UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Cerradura Electrónica Inalámbrica Asistida por una Aplicación Android

TESIS

Que para obtener el título de

Ingeniero Eléctrico Electrónico

P R E S E N T A

Ricardo David Silva Flores

DIRECTOR DE TESIS

M. en A. Adalberto Joel Durán Ortega

Ciudad Universitaria, Cd. Mx., 2018

A mis padres, Adriana y Ricardo, y a mi hermano Rodrigo, este no es un logro solo mío sino nuestro.

A mi asesor, por la paciencia y la confianza que siempre me tuvo.

CONTENIDO

1 INTRODUCCIÓN

Este proyecto comenzó como una solución personal para mi hogar debido a las constantes situaciones en que nos encontramos mi familia y yo al quedarnos fuera de casa dejando las llaves dentro, viéndonos obligados a hacer uso de los servicios de cerrajería ocupando tiempo y dinero, o las veces en que algún miembro de la familia prestaba a otro miembro su juego de llaves y diferían en su hora de llegada, obligando al miembro prestador a esperar al otro.

Con base en estos problemas, me di a la tarea de diseñar una cerradura con la cual pudiéramos ingresar a nuestro hogar haciendo prescindible el uso de las llaves, resolviendo de esta manera nuestros problemas constantes de quedarnos fuera de casa.

La mayoría de los sistemas de seguridad hacen uso de claves numéricas o sistemas biométricos para poder abrir o cerrar, en mi cerradura casera quería hacer uso de un método diferente, algo llamativo y a su vez entretenido, llegando a la conclusión de utilizar colores como método de seguridad.

La clave consistiría en una secuencia de 6 colores donde cada usuario, integrante de mi familia, tendría su propia clave que los distinguiría de los demás, desplegando en un display LCD un mensaje de bienvenida con el nombre de cada integrante. El teclado sería una matriz de 2X2 compuesto por 4 LEDs RGB con los colores rojo, verde, azul y amarillo, sobre ellos habría un pad de botones de 2X2 que, además de fungir como botones, ayudarían a difundir mejor los colores de los LEDs.

Como medidas de seguridad se implementaría un algoritmo que intercambiara los colores aleatoriamente de lugar y un buzzer que sería activado después de un cierto número de intentos fallidos.

Después de 2 años, retomo este proyecto para desarrollarlo de manera más completa y profesional, añadiendo el popular sistema Android como herramienta de apoyo para el funcionamiento de la cerradura y uno de los microcontroladores más populares de ATMEL.

i. Objetivo General

Diseñar y construir un sistema electrónico, basado en el microcontrolador ATMEGA328P-PU de ATMEL, y adaptarlo a una cerradura convencional que pueda ser manipulada de manera inalámbrica a través de una aplicación en Android.

ii. Objetivos Específicos

- Diseñar e implementar un sistema electrónico que sea capaz de manipular una cerradura convencional.
- Diseñar e implementar una aplicación basada en el sistema operativo Android, la cual sea capaz de manipular el sistema electrónico de la cerradura.
- Implementar un medio inalámbrico por el cual ambos sistemas, la cerradura y la aplicación Android, puedan comunicarse.
- Diseñar y programar los métodos de seguridad necesarios para la manipulación de todo el sistema.
- Optimizar el sistema para que sea de bajo consumo energético.
- Diseñar y construir el prototipo de la cerradura integrando ambos sistemas.

iii. Justificación

El proyecto que aquí se plantea surge como una solución a los problemas más comunes que presentan las cerraduras convencionales y a los inconvenientes que poseen en cuanto a seguridad. Aprovechando las ventajas de los celulares inteligentes y, en conjunto con las bondades que actualmente brindan los microcontroladores de ATMEL, se desarrolló una solución práctica, innovadora y de bajo consumo.

iv. Hipótesis

Con esta tesis se pretende demostrar que, mediante un sistema electrónico acoplado a una cerradura convencional y modificando mínimamente la instalación de la misma cerradura, se puede mejorar la seguridad, resolviendo de esta manera algunas de las carencias de los sistemas actuales.

Además, se pretende hacer prescindible el uso de las llaves mediante la utilización de un dispositivo que es manejado día a día por la mayoría de las personas como son los smartphones, específicamente los que están basados en el sistema operativo Android. Aprovechando las características de este sistema se pretende también monitorear las entradas y salidas realizadas.

2 MARCO TEÓRICO

En este capítulo abordaremos algunos conceptos técnicos y teóricos que nos ayudarán a comprender los criterios utilizados a lo largo de esta Tesis, no es la intención abarcar en detalle cada tópico involucrado, pero sí la de dar una idea general sobre los elementos involucrados y sus particularidades.

2.1 COMUNICACIÓN BLUETOOTH

La tecnología inalámbrica Bluetooth™ [1] se ha convertido en algo muy popular debido a la amplia aplicación que tiene en nuestra vida diaria, la podemos encontrar en smartphones, tabletas, computadoras, sistemas de audio e incluso en sistemas médicos.

Diseñado por el Bluetooth SIG (Special Interest Group)[2], esta tecnología funciona a través de ondas de radio de baja potencia. Comunicándose entre los 2.4 GHz y los 2.485 GHz, la tecnología inalámbrica Bluetooth trabaja en la banda ISM, que es una banda de frecuencias reservada para aplicaciones industriales, científicas y médicas.

Una de las maneras en que los dispositivos Bluetooth evitan interferir con otros sistemas en estas frecuencias es enviando señales muy débiles, aproximadamente de 1 mW, limitando así el alcance de estos dispositivos.

Los dispositivos con tecnología Bluetooth pueden conectarse con hasta 8 dispositivos simultáneamente.

Para evitar la interferencia que pudiera ocurrir entre los dispositivos se utiliza una técnica llamada “espectro ensanchado por salto de frecuencia” que hace improbable que más de un dispositivo esté transmitiendo en la misma frecuencia al mismo tiempo.

