determinar si los pines puestos en la primera parte son de salida o entrada. En esta parte también se determina que el buzzer debe estar apagado al inicio, para eso se le coloca “low”. Al contrario del buzzer los pull-ups deben estar en “high”.



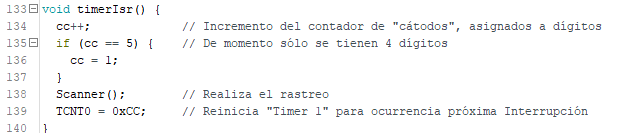
1. El siguiente es el lazo principal, que se encuentra dentro del void loop. En la primera parte de este lazo se declara una condicional “if” preguntando si el contador es igual a 0 de esta forma se le asigna el valor de 10 al contador que es equivalente a 1 segundo. Dentro del primer “if” se encuentran otros de forma anidada. En el segundo se pregunta si los minutos son mayores a 0 y en el tercero si los segundos son iguales a 0, si esto se cumple los minutos se disminuyen y los segundos valen 60 luego de esto es necesario determinar que los segundos también deben ir en decremento. También anidado con estos “if´s” se tiene otra condición que determina cuando debe sonar el buzzer y de esta forma lo pone en “high”.



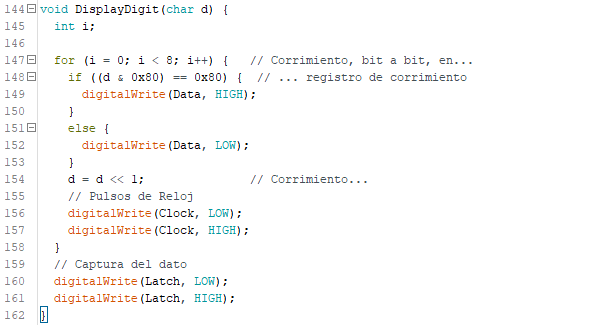
1. Dentro del mismo lazo principal se encuentran las condicionales de los pull-ups, en ellas cuando en el arduino se lee que ha sido presionado (low) se incrementan los minutos o segundos dependiendo de cuál se presione. Para los segundos también se requiere determinar que cuando estos sean mayores a 59 los minutos aumenten y los segundos queden en 0.



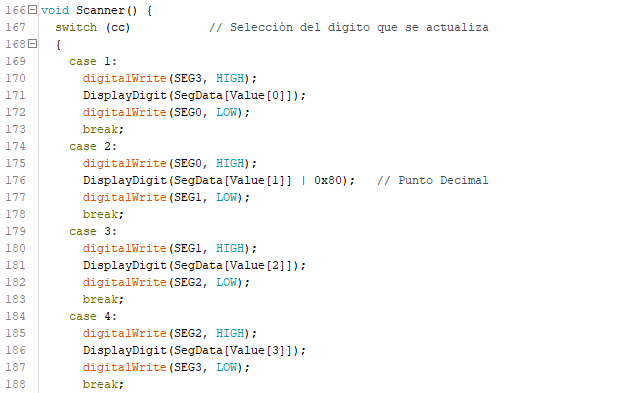
1. Lo siguiente en el código son las subrutinas. Una de ellas es la interrupción causada por el desbordamiento de timer 1. Aquí se incrementa el contador de cátodos.



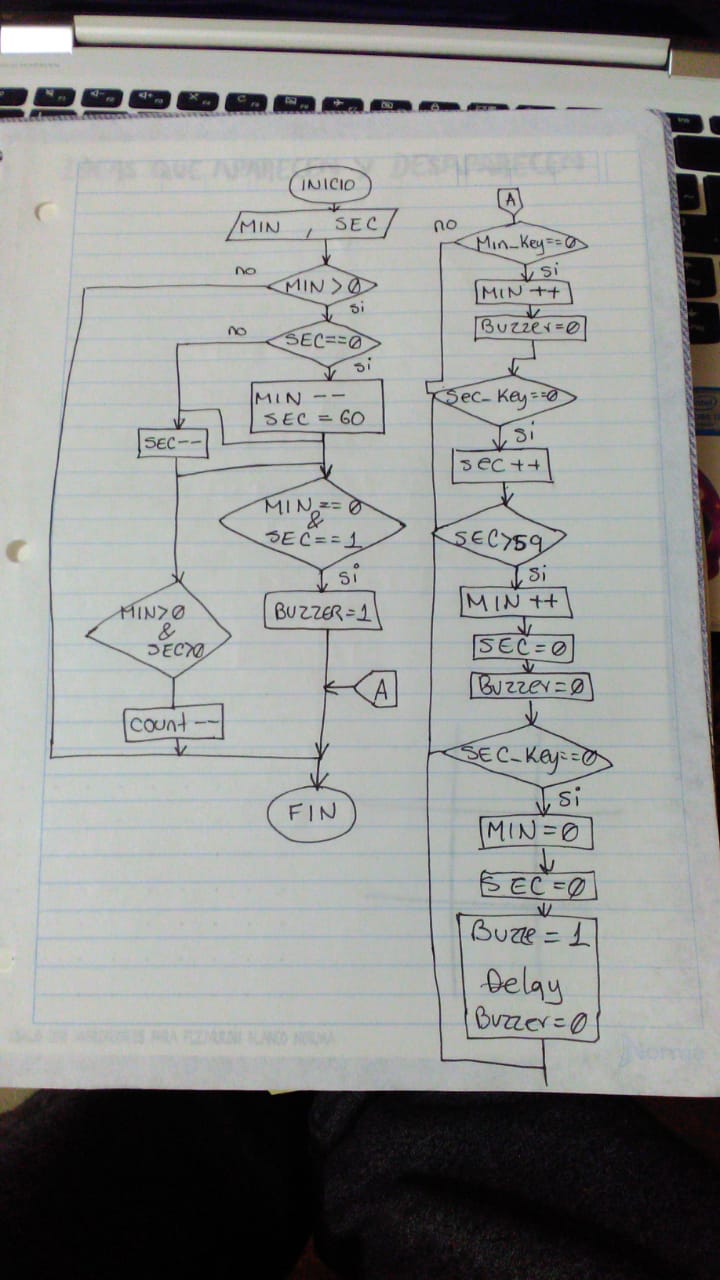
1. Otra subrutina es la generación de los dígitos. En esta parte se genera el corrimiento de bits determinando en código ASCII el registro de corrimiento. Se realizan los pulsos de reloj y las capturas del dato.



