



# UNIVERSIDAD DE GRANADA

---

## DSE

*Proyecto Final: Apertura de una puerta Musical*

---



Autores: Elvira Castillo Fernández  
Jose Luis Izquierdo Mañas  
Juan Carlos Martínez Guerrero  
Adrián Pérez Ortega

# Índice

<b>1. Introducción</b>	<b>1</b>
<b>2. Funcionamiento y configuración</b>	<b>1</b>
<b>3. Diagrama del sistema</b>	<b>5</b>
3.1. Montaje del Circuito . . . . .	5
3.2. Componentes lógicos inferidos por PSoC y su funcionamiento . . . . .	6
3.2.1. PWM . . . . .	6
3.2.2. Timer para medir distancias . . . . .	7
3.2.3. ADC Delta-Sigma . . . . .	7
3.2.4. Master I2C para comunicación con la LCD . . . . .	8
3.2.5. Interrupciones . . . . .	8
3.2.6. UART para comunicación con ESP32 . . . . .	9
<b>4. Diagrama de estados del sistema</b>	<b>10</b>
<b>5. Protocolo de comunicación</b>	<b>12</b>

## 1. Introducción

El proyecto final que nosotros hemos planteado para la asignatura **DSE** (Diseño de sistemas Electrónicos) es un sistema que realiza el desbloqueo de una puerta utilizando como contraseña una secuencia de notas, dicha secuencia está compuesta por tres notas, las cuales se mueven en el rango del **do** hasta el **la** incluyendo en la escala los sostenidos. Si la contraseña es correcta la puerta se abrirá por un tiempo determinado y después se volverá a cerrar, si la contraseña es incorrecta y se exceden cierto número de intentos la trampilla colocada en el umbral de la puerta se abrirá siguiendo el mismo comportamiento que la puerta.

Cuando se introduce una contraseña, las notas escogidas se reproducen por un altavoz (en nuestro caso es un pequeño auricular), el sistema además detecta cuando hay un objeto en el umbral de la puerta para iniciar la cuenta atrás, en la que, se deberá introducir la contraseña correcta o en caso de que el tiempo se agote, se consumirá un intento.

Todo lo referente a la configuración de la puerta (fijar la cuenta atrás, fijar la contraseña de desbloqueo...) y algunas funciones como abrir/cerrar la puerta/trampilla se controla a través de una aplicación de **Android**.

## 2. Funcionamiento y configuración

Nuestro sistema tiene 2 etapas diferenciadas, una etapa de **configuración** y una etapa de **funcionamiento** (pudiendo volver en cualquier momento a la etapa de configuración si es necesario).

Indicaremos paso a paso el procedimiento a seguir para un correcto uso del sistema:

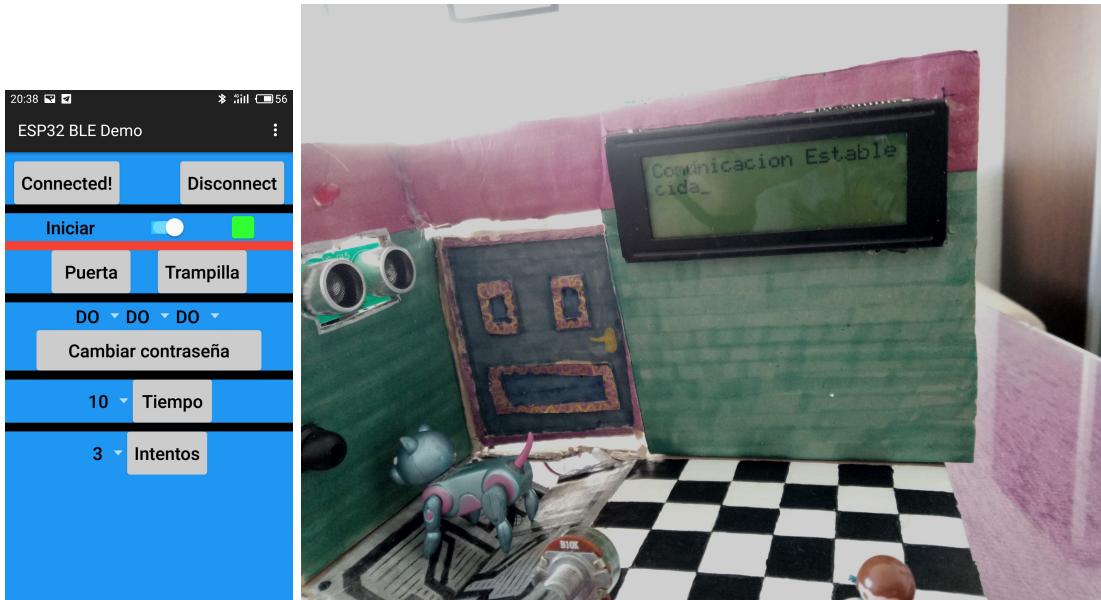
- 1.1 Lo primero que tenemos que hacer es establecer la configuración del sistema, tiempo de la cuenta atrás, numero de intentos y la contraseña deseada, todo desde una APP desarrollada por nosotros compatible con sistemas **Android**.