Con esta técnica, un dispositivo utilizará 79 frecuencias individuales y aleatorias dentro de un rango asignado, cambiando de una a otra en una base regular. En el caso de la tecnología inalámbrica Bluetooth, los transmisores cambian de frecuencia 1,600 veces cada segundo, lo que significa que más dispositivos pueden hacer pleno uso de una porción limitada del espectro de radio.

[1] Bluetooth es una marca que pertenece a Bluetooth SIG, Inc., USA.
[2] Grupo conformado inicialmente por Ericsson, IBM, Intel, Nokia y Toshiba para desarrollar y promover una solución global para la comunicación inalámbrica de corto alcance, actualmente lo conforman más de 30,000 miembros.

Los sistemas Bluetooth crean redes de área personal, personal-area network (PAN), o piconets. Una vez que una piconet es establecida, los miembros aleatoriamente saltan de frecuencias al unísono para poder estar en contacto unos con otros y así evitar que otras piconets interfieran en la comunicación. Cada piconet salta aleatoriamente a través de las frecuencias disponibles, por lo tanto, todas las piconets están separadas entre ellas. Si las piconets resultan estar a la misma frecuencia, la confusión resultante sólo se dará por una pequeña fracción de segundo, y el software designado para corregir estos errores eliminará la información de confusión y continuará con las tareas de la red. [1]

Desde la creación del Bluetooth SIG y hasta nuestros días, la tecnología inalámbrica Bluetooth ha ido evolucionando, añadiendo nuevas características y mejorando las ya establecidas.

2.2 MICROCONTROLADOR AVR ATMEGA328P-PU

Un microcontrolador es un dispositivo que integra la mayoría de las características que componen a una computadora, todo construido dentro de un solo chip. Tienen la ventaja de ser fáciles de programar y cuentan con herramientas de desarrollo en distintos lenguajes de programación como son C, ensamblador, etc.

Entre los más destacados se encuentran los microcontroladores PIC desarrollados por Microchip y los AVR desarrollados por ATMEL. Una de las familias más extensas de ATMEL son los microcontroladores de 8 bits, donde la diferencia más significativa se encuentra en la velocidad de procesamiento, en comparación de los PIC, ya que realizan una instrucción por cada ciclo de reloj mientras que los PIC necesitan 4 ciclos de reloj.

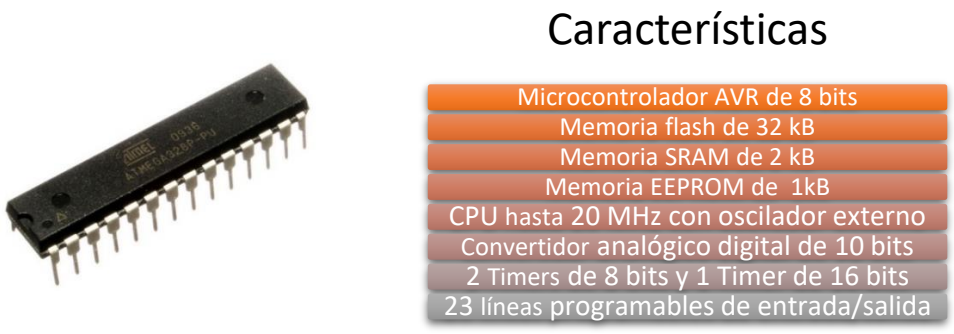


Figura 2.1 ATMEGA328P-PU.
(Fuente: https://upload.wikimedia.org/wikipedia/commons/thumb/0/0c/ATMEGA328P-PU.jpg/1200px-ATMEGA328P-PU.jpg. Último acceso: 04 de julio de 2017)

Dentro de los microcontroladores AVR de 8 bits se encuentra la familia megaAVR donde se encuentra la serie ATMEGA, la cual es de propósito general. Dentro de esta serie se

encuentra el ATMEGA328P-PU el cual cuenta con las características mostradas en la Figura 2.1.

Se optó por este microcontrolador principalmente por el tamaño de la memoria EEPROM, su bajo costo, su fácil obtención en el mercado y por todo el soporte que se obtiene con las herramientas de desarrollo de ATMEL.

2.3 Comunicación Serial

Para que dos o más sistemas puedan intercambiar información deben compartir el mismo protocolo de comunicación. Uno de los protocolos más conocidos es la comunicación serial, esta interfaz transmite los datos un solo bit a la vez. Estas interfaces pueden operar con tan solo un cable, generalmente no más de cuatro (Figura 2.2).

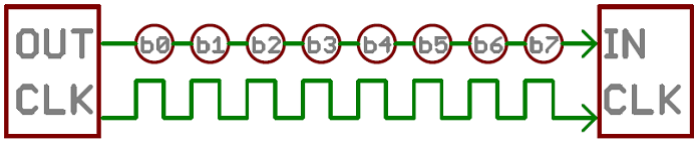


Figura 2.2 Ejemplo de una interfaz serial, transmitiendo un bit cada pulso de reloj.
(Fuente: https://learn.sparkfun.com/tutorials/serial-communication#rules-of-serial. Último acceso: 04 de diciembre de 2017)

Serial Asíncrono

Este protocolo es una de las formas más comunes de transmisión asíncrona. Asíncrono quiere decir que los datos son transferidos sin el apoyo de una señal de reloj externa. Este método de transmisión es perfecto si se requiere minimizar la utilización de cables y pines de entrada/salida.

El protocolo serial asíncrono cuenta con una serie de reglas y mecanismos que ayudan a garantizar grandes transferencias de datos y libres de errores. Estos mecanismos, que son necesarios al eliminar la señal de reloj externo, son:

- Bits de datos
- Bits de sincronización
- Bits de paridad
- Baud rate

El protocolo es altamente configurable. La parte crítica es asegurarse que ambos dispositivos dentro de un bus serial están configurados con el mismo protocolo.