1. La ultima subrutina es la función de rastreo del desplegador, aquie se determina el digito que se actualiza. Para eso se utilizan las condicionales “case” en las que se determina cuáles han sido activadas al pulsar los pull-ups.



**Diagrama de flujo**

****

***RPM Meter (Contador de Revoluciones por minuto)***

**¿Qué se implementa?**

En este sistema se implementó un Arduino UNO, un desplegador de 7 segmentos de 4 dígitos, un convertidor de serie-paralelo, resistencias y un generador de pulsos (en el caso de simulación).

**Entradas:**

En la entrada se utilizó un generador de pulsos para simular el funcionamiento de un odómetro o similar con el cual se medirían las RPM de manera física.

**Salidas:**

Para 3 salidas se utilizó un convertidor de serie-paralelo el cual nos facilita el manejo de las variables a mostrar en el desplegador de 7 segmentos de 4 dígitos el cual mostrará el número correspondiente a las RPM. Aparte de los 4 pines que van conectados a los cátodos del desplegador, en total tenemos 7 salidas.

**Procesos:**

El sistema arranca cuando se alimenta el arduino, entonces la entrada acumulará la duración del pulso y una vez que el tiempo del ciclo termine los almacenará en la variable de previousmillis la cual se necesitará para poder ir actualizando la medición. Una vez que se tienen los primeros datos el código tendrán un proceso matemático de conversión de frecuencia a RPM. Los datos de la salida se actualizarán dependiendo el caso del dígito, en este caso solo pueden ser 4 dígitos, esto se estará actualizando ya que la entrada está en constante cambio y el código está hecho para que se lean los datos constantemente.

**Características:**

Este sistema está diseñado para que la medición de las RPM sea constante y se muestre las lecturas en el desplegador. Tiene la ventaja del uso del timer para tener una medición constante de la entrada.

**Lista de partes:**

\*Arduino UNO

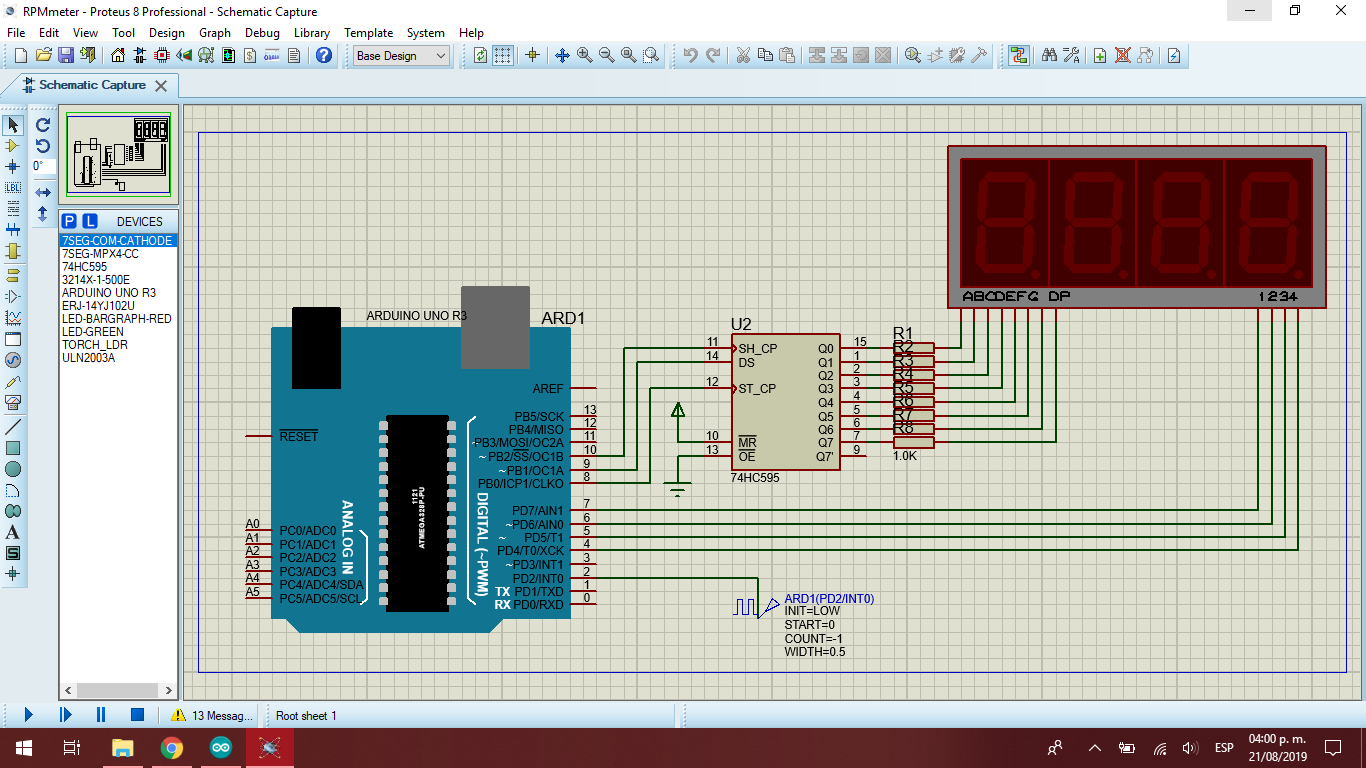
\*Desplegador de 7 segmentos de 4 dígitos