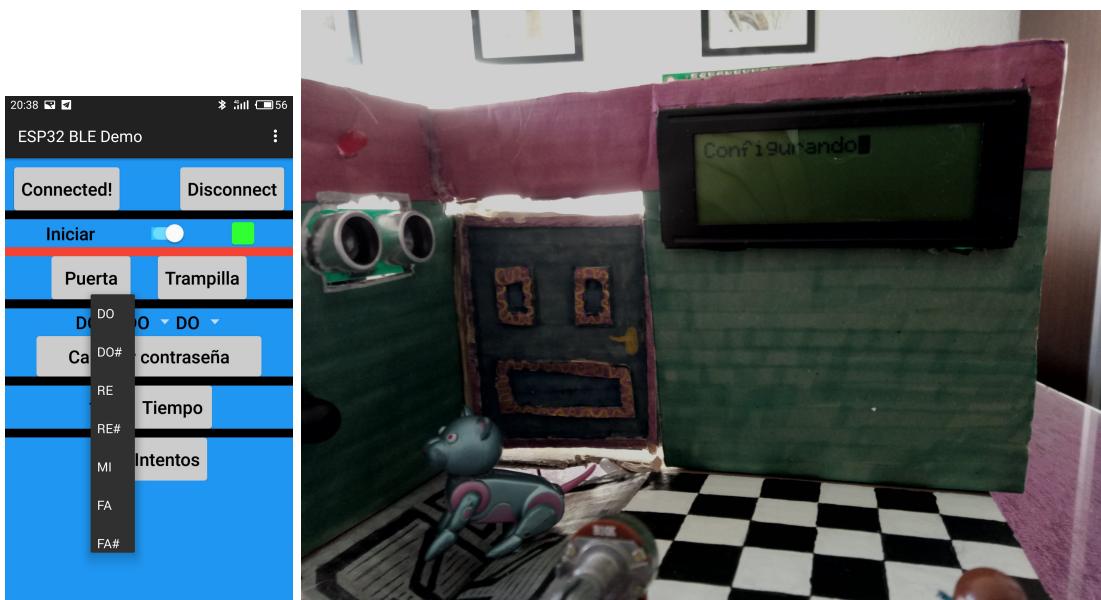
Como se puede apreciar la interfaz es bastante simple, con unos pocos botones se realiza toda la configuración.

- 1.2 Para iniciar la conexión con el módulo de bluetooth pulsamos el botón **"Connect"** y seleccionamos la dirección MAC del módulo **ESP32**. Una vez conectado cambiará el texto del botón.

- 1.3 Para iniciar la conexión con el Psoc pulsamos el botón **"Iniciar"**, cuando haya conectado correctamente el cuadrito de color **rojo** se pondrá en **verde** y aparecerá por la pantalla LCD del sistema que se ha establecido la conexión. A partir de este punto podremos utilizar el resto de botones de la aplicación.



- 1.4 La configuración de cada uno de los parámetros se puede hacer en cualquier orden, basta con seleccionar el valor deseado para ese parámetro y pulsar el botón correspondiente.



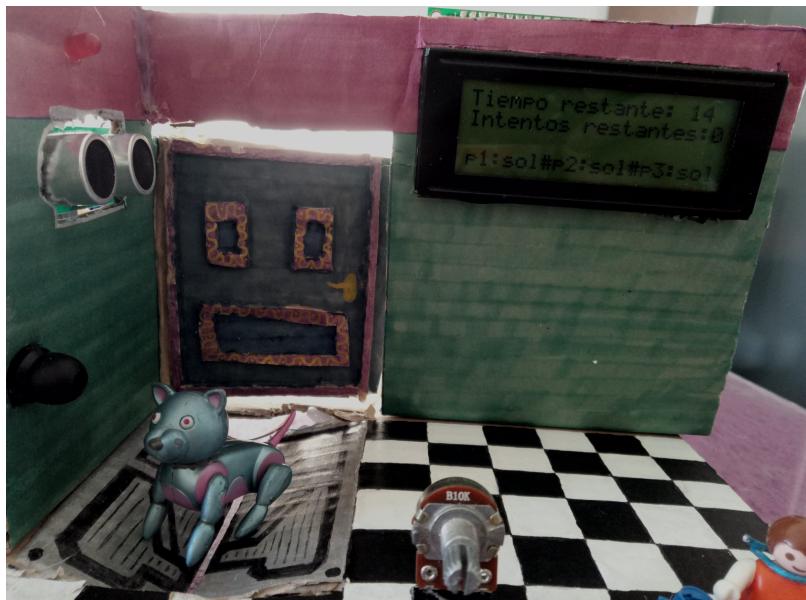
- 1.5 Además podemos abrir/cerrar de forma manual la puerta o la trampilla con los botones **"Puerta"** y **"Trampilla"** desde la aplicación si lo deseamos.

Una vez configurados los tres parámetros el sistema pasará a la etapa de **funcionamiento** como hemos citado con anterioridad.

- 2.1 En dicha etapa, esperará a que alguien se acerque a la puerta (midiendo mediante el sensor de ultrasonidos).