Baud Rate

El baud rate determina qué tan rápido los datos serán transmitidos a través de la línea serial. Generalmente se expresa en unidades de bits-por-segundo (bps). Si se invierte el baud rate, se puede calcular cuánto tiempo es necesario para transmitir un solo bit. Este valor determina cuánto tiempo tarda el transmisor en mantener una línea serial en alto o bajo, o a qué periodo el receptor muestrea esa línea.

El baud rate puede ser de cualquier valor. La única condición es que ambos dispositivos operen a la misma velocidad. En la Tabla 2.1 se presentan los baud rate más comunes.

Baud Rate [bps]	1200	2400	4800	9600	19200	38400	57600	115200

Tabla 2.1 Baud Rates estándar.
(Fuente: Elaboración propia)

Estructura de los Datos

Cada bloque, generalmente de un byte, de datos transmitidos es enviado dentro de un paquete de bits, como se muestra en la Figura 2.3, los paquetes son creados agregando bits de sincronización y de paridad a los bloques de datos.

Figura 2.3 Paquete serial.
(Fuente: https://learn.sparkfun.com/tutorials/serial-communication#rules-of-serial. Último acceso: 04 de diciembre del 2017)

Segmentos de Datos

Dentro de cada paquete serial se encuentra un segmento de datos de entre 5 o 9 bits. Ciertamente, el tamaño de datos estándar es un byte (8 bits), sin embargo, otros tamaños también tienen sus usos.

Una vez que el tamaño del segmento de datos es establecido, ambos dispositivos deben concordar en el endianness de los datos. Del bit más significativo al menos significativo (msb, Big Endian) o viceversa (lsb, Little Endian).

Bits de Sincronización

Los bits de sincronización son dos o tres bits especiales transmitidos con cada segmento de datos. De los cuales se encuentran el bit de inicio (start) y el bit o bits de alto (stop). Estos bits marcan el inicio y final de un paquete serial.

El bit de inicio siempre se indica mediante una línea de datos inactiva que va de 1 a 0, mientras que el bit(s) de alto volverá al estado inactivo manteniendo la línea en 1.

Bits de Paridad

La paridad es una forma de comprobación de errores muy simple y de bajo nivel. Se configura de dos maneras: par o impar. Para generar el bit de paridad, se suman todos los bits del segmento de datos (de 5 a 9 bits) y la uniformidad de la suma decide si el bit se encuentra configurado o no. Por ejemplo, asumiendo que la paridad está configurada en par y se ha agregado a un byte de datos como 0b01011101, que tiene un número impar de 1 (5), el bit de paridad sería configurado a 1. De manera inversa, si la paridad se encuentra configurada en impar, el bit de paridad sería 0.

La paridad es opcional y no es ampliamente usada.

9600 8N1

9600 8N1 – 9600 baud, datos de 8 bits, sin paridad, y 1 bit de alto – es uno de los protocolos seriales más comúnmente usados. A continuación, se enuncia un ejemplo de este protocolo:

Un dispositivo transmitiendo los caracteres ASCII ‘O’ y ‘K’ tendría que crear dos paquetes seriales. El valor ASCII de O es 79, que en código binario de 8 bits es representado como 01001111, mientras que el valor binario de K es 01001011.

Agregando los bits de sincronización y asumiendo que los datos se transmiten del bit menos significativo al más significativo, en la Figura 2.4 tenemos:

Figura 2.4 Paquetes de datos de las letras O y K.
(Fuente: https://learn.sparkfun.com/tutorials/serial-communication/rules-of-serial. Último acceso: 04 de diciembre de 2017)

Dado que se está transfiriendo a 9600 bps, el tiempo dedicado a mantener cada uno de esos bits en alto o bajo es $\frac{1}{9600\ bps}$ o 104 µs por bit.

Conexiones y Hardware

Un bus serial está constituido por tan solo dos cables, uno para enviar los datos y otro para recibirlos. Como tal, de acuerdo con la Figura 2.5, los dispositivos seriales deben tener dos pines seriales: el receptor, RX y el transmisor, TX.

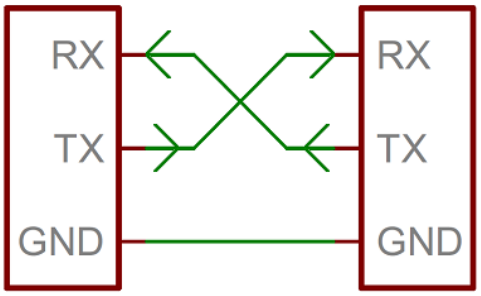


Figura 2.5 Conexión de un bus serial entre dos dispositivos.
(Fuente: https://learn.sparkfun.com/tutorials/serial-communication/wiring-and-hardware. Último acceso: 02 de diciembre de 2017)

Una interfaz serial donde ambos dispositivos pueden enviar y recibir datos es llamada full-duplex o half-duplex. Full-duplex significa que ambos dispositivos pueden enviar información simultáneamente. Half-duplex significa que los dispositivos seriales deben turnarse para enviar y recibir información.

Actualmente, existen una gran variedad de estándares con los cuales se implementa la comunicación serial. Una de las implementaciones de hardware más populares es la TTL (transistor-transistor logic).

Las señales seriales TTL están comprendidas entre el rango de alimentación de un microcontrolador, generalmente de 0V a 3.3V o 5V. Una señal al nivel de Vcc (3.3V o 5V, etc.) puede indicar una línea inactiva, un bit con valor de 1 o un bit de alto. Una señal de 0V (GND) puede representar un bit de inicio o un bit de datos con valor 0 (ver Figura 2.6). [2]

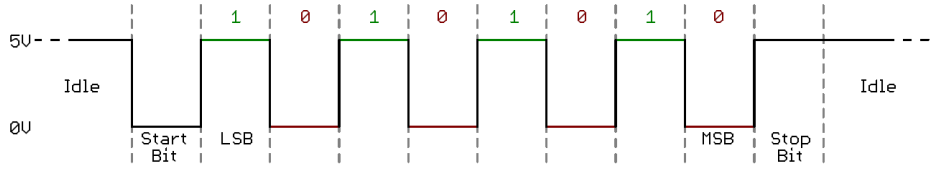


Figura 2.6 Diagrama de comunicación serial asíncrona TTL.
(Fuente: https://learn.sparkfun.com/tutorials/serial-communication/wiring-and-hardware. Último acceso: 04 de diciembre de 2017)