\*Convertidor serie-paralelo

\*Resistencias de 1k Ohm

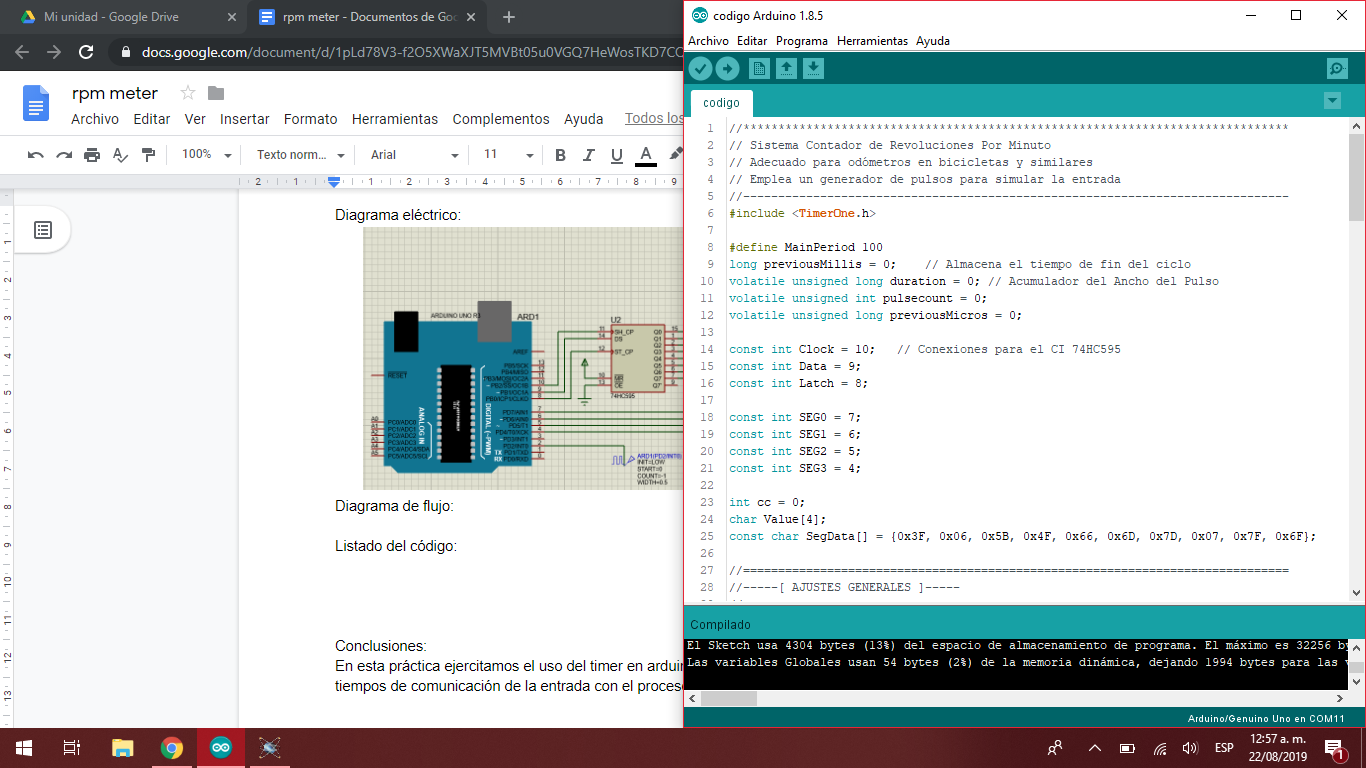
\*Generador de pulsos

**Diagrama eléctrico:**

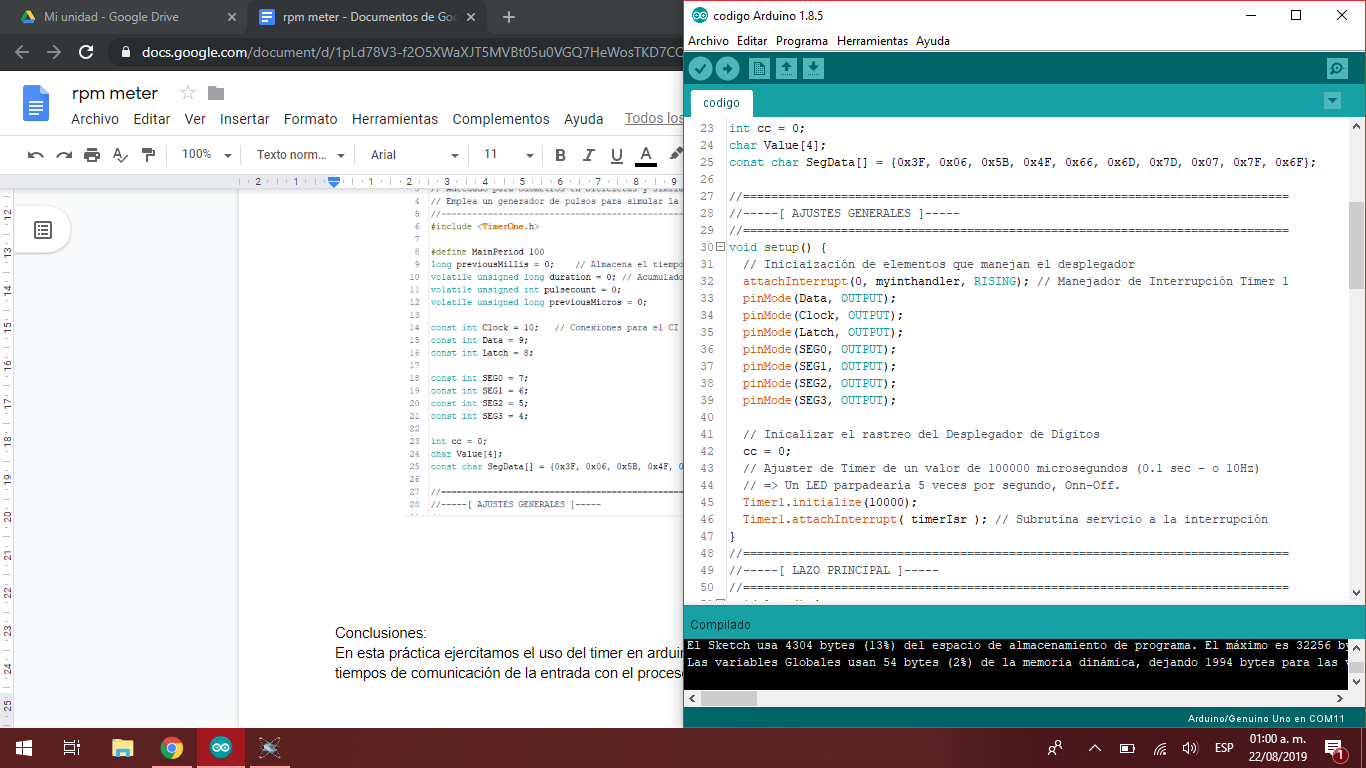


**Listado del código:**

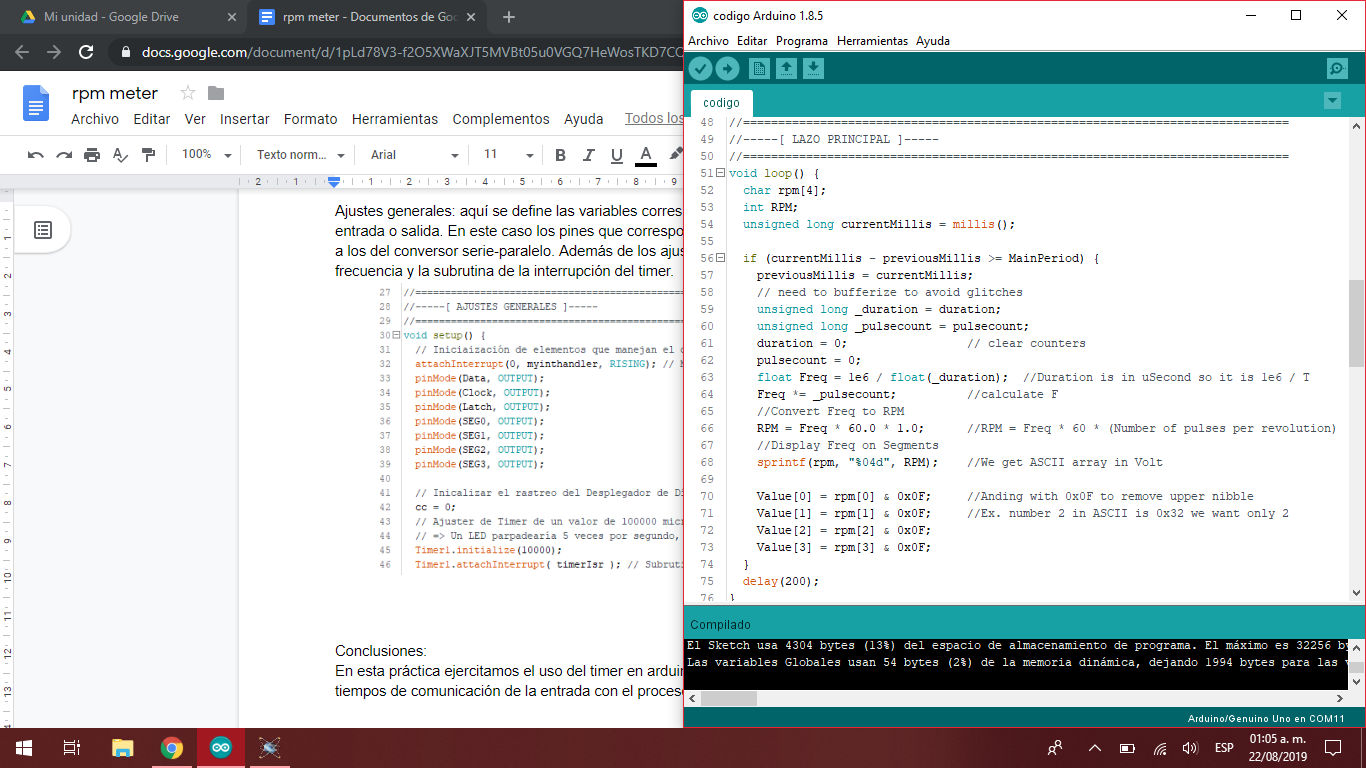
1. El encabezado, inclusión de librerías necesarias, en este caso TimerOne, definición de pines y variables.