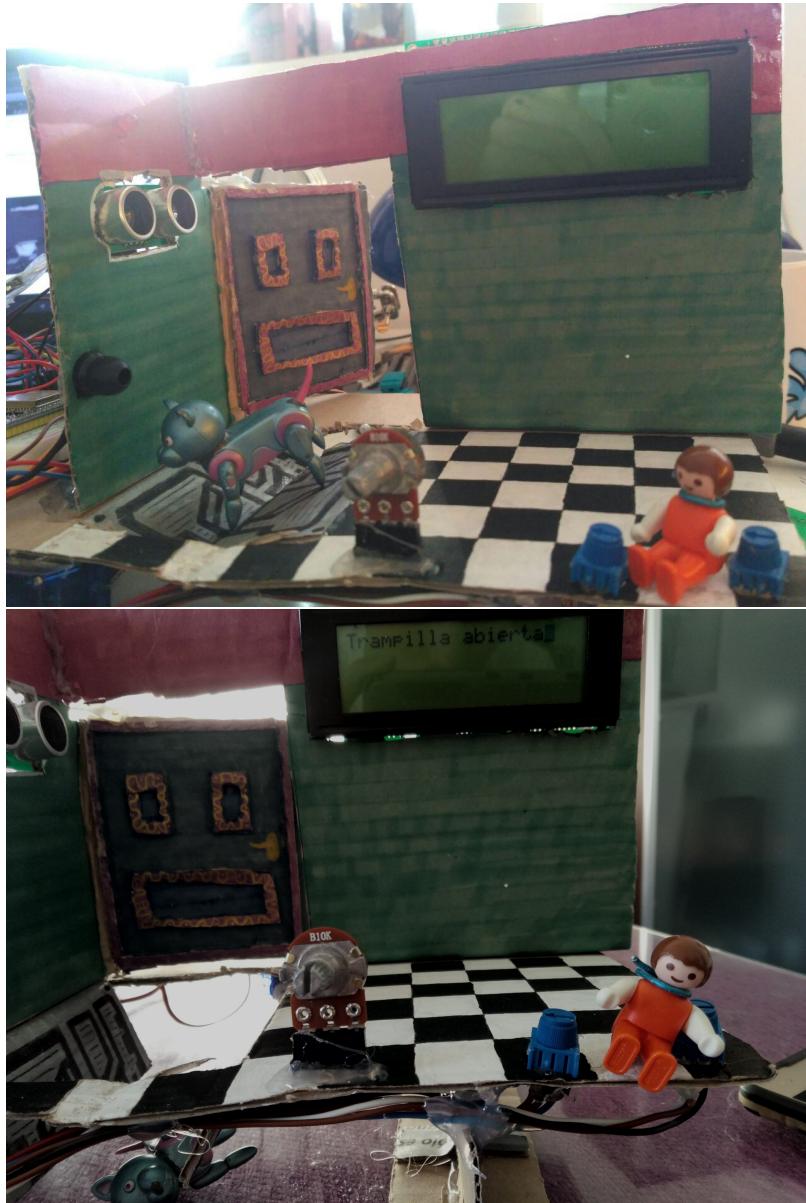


- 2.2 Una vez se halla detectado algo cerca de la puerta comenzará la cuenta atrás para colocar la contraseña mediante los potenciómetros.



- 2.3 Cuando se acaba el tiempo, si la contraseña es correcta se **abrirá** la puerta durante 5 segundos. Sin embargo, si la contraseña introducida es incorrecta se **restará un intento**. Cuando los

intentos lleguen a 0 se **abrirá** la trampilla. Si por cualquier casual, se abre la trampilla o la puerta desde la aplicación se **reinician** los intentos restantes.



### 3. Diagrama del sistema

Para explicar el sistema vamos a hacerlo haciendo alusión a dos partes: la primera corresponde al diagrama del circuito electrónico del sistema y en él se muestran los componentes **físicos** que se emplean. En la segunda parte se mostrará toda la **lógica** que infiere *PSoC 5LP* para poder utilizar todos los sensores y actuadores, así como proveer de mecanismos de comunicación a todos los componentes físicos del circuito.

#### 3.1. Montaje del Circuito

El circuito electrónico que utiliza el sistema es el siguiente:

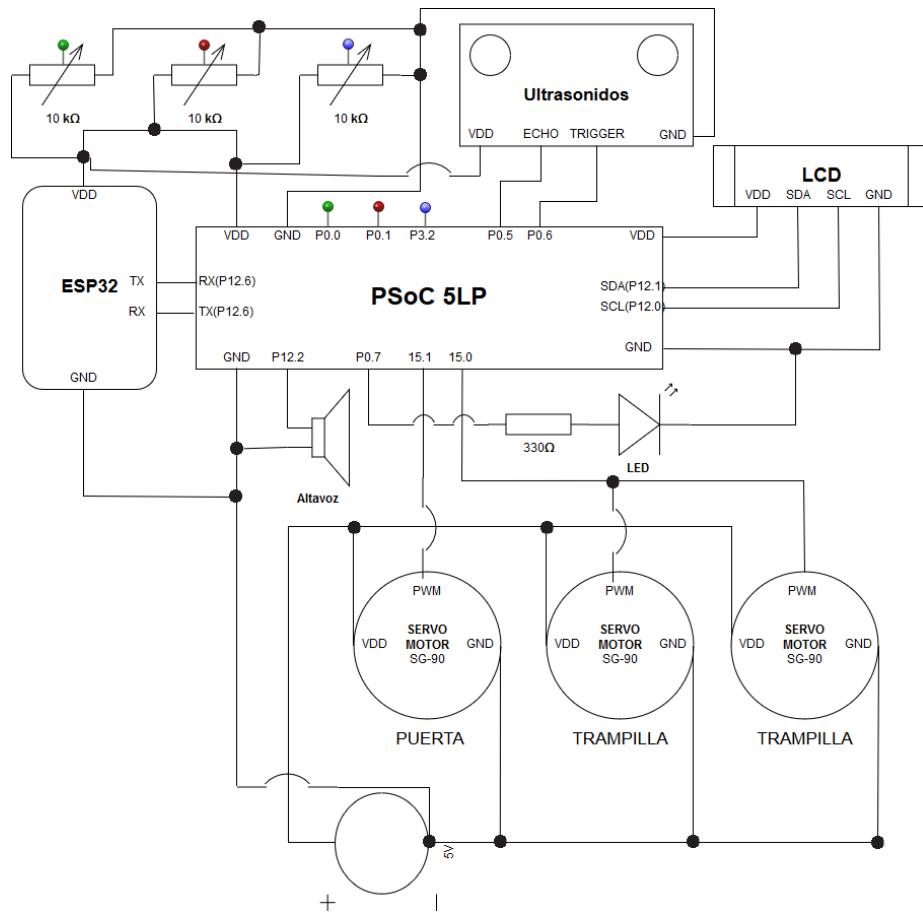


Figura 1: Circuito electrónico del sistema

Comentaremos brevemente algunas características del circuito de mayor relevancia:

- Por comodidad y para reducir la complejidad, las terminales centrales de los potenciómetros de  $10\text{ k}\Omega$  están conectadas a los pines **P0.0**, **P0.1** y **P3.2** para no tener que cruzar mucho cable dichas conexiones se representan utilizando unos pines de colores.
- El **PSoC 5LP** provee de energía a todos los sensores (LCD, Ultrasonidos, altavoz y LED) pero es incapaz de alimentar los tres servos por lo que una batería externa es necesaria para su correcto funcionamiento.
- el **PSoC** y el **ESP32** reciben la energía a través de su conexión USB.

### 3.2. Componentes lógicos inferidos por PSoC y su funcionamiento

A continuación se comentará la distinta lógica inferida por el PSoC para poder controlar los sensores, actuadores e incluso la comunicación con el ESP32 y la pantalla LCD.

#### 3.2.1. PWM

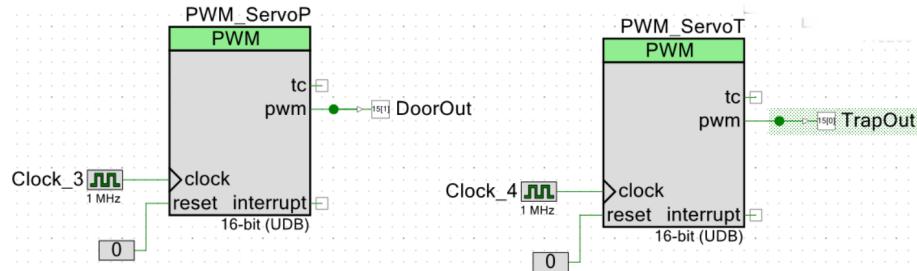


Figura 2: PWM para control de servos

Ambos componentes son idénticos, uno controla el servo de la puerta (**PWM\_ServoP**) y otro controla los dos servos que abren la trampilla (**PWM\_ServoT**). Los servos que utilizamos tienen tres pines, dos para alimentación y un tercero que recibe una señal PWM cuadrada y positiva (en el rango 0V - 5V) a una frecuencia de 50Hz y en función del ancho del pulso de dicha frecuencia el servo se mueve una cantidad de grados desde 0 hasta 180. Dicho ancho está en el intervalo  $[1000,2000]\mu\text{s}$ , siendo  $1000 \mu\text{s}$  equivalente a  $0^\circ$  y  $2000 \mu\text{s}$  a  $180^\circ$ , y cualquier valor intermedio supone un movimiento intermedio entre  $[0,180]^\circ$ .

Nosotros hemos insertado un reloj de 1 MHz al PWM para que sobre este, el propio PWM haciendo uso de un contador de 16 bits genere una señal de 50 Hz y desde el software que ejecuta PSoC hemos programado el código necesario para fijar el valor dentro del contador que queremos que el pulso esté en positivo, consiguiendo así ajustar el ancho del pulso según nuestras preferencias y conseguir el movimiento que deseemos.

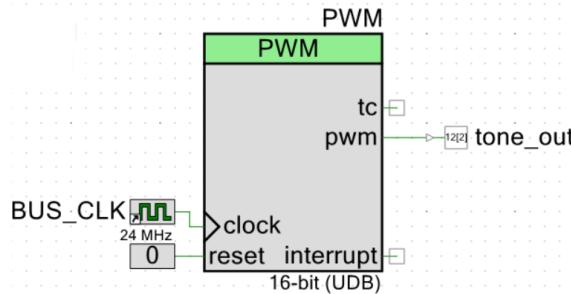


Figura 3: PWM para producir sonidos

Finalmente hemos hecho uso de un tercer PWM para poder reproducir las notas musicales. Las notas son audibles porque se producen como consecuencia de una vibración de una determinada frecuencia en el aire, por tanto basta con conectar en el canal de un altavoz un cable que emita las frecuencias deseadas, conectar el altavoz a alimentación y esté comenzará a reproducir la nota de dicha frecuencia. Con este PWM generaremos las frecuencias deseadas modificando el valor del contador interno desde el programa y utilizamos el ancho del pulso para subir o bajar el volumen del mismo.