2.4 ANDROID

Android fue creado por un grupo de compañías conocidas como Open Handset Alliance[3], liderados por Google. Los miembros de esta alianza comparten la idea de que una plataforma abierta es necesaria y con el objetivo de generar un producto compartido que cada contribuidor pueda entallar y personalizar. Actualmente, Google es quien mantiene y desarrolla Android. [3]

Android fue construido desde cero para permitir a los desarrolladores crear aplicaciones móviles atractivas que aprovechen al máximo todo lo que un teléfono tiene para ofrecer; es un sistema operativo basado en java que corre bajo el kernel de Linux. [4]

Debido a que Android es un sistema operativo open-source, y el impacto que ha tenido sobre los dispositivos móviles ha crecido considerablemente, ha traído con ello una gran evolución (ver Figura 2.7).

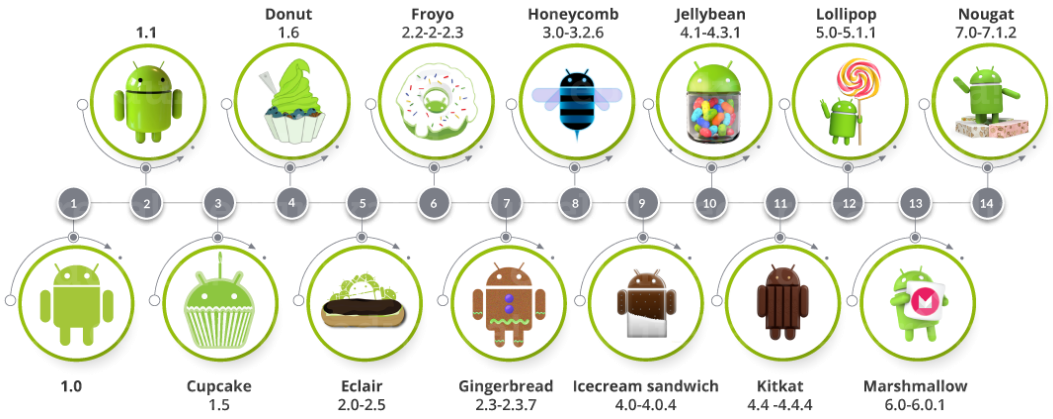


Figura 2.7 Evolución de Android.
(Fuente: https://www.edureka.co/blog/android-tutorial/#evolution. Último acceso: 14 de abril de 2018)

Para el desarrollo del proyecto se optó por utilizar esta plataforma debido a la versatilidad y popularidad que tiene en el área de los dispositivos móviles.

Para el desarrollo de la aplicación se eligió la plataforma de desarrollo MIT App Inventor[4], ya que es una plataforma muy intuitiva y visual de programación, basada en la conexión de bloques, y además, las aplicaciones desarrolladas en la misma son compatibles con la mayoría de los dispositivos basados en Android 2.3 (Gingerbread) o superior.

[3] Fundada en 2007, la OHA está constituida por 84 compañías entre los que se incluyen fabricantes de dispositivos móviles, desarrolladores de aplicaciones, operadores de comunicaciones y fabricantes de chips.
[4] Plataforma educativa para el desarrollo de aplicaciones para el sistema Android, desarrollada por el Instituto Tecnológico de Massachusetts.

2.5 LENGUAJE DE PROGRAMACIÓN C

Es uno de los lenguajes de programación de alto nivel más utilizado en el mundo. Su origen se remonta a los años 1969-1973, cuando Dennis Ritchie generó un nuevo lenguaje a partir del lenguaje "B" creado por Ken Thompson. Este nuevo lenguaje pretendía simplificar la programación de los procesadores (inicialmente solo se podían programar mediante unos y ceros) a través de reglas sintácticas mucho más parecidas a nuestro idioma natural.

En 1978, Ritchie y Brian Kernighan publicaron el primer libro de C (The C Programming Language) y, a partir de ese momento, C se convirtió en un referente para la programación. Actualmente, existen muchas pequeñas variantes de este lenguaje. Son tantas que se tuvo que crear un estándar, el ANSI, compatible con prácticamente todos los compiladores de C. El compilador es el encargado de obtener el código ejecutable a partir del código fuente, éste lee todo el fichero con el programa y lo traduce a otro fichero en lenguaje máquina, que es el que podemos ejecutar. [5]

Actualmente, existen distintas plataformas de desarrollo de software gratuitas que permiten la programación en C de manera práctica y sencilla, tales como Eclipse, Dev C++ o Code: Blocks, éstas ofrecen herramientas de depuración y compilación en ambientes totalmente gráficos.

2.6 TIPOS DE CERRADURAS

En la actualidad existen distintos sistemas enfocados a la seguridad en los hogares, desde cerraduras mecánicas convencionales hasta cerraduras con un funcionamiento eléctrico-mecánico (ver Figura 2.10) o electrónico-mecánico (ver Figuras 2.11 a 2.15).