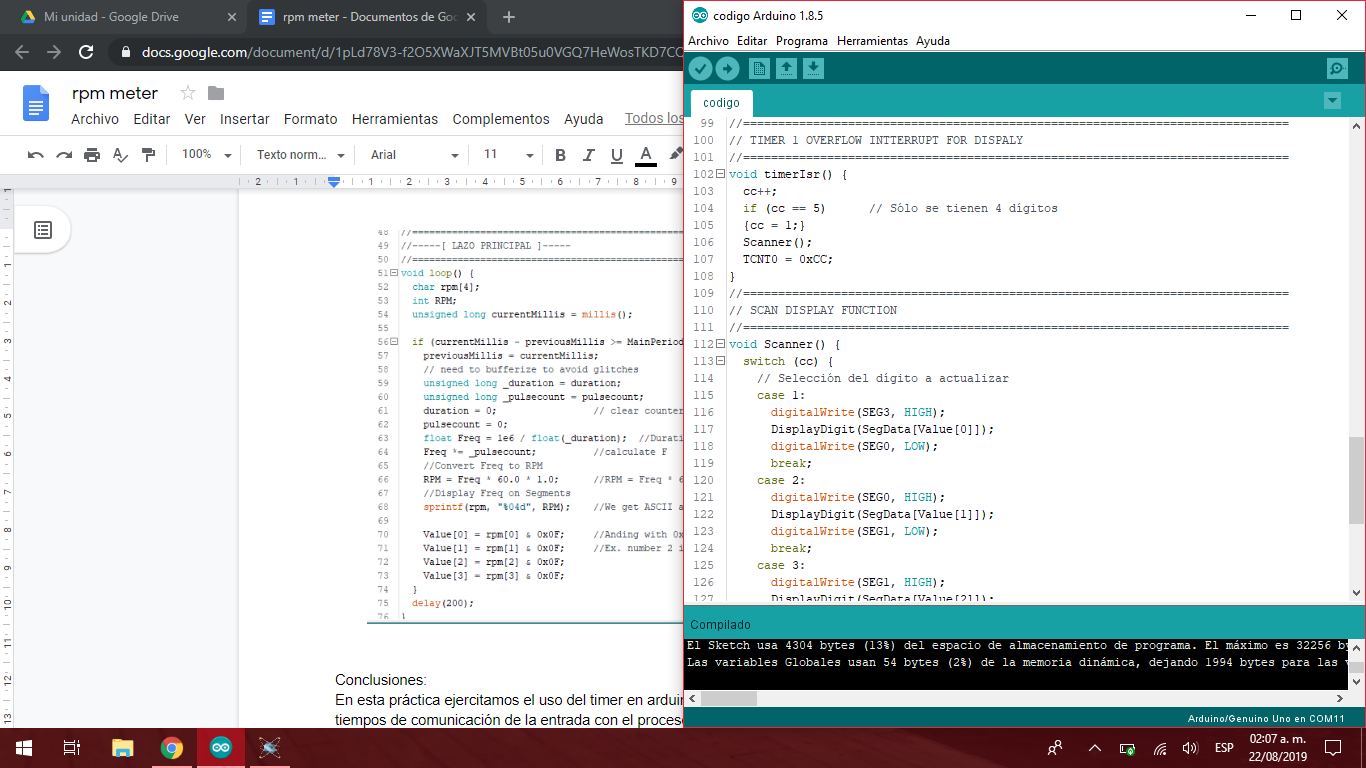
1. Ajustes generales, aquí se define las variables correspondientes a cada pin y definición de entrada o salida. En este caso los pines que corresponden a los cátodos del desplegador y a los del conversor serie-paralelo. Además de los ajustes del timer correspondientes a la frecuencia y la subrutina de la interrupción del timer.