### 3.2.2. Timer para medir distancias

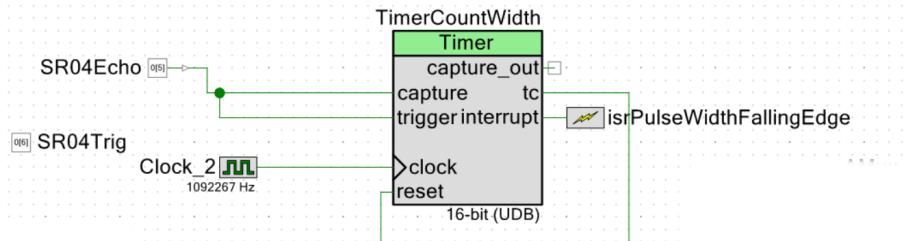


Figura 4: Timer para medir distancias

El sensor de ultrasonidos que utilizamos para medir la distancia a la que está un objeto de la puerta funciona con dos pines (los otros dos que tiene se usan para proveerle de alimentación), uno de los pines es el de **Trigger** o disparador que es una entrada que cuando recibe un único pulso cuyo ancho es de  $10\mu s$  el sensor entiende que queremos una medición y comienza a producir ultrasonidos inaudibles que viajan por el espacio y vuelven en caso de encontrarse un obstáculo. El otro pin es el pin de salida **Echo** y cuando no está midiendo produce un 0 lógico (el pin se encuentra a 0V), en cambio cuando comienza una medición este pin emite un 1 lógico (el pin se encuentra a 5V) y deja la salida en este estado hasta que recibe de vuelta los ultrasonidos. Por tanto este timer está inactivo mientras el sensor no está midiendo, en el momento en el que este comienza una medición y pone la salida de **Echo** a 1, el timer se activa y almacena un valor inicial, a partir de este momento va decrementando dicho valor a la frecuencia de un reloj muy rápido para que la precisión sea muy alta y cuando el sensor termina la medición y pone la salida **Echo** a 0 la cuenta se detiene y almacena un segundo valor final, con ambos valores el de inicio menos el del final, obtenemos el número de veces que se ha decrementado el primer valor capturado y como conocemos la frecuencia a la que se han hecho dichos decrementos, podemos saber el tiempo transcurrido desde que el pin se puso a 1 y después a 0 y por ende la distancia a la que se encuentra el objeto aplicando algunas operaciones más.

### 3.2.3. ADC Delta-Sigma

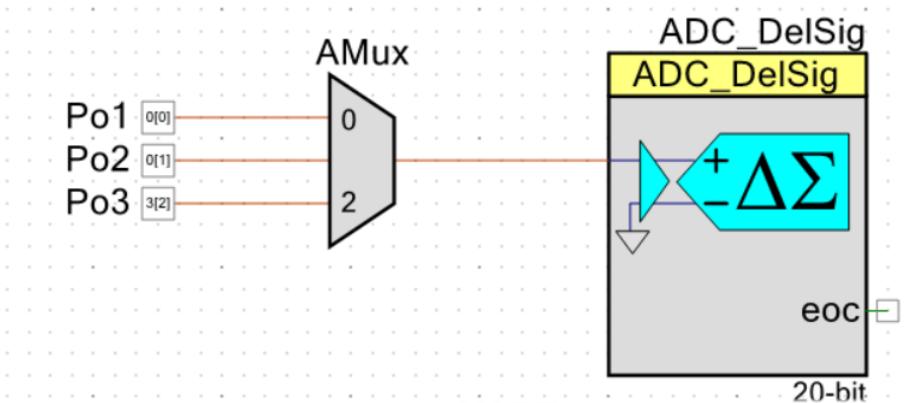


Figura 5: ADC para extraer rangos de voltaje de los potenciómetros

El conversor analógico a digital delta sigma, nos sirve para obtener con bastante precisión los rangos de voltaje que los tres potenciómetros del sistema otorgan cuando la perilla de estos se

desplaza en un sentido o en otro. Los potenciómetros los utilizamos para mapear las diez notas musicales, es decir el valor de 0V de un potenciómetro indica que la nota musical en cuestión es un **do** mientras que si el potenciómetro está dando 5V la nota musical que codifica es **la** y en el rango ]0, 5[ V codifica el resto de notas.

### 3.2.4. Master I2C para comunicación con la LCD

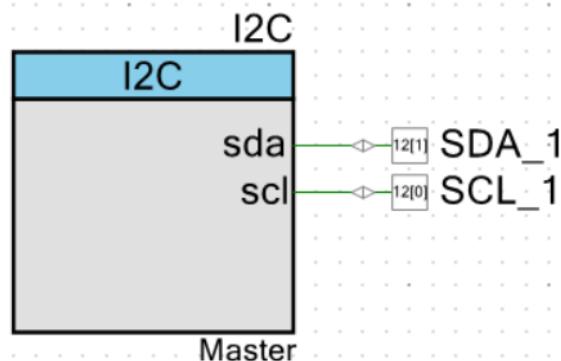


Figura 6: I2C que funciona como Maestro de la comunicación

Hacemos uso de un bloque lógico I2C que hace de maestro en la comunicación, para poder controlar una pantalla LCD de 20x4 caracteres con módulo I2C (la pantalla juega el rol de esclavo en la comunicación). Y básicamente hemos implementado una pequeña biblioteca para su control.