Figura 2.8 Cerradura sobrepuesta de barra fija, FANAL. (Fuente: http://www.homedepot.com.mx/comprar/es/centro/cerradura-sobreponer-barra-fija-izq. Último acceso: 06 de junio de 2017)

Figura 2.9 Cerradura sobrepuesta, PHILIPS. (Fuente: http://www.homedepot.com.mx/comprar/es/centro/cerradura-de-sobreponer-izquierda. Último acceso: 06 de junio de 2017)

Figura 2.10 Chapa eléctrica, LLOYDS. (Fuente: http://www.homedepot.com.mx/comprar/es/centro/chapa-electrica-p-video-interfon. Último acceso: 07 de junio de 2017)

Figura 2.11 Cerradura inteligente Touch-to-open, Kwikset. (Fuente: http://www.homedepot.com.mx/comprar/es/centro/cerrojo-smart-code-niquel-satin. Último acceso: 06 de junio de 2017)

Figura 2.12 Cerrojo Smart Code, Kwikset. (Fuente: http://www.homedepot.com.mx/comprar/es/centro/cerrojo-smart-key-bluetooth-ns. Último acceso: 06 de junio de 2017)

Figura 2.13 Cerradura Smart Door Lock, SAMSUNG. (Fuente: http://www.samsungdigitallife.com/SHS-5050.php. Último acceso: 06 de junio de 2017)

Figura 2.14 Cerradura Push Pull, SAMSUNG. (Fuente: http://www.samsungdigitallife.com/SHP-DP728.php. Último acceso: 16 de junio de 2017)

Figura 2.15 Cerradura Sesame, Candy House. (Fuente: https://www.kickstarter.com/projects/candyhouse/sesame-your-key-reinvented?lang=es. Último acceso: 16 de junio de 2017)

Desde cerraduras con barras deslizables, pasando por cerraduras que utilizan tarjetas o llaveros RFID[5], hasta cerraduras biométricas que emplean la identificación por medio de huellas dactilares, son algunas de las variantes de cerraduras que se pueden encontrar en el mercado.

[5] La tecnología RFID (Radio Frequency Identification) utiliza las ondas de radio para identificar personas u objetos.

3 DISEÑO DEL SISTEMA

El sistema para la cerradura está constituido principalmente por tres subsistemas: el microcontrolador, la aplicación Android y la alimentación.

Como se observa en la Figura 3.1, para la comunicación entre el microcontrolador y la aplicación se utilizó un módulo Bluetooth; y para la interacción con los usuarios y la identificación de los distintos estados en que se podría encontrar la cerradura se emplearon distintos periféricos como interruptores, LEDs y un motor.

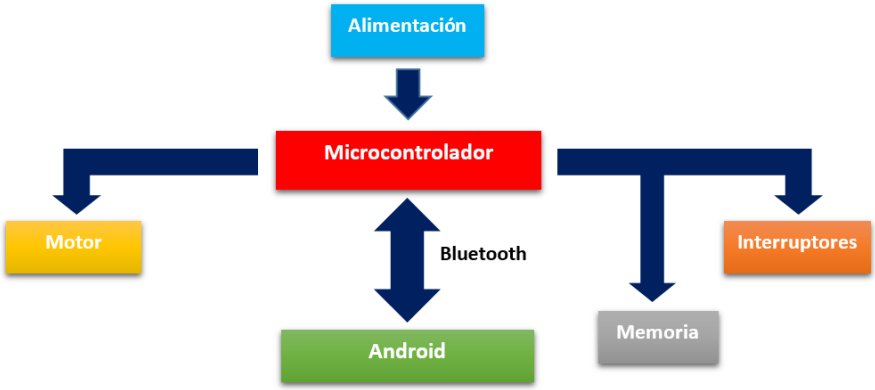


Figura 3.1 Diagrama de bloques del sistema.
(Fuente: Elaboración propia)

3.1 SELECCIÓN DEL MÓDULO PARA LA COMUNICACIÓN VÍA BLUETOOTH

Como primer paso a desarrollar, se eligió establecer la comunicación a través del módulo Bluetooth, y para ello se tomaron en cuenta ciertas condiciones que debía cumplir el módulo.

Las características deseadas para el módulo Bluetooth fueron las siguientes:

- Comunicación bidireccional entre el sistema y la aplicación Android.
- Alcance por lo menos de 10 m para tener una distancia aceptable entre el sistema y el smartphone.
- Consumo bajo de corriente.
- Que cuente con la tecnología Bluetooth Low Energy.

Bluetooth Spec. Evolution

Specifications	1.1	1.2	2.0 + EDR	2.1 + EDR	3.0 +HS	4.0
Adopted	2002	2005	2004	2007	2009	2010
Transmission Rate	723.1 kbps	723.1 kbps	2.1 Mbps	3 Mbps	24 Mbps	25 Mbps
Standard PAN Range	10 m	10 m	10 m	10 m	10 m	50 m
Improved Pairing (without a PIN)				Yes	Yes	Yes
Improved Security		Yes	Yes	Yes	Yes	Yes
NFC Support			Yes	Yes	Yes	Yes

Bluetooth Feature Evolution

Specifications	1.1	1.2	2.0 + EDR	2.1 + EDR	3.0 + HS	4.0
Voice Dialing	Yes	Yes	Yes	Yes	Yes	Yes
Call Mute	Yes	Yes	Yes	Yes	Yes	Yes
Last-Number Redial	Yes	Yes	Yes	Yes	Yes	Yes
Fast Transmission Speeds			Yes	Yes	Yes	Yes
Lower Power Consumption			Yes	Yes	Yes	Yes
Bluetooth Low Energy						Yes

Figura 3.2 Evolución del Bluetooth.
(Fuente: https://www.coursera.org/learn/wireless-communication-technologies/lecture/xbvzP/bluetooth. Último acceso: 08 de junio de 2017)

Después de analizar las distintas versiones de Bluetooth que se encontraban disponibles en el mercado y sus características, se concluyó que la versión 4.0 cumplía con los atributos solicitados (ver Figura 3.2).

El dispositivo seleccionado para esta tarea fue el módulo Bluetooth HM-10, que a continuación se definirá con mayor detalle.

Módulo Bluetooth HM-10

El módulo HM-10 integra la versión de Bluetooth 4.0 basado en el circuito integrado CC2541F256 desarrollado por Texas Instruments, que permite realizar una comunicación vía inalámbrica a través del protocolo de comunicación serial, como se explica en el capítulo 2.3.