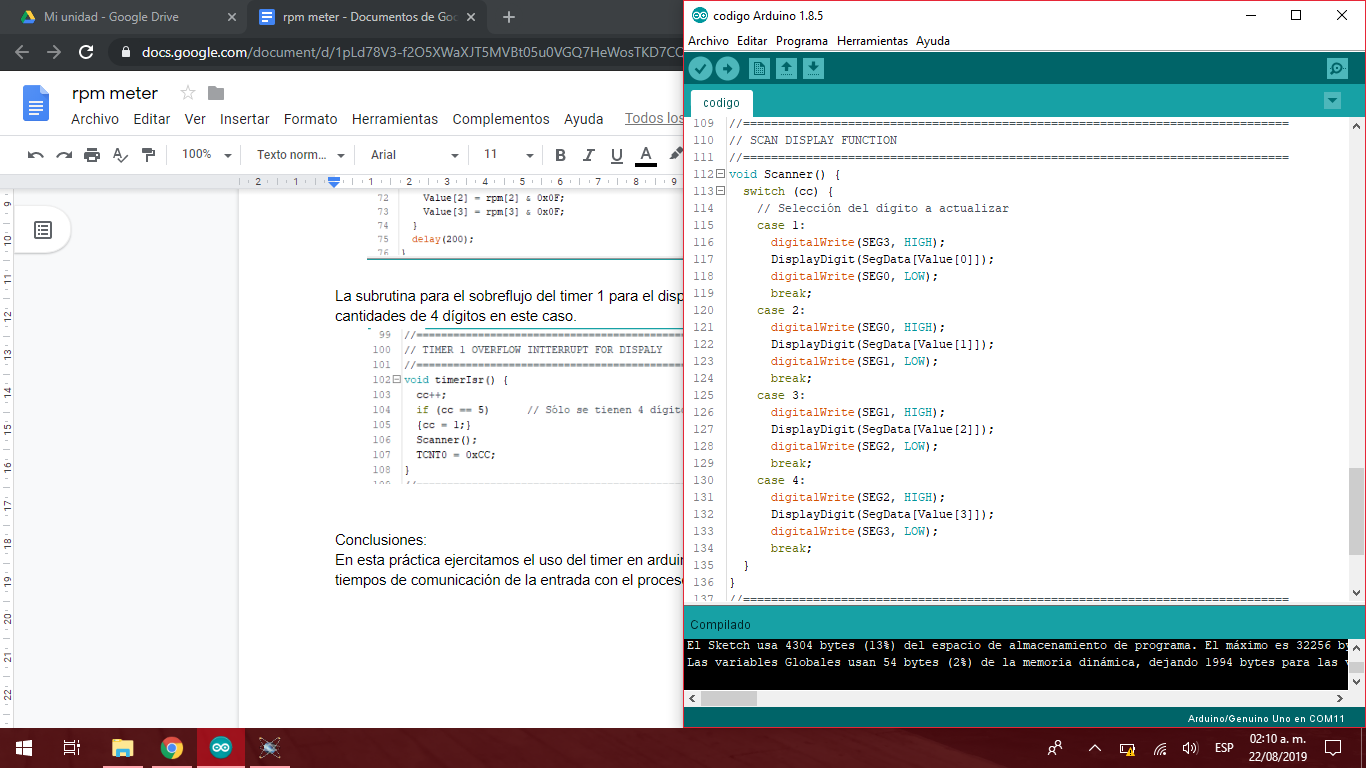
1. El lazo principal o loop es el que se estará ejecutando siempre, entonces empieza por definir las variables que se mostrarán en el desplegador, después con la rutina if se hace una diferencia de la primera lectura guardada (PreviousMillis) con la nueva (currentMillis) para ver si el tiempo es mayor o igual a el periodo principal (MainPeriod), si es así el valor nuevo se sobrescribe en el ya guardado. se define las variables de duración y contador de pulsos las cuales se inician de 0. La variable duración se cuenta en segundos, después se calcula la frecuencia a partir de la variable que cuenta los pulsos. Una vez obtenida la frecuencia se puede calcular la RPM, se asignan los valores a los 4 dígitos del desplegador. Y se define un retardo de 200 ms para el envío de los datos.



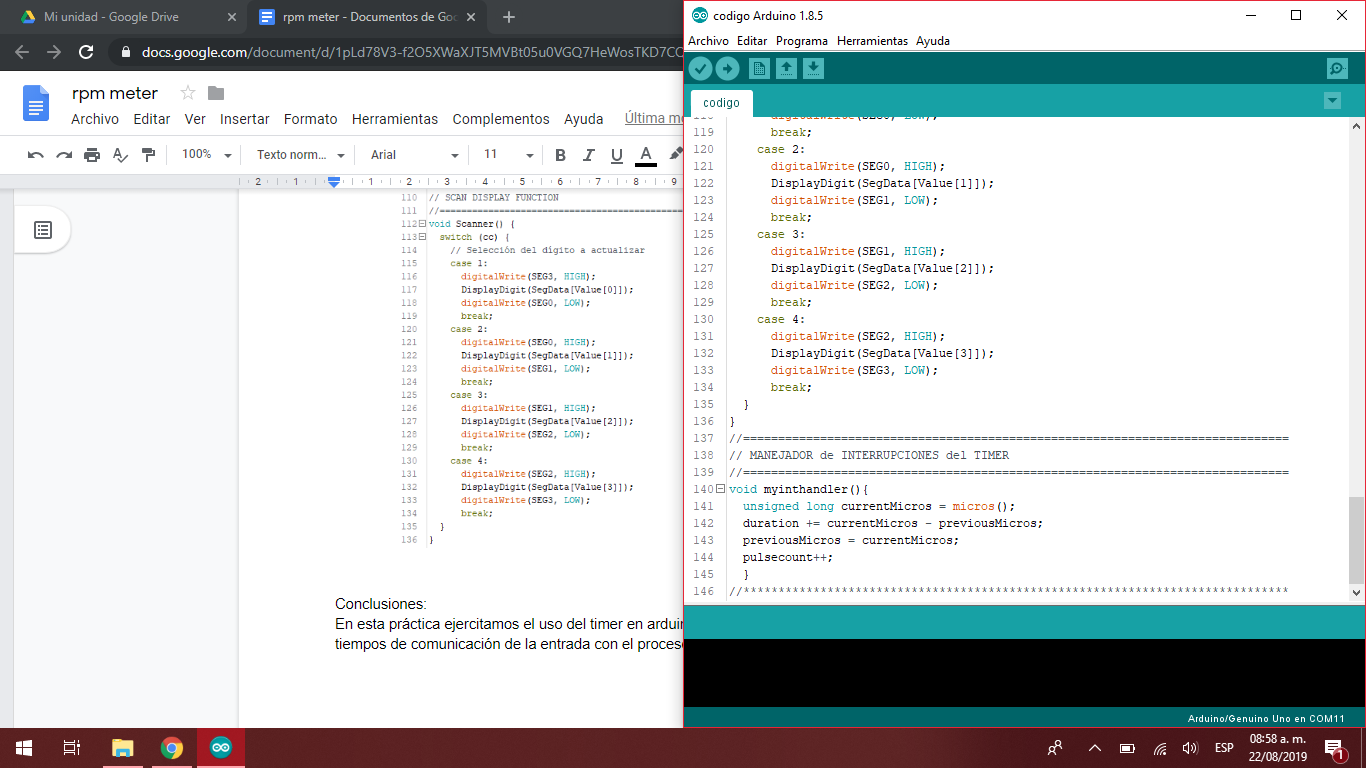
1. La subrutina para el sobreflujo del timer 1 para el display. Ya que se tiene que manejar cantidades de 4 dígitos en este caso. Cuando llegue a una cantidad mayor a 5 dígitos este tendrá que r