### 3.2.5. Interrupciones

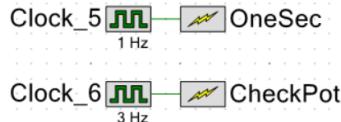


Figura 7: Interrupciones destacables del sistema

Mientras programábamos el sistema vimos que ciertas actividades, como podía ser iniciar una medición en el sensor de ultrasonidos o comprobar el valor de los potenciómetros, necesitaban hacerse de forma periódica y realizarlas a la velocidad del micro, suponía una sobrecarga innecesaria para el sistema y derivaba en fallos en muchas ocasiones, por lo que agregamos las dos interrupciones mostradas aquí, que básicamente activan un flag cuando estas se disparan. La que funciona a 1Hz se dispara como es lógico cada segundo y la empleamos para realizar mediciones con el sensor de ultrasonidos, la interrupción que tiene conectada un reloj de 3Hz se dispara cada 0.33 segundos y la utilizamos para comprobar los valores de los potenciómetros y mostrarlos por la LCD.

### 3.2.6. UART para comunicación con ESP32

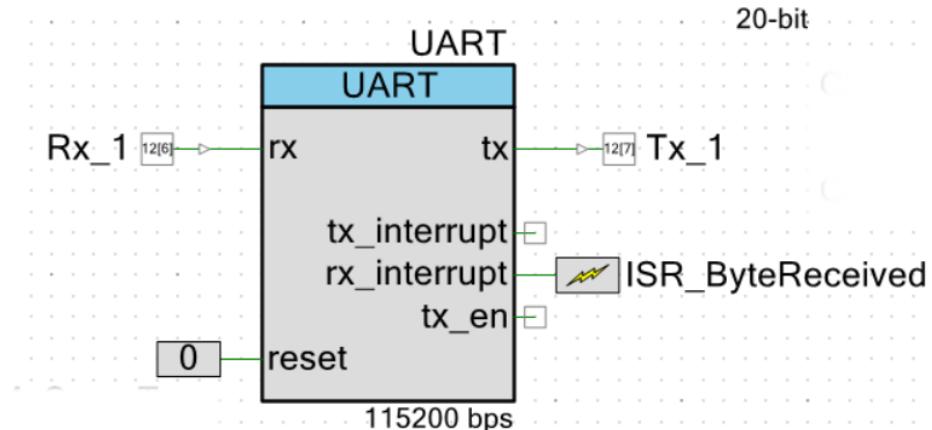


Figura 8: UART para comunicación serie con el ESP32

Para poder conectar el sistema vía bluetooth con nuestro móvil barajamos entre varias posibilidades, una de ellas era la de adquirir un módulo bluetooth y controlarlo con el PSoC pero vimos que la opción más simple era utilizar un sistema empotrado como el ESP32 que básicamente está compuesto por un microcontrolador al que le han conectado un módulo de bluetooth y además otro de WiFi y que se programa con el IDE de [Arduino](#). Para la comunicación conectamos el pin Rx del PSoC al pin Tx del ESP32 y el pin Tx del PSoC al pin Rx del ESP32 y ya está todo listo. La recepción en la parte del PSoC se realizan mediante una interrupción, cuando se recibe un byte por el pin Rx esta interrupción se dispara para activar un flag que indica que hay un byte disponible para su consumo. El protocolo de comunicación que hemos implementado entre el PSoC y el ESP32 se discute en mayor detalle en la sección ["Protocolo de comunicación"](#).

## 4. Diagrama de estados del sistema

En este punto realizaremos una breve descripción de la funcionalidad del sistema, el sistema se ha programado como un autómata de estados que cambia entre estados cuando un evento determinado ocurre, el autómata del sistema es el siguiente:

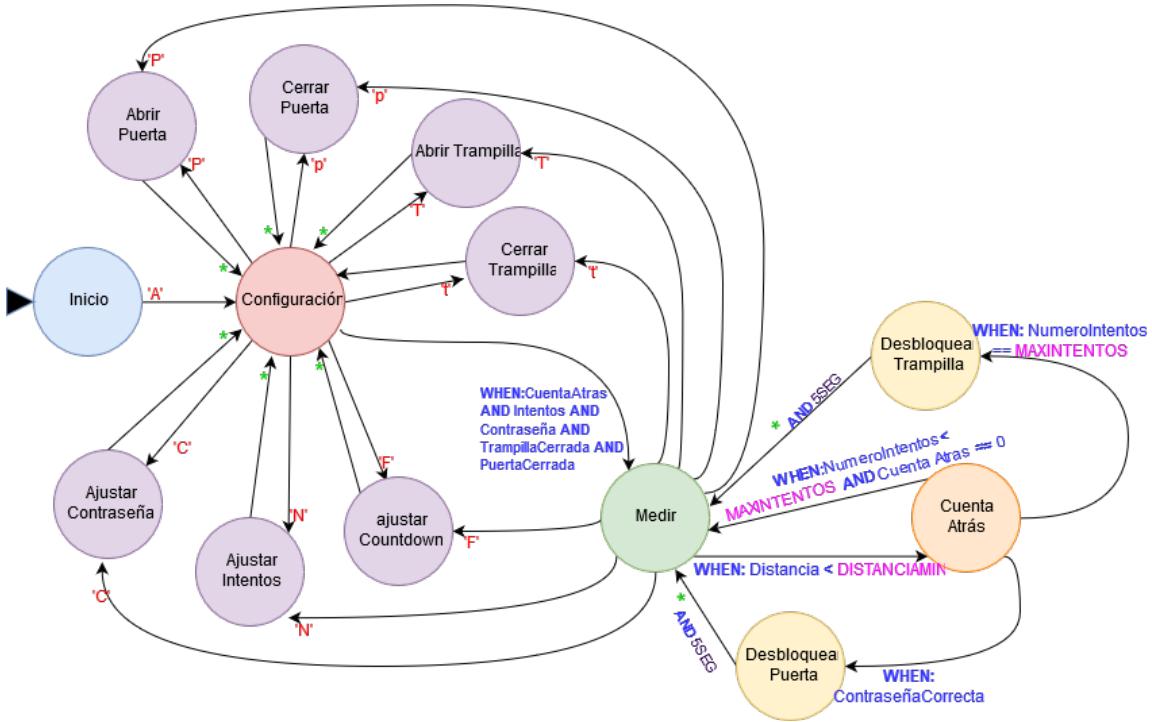


Figura 9: Autómata de estados del sistema

De este diagrama hay que tener muy claras la siguientes características:

- La flechita en color **negro** indica que ese es el estado inicial de partida del sistema.
- Los eventos en color **rojo** y con comillas simples, significan que el PSoC ha recibido por medio de la **UART** el carácter que se muestra, por ejemplo si estamos en el estado de *configuración* y se produce el **evento 'N'** (lo que significa que hemos recibido por la UART el carácter N desde el ESP32) pasamos al estado *Ajustar Intentos*.
- Los eventos en color **verde** representados con un \* simbolizan que se produce la acción descrita por el propio estado, por ejemplo si estoy en el estado *Ajustar Intentos* y en dicho estado entre el PSoC y el ESP32 se ajusta el número de intentos de forma correcta volvemos al estado de configuración. Esto lo hemos realizado así para simplificar el diagrama y no llenarlo de texto por todas partes.
- Los eventos en color **azul** indican condiciones que se deben de satisfacer. Por ejemplo si me encuentro en el estado *Desbloquea Trampilla* y la trampilla se desbloquea y trascurren 5 segundos vuelvo al estado *Medir*.
- El texto en color **rosa** representa variables que guardan información critica del sistema.
- Falta un estado de error, pero puesto que desde casi cualquier estado hay posibilidad de llegar a este estado, no se ha incluido por simplicidad del diagrama.

Una vez que entendemos el diagrama pasaremos a explicar qué realiza cada estado:

- **Inicio:** Estado inicial del sistema, en dicho estado se sincronizan el PSoC y el ESP32.
- **Configuración:** Estado en el que se permanece hasta que se haya configurado el sistema completo.
- **Abrir Puerta:** Se abre la puerta.
- **Cerrar Puerta:** Se cierra la puerta.
- **Abrir Trampilla:** Se abre la trampilla
- **Cerrar Trampilla:** Se cierra la trampilla.
- **Ajustar la Contraseña:** Se ajusta la contraseña, almacenando las tres notas musicales pertinentes en el sistema.
- **Ajustar Intentos:** Se fija el número de intentos que se tienen para introducir la contraseña.
- **ajustar CountDown:** Se fija el tiempo en segundos de la cuenta atrás.
- **Medir:** En este estado se realizan mediciones en los ultrasonidos cada segundo para detectar si hay o no un objeto cerca.
- **Cuenta atrás:** En este estado se muestra en la pantalla la cuenta atrás que se tiene para introducir la contraseña, cada 0.33 segundos se toman medidas de los tres potenciómetros y se muestra la nota musical que codifica en la pantalla.
- **Desbloquea Trampilla:** Se abre la trampilla durante un periodo de tiempo.
- **Desbloquea Puerta:** Se abre la puerta durante un periodo de tiempo.

## 5. Protocolo de comunicación

Para la comunicación de los diferentes dispositivos hemos conectado un módulo **ESP32** con un servidor de bluetooth para la aplicación (app) compatible con **Android**. El protocolo de comunicación entre la **APP**, el **ESP32** y el **PSoC** se explica en el siguiente diagrama.