Las características principales del dispositivo son las siguientes:

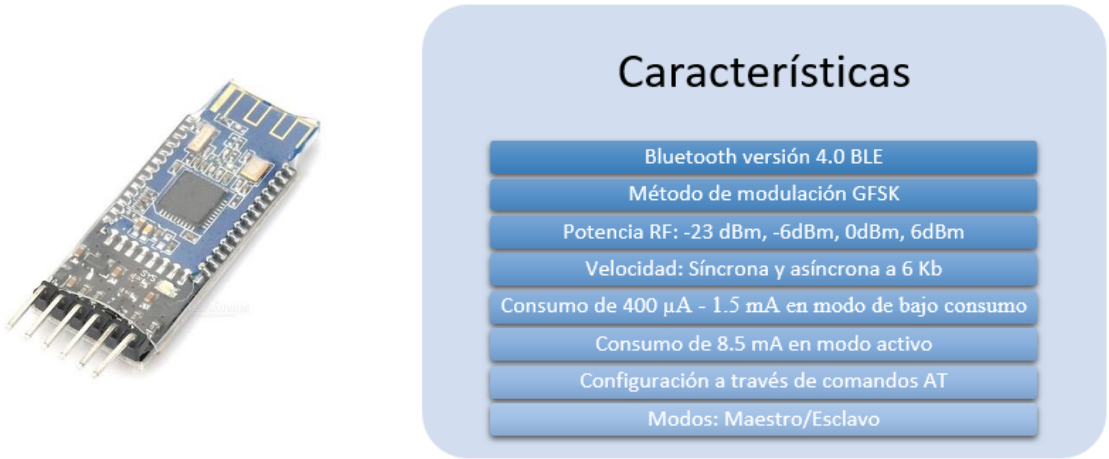


Figura 3.3 Módulo Bluetooth HM-10.
(Fuente: http://www.etechpk.net/wp-content/uploads/2016/08/hm10.jpg. Último acceso: 14 de junio de 2017)

El circuito integrado CC2541F256 cuenta con un total de 40 pines en un empaquetado RHA, la asignación de pines puede observarse en la Figura 3.4.

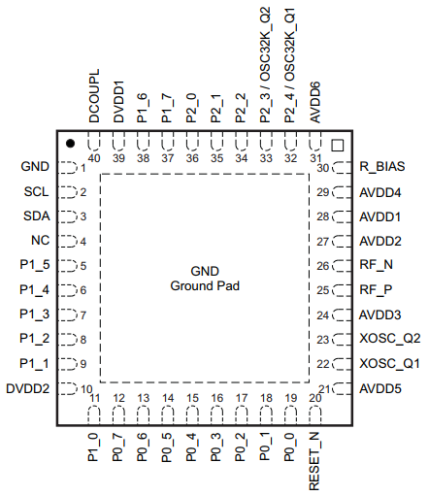


Figura 3.4 Asignación de pines del CC2541F256 (visto desde arriba).
(Fuente: http://www.ti.com/lit/ds/symlink/cc2541.pdf. Último acceso: 14 de junio de 2017)

Dado que para el desarrollo del proyecto se utilizó el módulo HM-10, como se muestra en la Figura 3.3, el sistema se enfocó únicamente en 5 pines (ver Tabla 3.1).

Nombre del PIN	PIN	Tipo de PIN	Descripción
GND	1	Pin de tierra	Conectar a tierra
P1_6	38	Digital I/O	Puerto 1.6
P1_7	37	Digital I/O	Puerto 1.7
P1_1	9	Digital I/O	Puerto 1.1 (20mA máx.)
DVDD2	10	Power (Digital)	Voltaje de alimentación 2V – 3.6V

Tabla 3.1 Funciones de los pines utilizados.
(Fuente: Elaboración propia)

Los pines P1_6 y P1_7 corresponden a la interfaz de comunicación serial, el pin P1_1 corresponde a un pin de estado, que se encargará de indicar si algún dispositivo se encuentra conectado con el módulo Bluetooth, y el pin DVDD2 que corresponde a la alimentación del circuito integrado.

Potencia de la Señal

La potencia de la señal del Bluetooth se puede configurar a través del comando AT, “AT+POWE”, donde se puede elegir entre 4 distintos niveles de potencia, mostrados en la Tabla 3.2.

AT+POWE	mA	dBm
0	8.44	-23
1	8.45	-6
2	8.47	0
3	8.47	6

Tabla 3.2 Ajustes de potencia de la señal para el módulo HM-10.
(Fuente: Elaboración propia)

3.2 DISEÑO DEL SOFTWARE

El software del sistema está comprendido por dos partes:

- Programación del microcontrolador ATMEGA328P-PU
- Programación de la aplicación Android

En ambos casos se describirán los procesos y métodos desarrollados para el correcto funcionamiento del sistema, así como también las estrategias que se tomaron para manejar los datos y la respuesta a las peticiones que el usuario podría realizar.

3.2.1 Programación del ATMEGA328P-PU

Para la programación del microcontrolador se plantearon diversas funciones básicas necesarias que debía cumplir el sistema, de tal manera que pudiera cubrir las mismas funciones que una cerradura básica.

- Establecer código de entrada/salida
- Cambiar código de entrada/salida
- Limitar número de intentos de acceso
- Configuración de la comunicación Bluetooth

Una vez cubiertas las funciones básicas de una cerradura, se agregaron otras características basadas en los dispositivos utilizados y las necesidades que los usuarios podrían tener.

- Identificación de usuarios
- Creación de un administrador
- Registro de cada usuario a través de contraseña
- Bloqueo de algún usuario
- Eliminación de algún usuario
- Cambio de contraseña de registro
- Cambio del nombre del módulo Bluetooth

- Cambio del pin de vinculación del módulo Bluetooth
- Utilización de clave de emergencia
- Salida de emergencia
- Creación de un perfil de bajo consumo de energía

Con base en las funciones establecidas, se programó el microcontrolador de acuerdo con el diagrama de flujo presentado en la Figura 3.5.