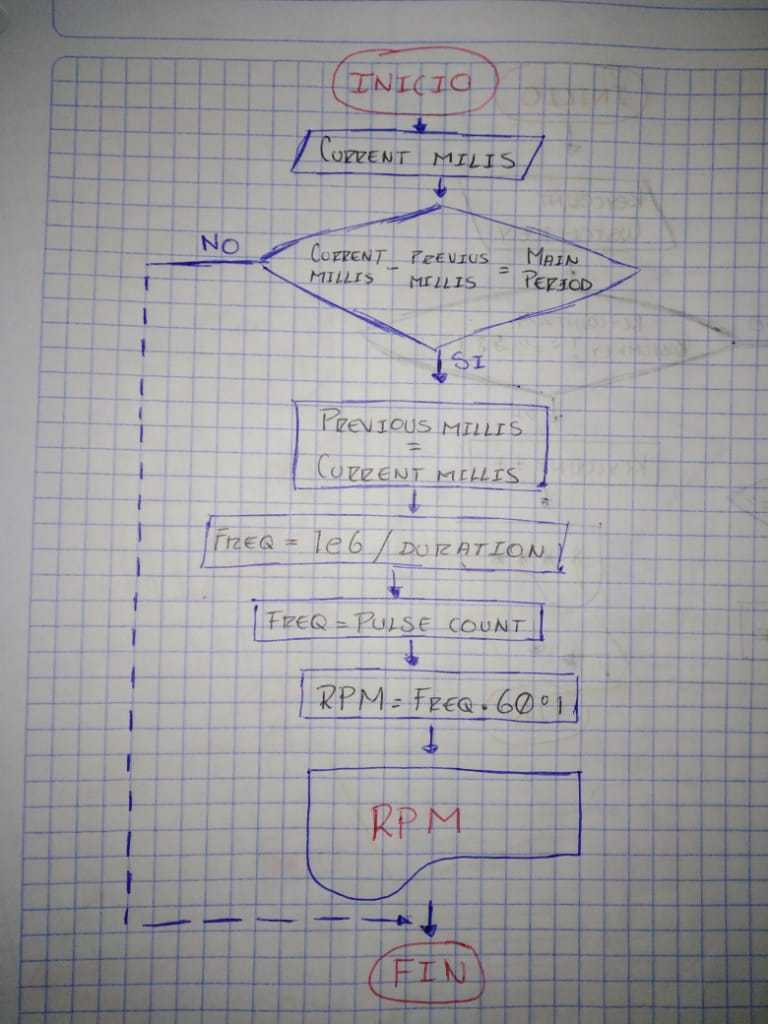
1. La rutina para el escaneo el cual se encargará de ver que digito del desplegador tendrá que ser actualizado debido a un cambio en la variable que se lee.



1. El manejador de interrupciones se configura para que vaya acumulando los datos que se reciben de la entrada, la cual se guardará en micros.



**Diagrama de flujo:**



***Based digital code lock (Sistema de cerradura de puerta)***

**¿Qué se implementa?**

En esta práctica se implementó el uso de un Arduino UNO, un teclado matricial de 4\*4, un display de cristal líquido, resistencias, leds, relay, transistor y pull ups.

**Entradas:**

Se tienen 8 pines entradas que corresponden a las variables que se digitan en el teclado. 4 de filas y 4 de columnas ya que es un teclado matricial. Además de una entrada para el reset el cual se tiene en caso de que se quiera reiniciar el sistema.

**Salidas:**

De salidas tenemos una para el relevador de apertura de la puerta, 2 hacia los leds indicadores de estado de cerrado y abierto, 6 salidas que corresponden a las conexiones de la LCD.

**Procesos:**

El sistema inicia con el estado de la puerta cerrado, lo cual desplegará el mensaje para ingresar una contraseña en la LCD. cuando se ingresa una contraseña seguido de la tecla on el estado de la puerta cambia a abierto el led verde enciende y el relevador se cierra. Y se imprime el mensaje de puerta abierta.