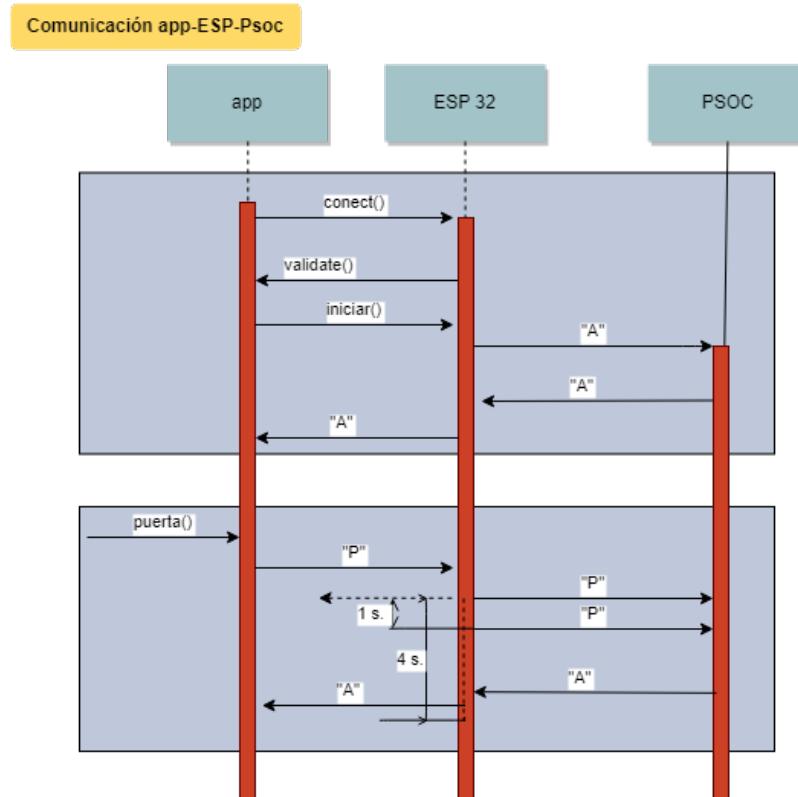


Figura 10: Diagrama de comunicación global del sistema

Al pulsar el botón "**Connect**", la **aplicación** buscará los dispositivos disponibles y nos sincronizará con el **ESP32**. Al pulsar "**Iniciar**", mandará una '**A**' para saber si el **PSoC** está listo, si es así responderá con otra '**A**'.

Para el resto de botones, la aplicación enviará todo el mensaje como en una especie de paquete (envia del tirón carácter a carácter). El **ESP32** almacenará todos los caracteres en un **buffer** ya codificados para ser enviados. Y para asegurarnos de que el mensaje se recibe, en caso de no obtener respuesta por parte del **PSoC** reenviará el mensaje cada segundo durante 4 veces hasta recibir una respuesta o la comunicación se aborta entendiendo que algo inesperado ocurre en el PSoC pues, este no responde. En caso de recibir dicha respuesta, el **ESP32** mandará otro **ack** para notificar a la **aplicación** que ha sido recibido.

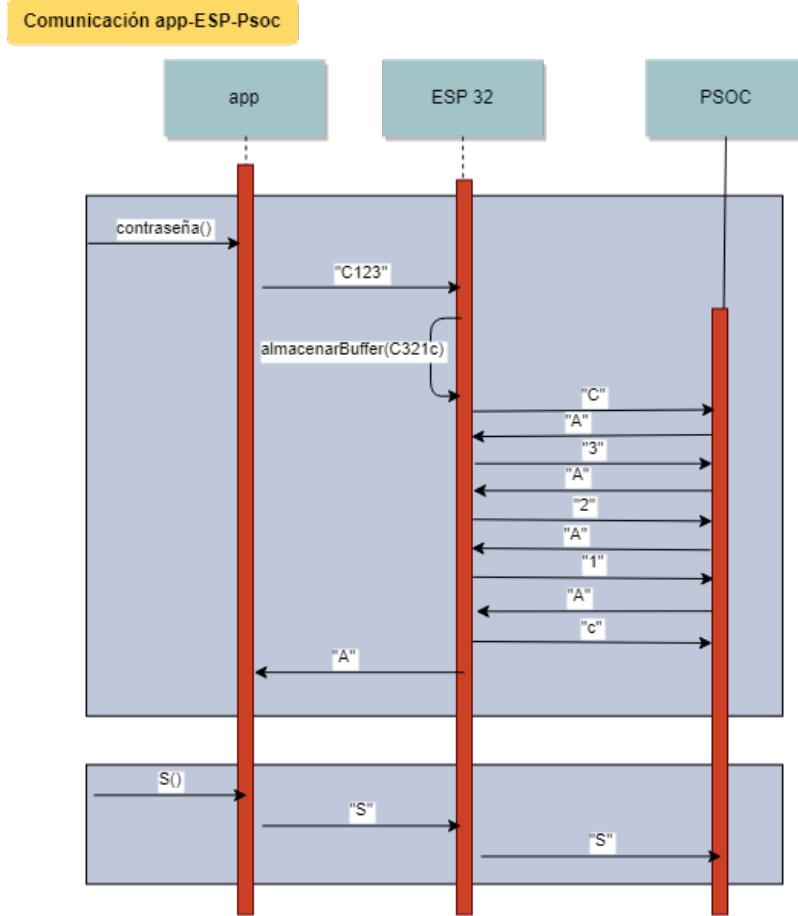


Figura 11: Ejemplo de envío de un ”paquete”

En el caso de la contraseña, tiempo e intentos, se almacenarán en el **buffer** codificándolo de tal manera que se envíe de forma **”X321x”**, carácter a carácter (primero el menos significativo).

Al recibir el último **ack**, se envía otro a la aplicación.

Para la enviar el mensaje de puesta en marcha del PSoC, tras ocurrir un error en este mandamos el carácter **”S”**, este mensaje, no se almacena en el **buffer**, lo manda directamente y no espera ningún **ack**. Los mensajes de cada botón se especifican abajo. Siendo x números entre 0 y 9:

Mensaje a enviar	Aplicacion envia a ESP32	ESP32 envia a PSoC
Acknowledgement	A	A
Puerta	P(abre)/ p(cierra)	P/p
Trampilla	T(abre)/t(cierra)	T/t
Enviar Contraseña	Cxxx	Cxxxc
Enviar Tiempo	Fxx	Fxxf
Enviar Numero de Intentos	Nx	Nxn

Cuadro 1: Mensajes del sistema