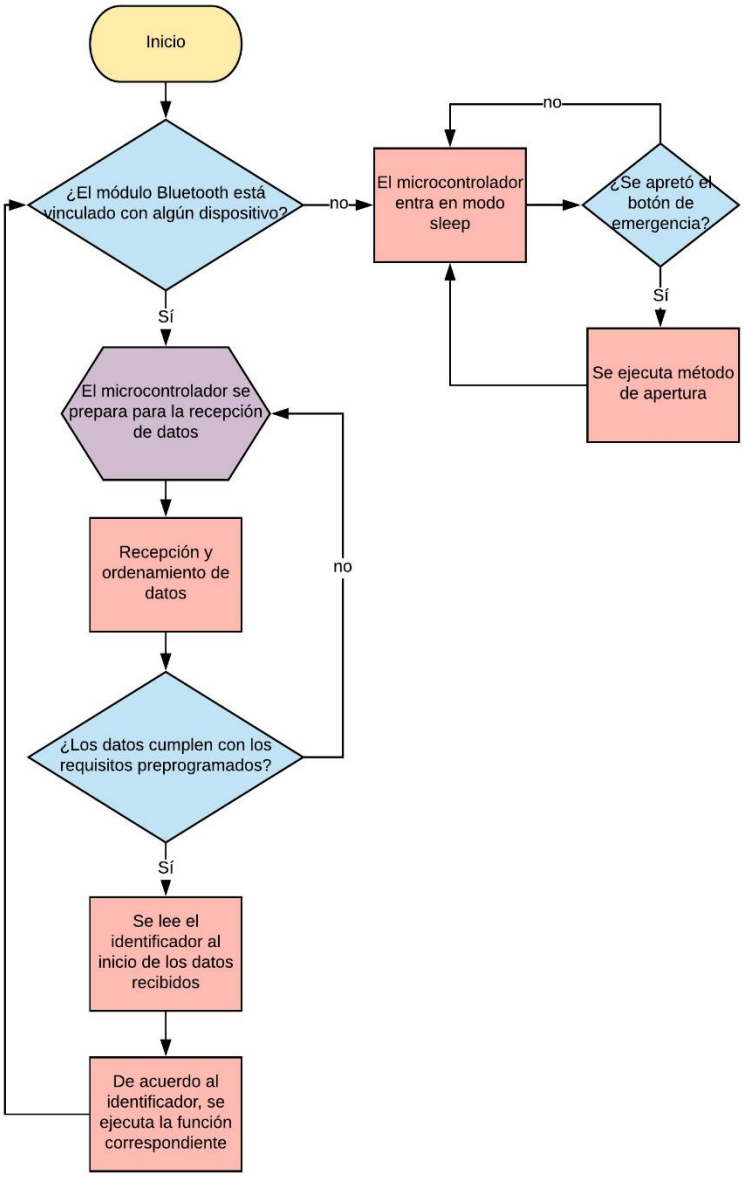


Figura 3.5 Diagrama de flujo resumido del sistema.
(Fuente: Elaboración propia)

A continuación, se muestra de manera más detallada cada una de las partes que componen la programación del microcontrolador.

Inicialización y Configuración de Registros

Como se observa en la Figura 3.6, primero se deben configurar ciertos registros del microcontrolador para que se puedan utilizar algunos recursos como el USART, las interrupciones, puertos digitales, etc.

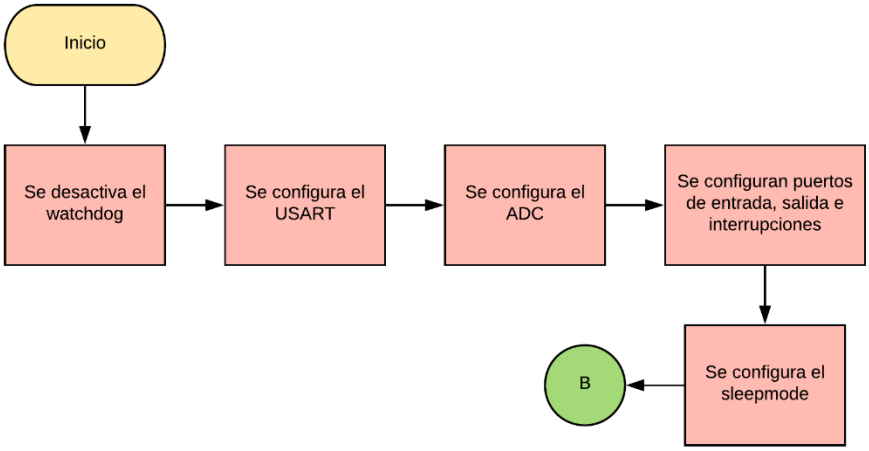


Figura 3.6 Configuración inicial de los registros del ATMEGA329P-PU.
(Fuente: Elaboración propia)

Inicialización del USART

Para lograr que el módulo Bluetooth y el microcontrolador se comuniquen a través del protocolo de comunicación serial se configuró el USART del microcontrolador de tal manera que estuviera siempre a la espera de datos.

Para garantizar que la comunicación se llevara a cabo de manera correcta se ajustó el baud rate de ambos dispositivos a 9600 bps y se calculó el valor del registro USART Baud Rate Register (UBRR) para el modo asíncrono normal.

$$UBRRn = \frac{f_{osc}}{16BAUD} - 1 \qquad (1)$$

Donde:

$$f_{osc} = 8\ [MHz]$$

$$BAUD = 9600\ [bps]$$

$$UBRRn = 51.08\overline{33} \cong 51$$

Inicialización del Convertidor Analógico Digital

La cerradura será operada a través de un servomotor modificado donde se aprovechará el potenciómetro integrado en el eje, mediante un divisor de voltaje, para saber en qué posición se encuentra el motor y así girar en un cierto sentido para abrir o cerrar la cerradura.

También se utilizará el convertidor analógico digital, más conocido por sus siglas en inglés como ADC, para monitorear el nivel de las baterías de respaldo.

Al encender el sistema o vincularse el módulo Bluetooth con algún dispositivo, una de las primeras acciones a realizar será el comprobar que las baterías de respaldo se encuentren en un nivel adecuado.