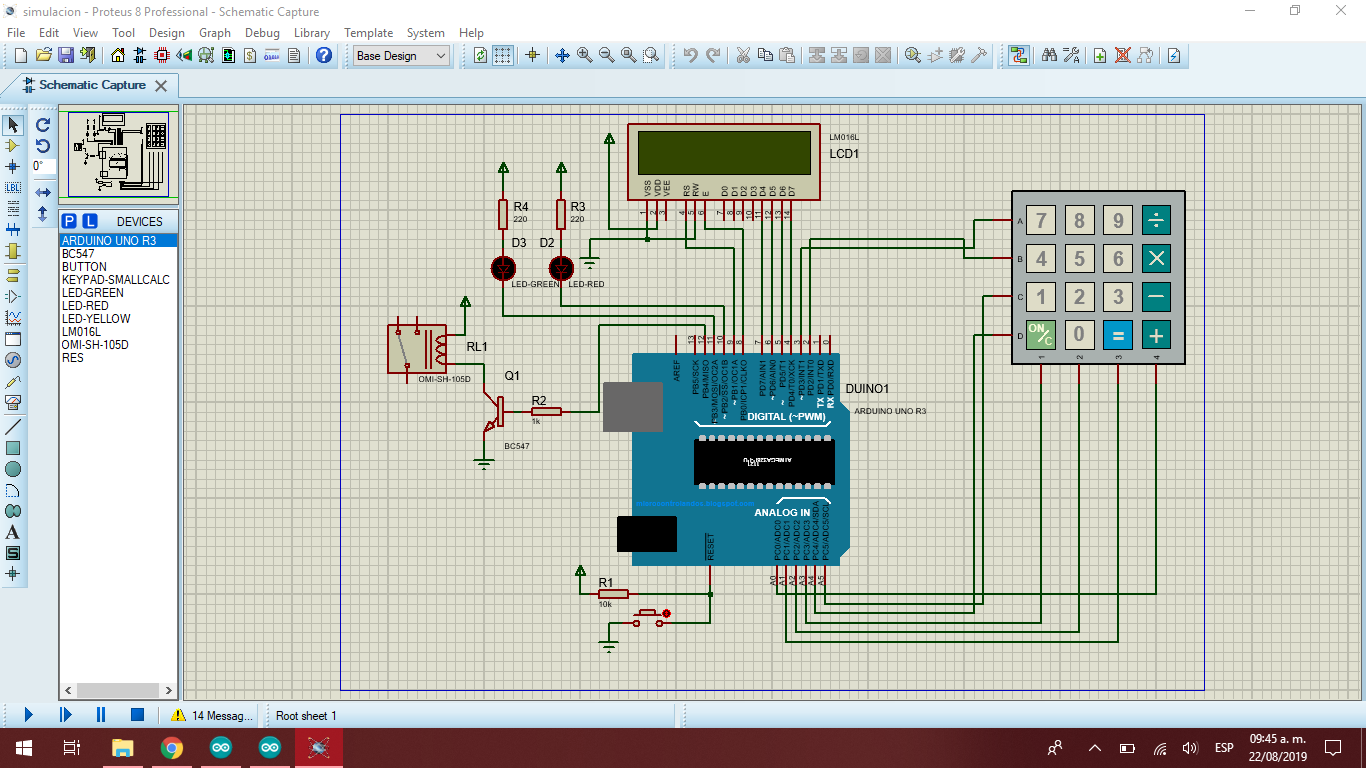
**Características:**

El sistema se diseñó como un sistema de seguridad, el cual tiene muchos usos, la aplicación del teclado está visualizada para muchos usos ya que se puede mejorar con teclado hexadecimal.

**Lista de partes:**

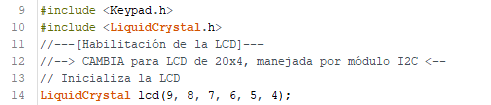
\*Arduino UNO  
\*Teclado matricial  
\*LCD  
\*Relay  
\*Transistor  
\*Pull up  
\*Resistencias

**Diagrama Eléctrico:**

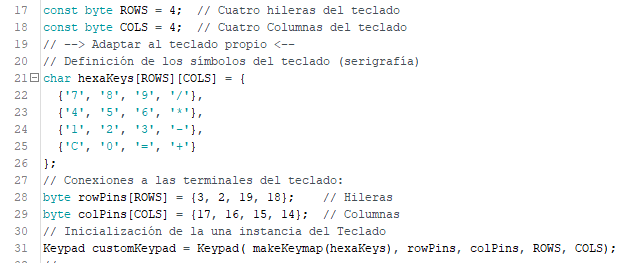


**Listado de código.**

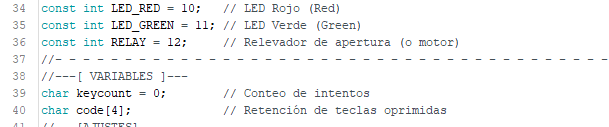
1. La primera parte del código incluye las librerías que se requieren. En este caso <Keypad.h> para la utilización del teclado y <LiquidCrystal.h> para la utilización de una LCD.



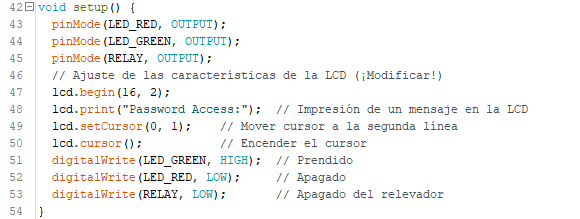
1. Para la habilitación del teclado se debe determinar cuántas filas y columnas va a tener. Luego de esto se establece el orden en el que se encuentran los dígitos del teclado. Finalmente, en base al diagrama eléctrico se escriben las conexiones con las terminales del teclado.



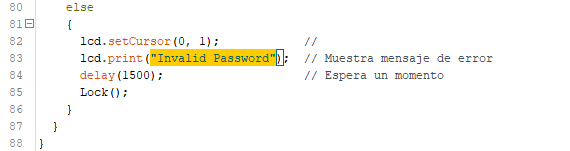
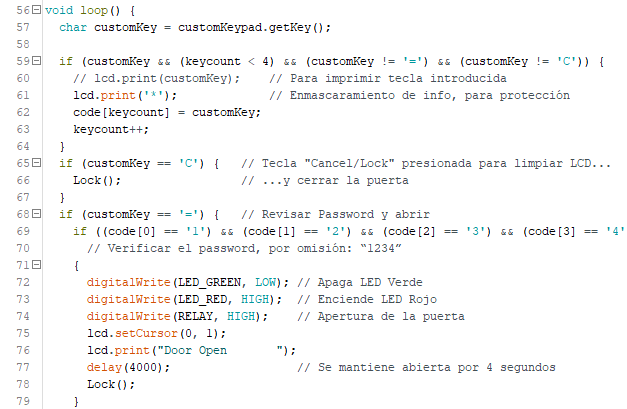
1. En la habilitación de otros elementos se determinan las líneas de conexión que tiene el arduino con el led rojo y verde, así como con el relay.   
   Se declaran variables una que va a decirnos cuantos intentos se han realizado y un arreglo de 4 que retenga las teclas oprimidas.



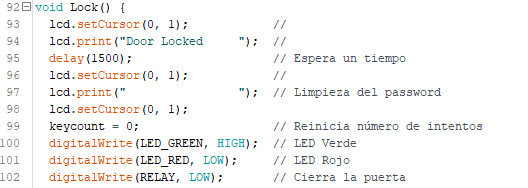
1. Los ajustes se realizan en el void setup. Se determinan los leds y el relay como salidas. Se ajustan las carateristicas de la LCD y se manda como inicio el led verde encendido y el relevador y el led rojo apagados.



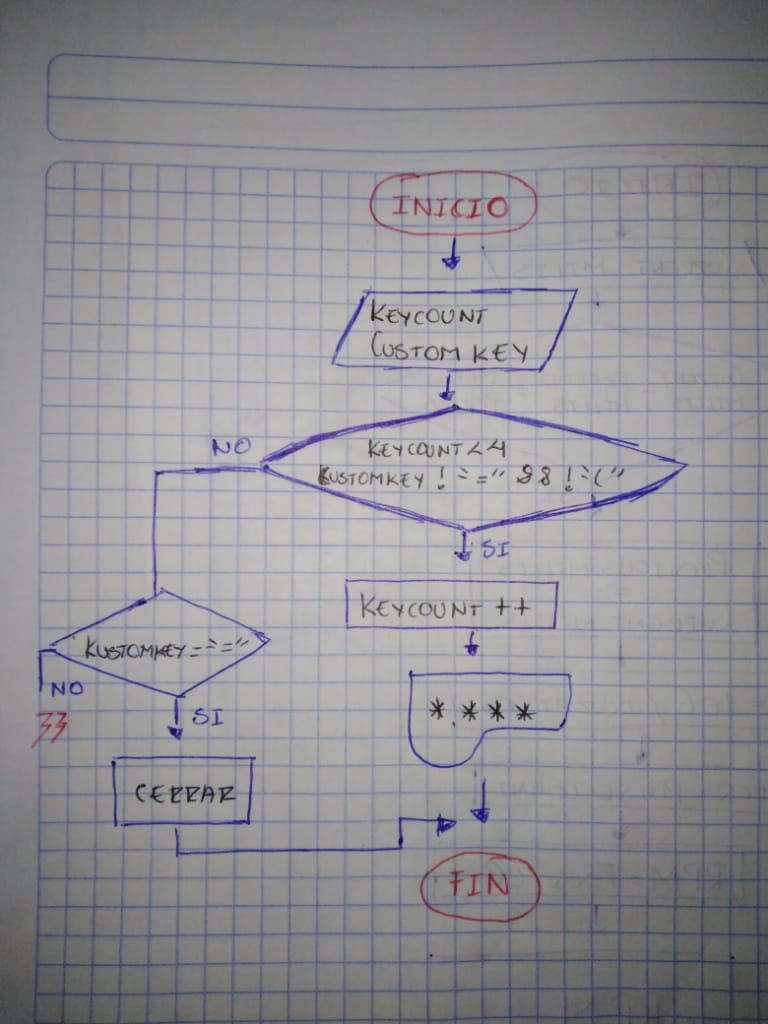
1. Dentro del ciclo del void loop se generan las condicionales para introducir la contraseña. En primera se pone el número de intentos que se tiene. Se enmascara la contraseña y por cada intento se aumenta el contador. La tecla “C” cancela la operación y cierra la puerta. Luego de escribir la contraseña es necesario presionar la tecla de “=” para verificar si la contraseña es correcta. De lo contrario mostrará el mensaje de "Invalid Password" esperará un momento y cerrará la puerta. En esta parte también se establece el encendido del led rojo el cual permite que el relay se abra lo que significa que la puerta está abierta.

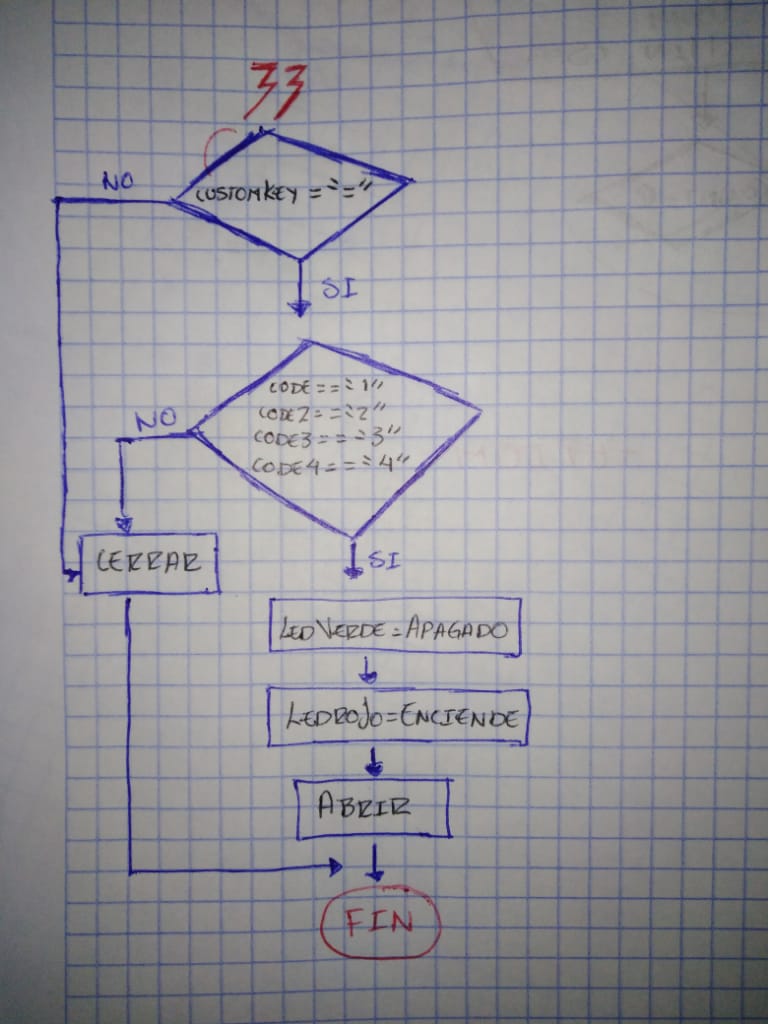


1. En la parte de subrutinas y funciones se imprime una nueva frase en la LCD lo que indica que la puerta se cierra. Se elimina la contraseña que se había ingresado, se reinicia el contador de numero de intentos y de nuevo se enciende el led verde que indica que la puerta está cerrada.



**Diagrama de flujo**

****

****

Conclusiones:

En esta práctica ejercitamos el uso del timer en arduino, el cual nos facilita el manejo de tiempos de comunicación de la entrada con el proceso principal. Este es útil al querer una medición constante de una variable. También comprendimos algunos comandos necesarios para controlar un desplegador de siete segmentos con 4 dígitos y una LCD. Obtuvimos más conocimiento sobre el manejo de la plataforma de arduino en conjunto con el simulador Proteus.