Monitoreo del nivel de voltaje de las baterías

La medición del voltaje de las baterías de respaldo es un punto crítico ya que, si no se le puede notificar al usuario de manera oportuna que es necesario cambiarlas o recargarlas existe una gran posibilidad que, si no hay electricidad, el usuario no pueda abrir la cerradura hasta que se reestablezca la electricidad.

Una práctica simple consiste en utilizar el ADC para medir el voltaje de la fuente de alimentación basado en el circuito mostrado en la Figura 3.7.

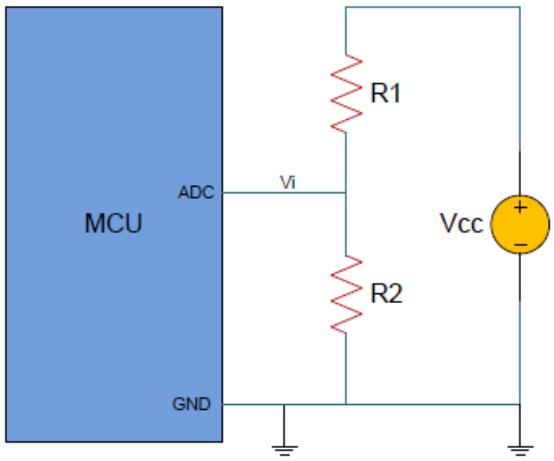


Figura 3.7 Divisor de voltaje para medir el valor de la fuente de alimentación.
(Fuente: http://ww1.microchip.com/downloads/en/AppNotes/00002447A.pdf. Último acceso: 17 de abril de 2018)

V_{cc} se podría determinar de la siguiente manera:

$$V_{cc} = \frac{V_i(R1 + R2)}{R1} \qquad (2)$$

Sin embargo, este modelo es más conveniente cuando el voltaje de la fuente de alimentación es mayor al voltaje de operación. En nuestro caso, el voltaje de alimentación es el mismo que el de operación, por lo tanto, no es necesario utilizar un divisor de voltaje, utilizándose el arreglo de la Figura 3.8.

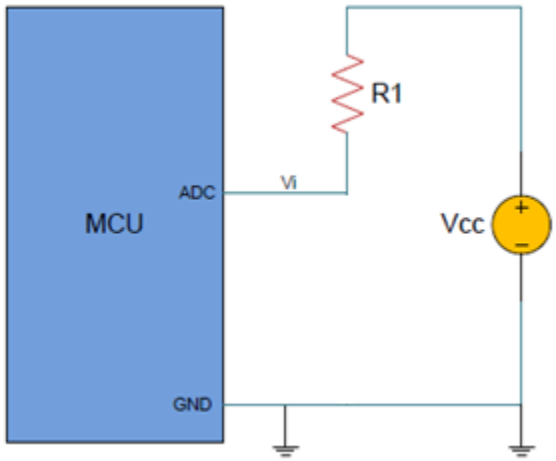


Figura 3.8 Modelo para medir el nivel de voltaje de las baterías de respaldo.
(Fuente: http://ww1.microchip.com/downloads/en/AppNotes/00002447A.pdf. Último acceso: 18 de junio de 2018)

La medición del voltaje puede ser calculada a través de la conversión obtenida del ADC, tomando en cuenta que en el ATMEGA328P-PU la resolución es de 10 bits.

$$\frac{Resolución\ del\ ADC}{Voltaje\ de\ referencia\ del\ ADC} = \frac{Resultado\ del\ ADC}{Voltaje\ de\ entrada\ al\ ADC} \quad (3)$$

Si el microcontrolador es alimentado por 5 V, donde generalmente es igual al voltaje de referencia del ADC, y el voltaje de entrada a convertir es de, por ejemplo, 3.8 V:

$$\frac{1023}{5\ V} = \frac{ADC_{val}}{3.8\ V}$$

$$ADC_{val} = \frac{(1023)(3.8\ V)}{5\ V} = 777$$

Como el único voltaje que puede variar es el voltaje de entrada al ADC, si disminuye el voltaje de entrada disminuirá el resultado de la conversión y viceversa.

Si el voltaje de entrada es igual a 5 V, tenemos:

$$ADC_{val} = \frac{(1023)(5\ V)}{5\ V} = 1023$$

O si es igual a 0V:

$$ADC_{val} = \frac{(1023)(0\ V)}{5\ V} = 0$$

Por lo tanto, si las baterías están totalmente cargadas se obtendrá un valor de 1023 del ADC o, en el caso contrario, se obtendrá un valor de 0.

En nuestro caso, solo necesitaremos notificar al usuario si las baterías cuentan con un nivel de voltaje óptimo para el funcionamiento del sistema o no, por consiguiente, se estableció un rango de voltajes dentro del cual el sistema se desempeña correctamente. Si el voltaje se encuentra por debajo del rango establecido, se deberá avisar al usuario que es necesario recargar o cambiar las baterías.

Para activar el ADC primero seleccionamos el voltaje de referencia, el cual será el mismo que el voltaje de alimentación (ver Tabla3.4), y lo configuramos en el registro ADMUX (ver Tabla3.3).

7	6	5	4	3	2	1	0
REFS1	REFS0	ADLAR		MUX3	MUX2	MUX1	MUX0

Tabla 3.3 Registro ADMUX del ADC.
(Fuente: http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-42735-8-bit-AVR-Microcontroller-ATmega328-328P_Datasheet.pdf. Último acceso: 25 de junio de 2018)

REFS [1:0]	Selección del voltaje de referencia
00	AREF, V_{ref} interno apagado
01	AV_{cc} con capacitor externo en el pin AREF
10	Reservado
11	Voltaje de referencia interno de 1.1 V con capacitor externo en el pin AREF

Tabla 3.4 Selección del voltaje de referencia del ADC.
(Fuente: http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-42735-8-bit-AVR-Microcontroller-ATmega328-328P_Datasheet.pdf. Último acceso: 25 de junio de 2018)

A continuación, seleccionamos el puerto de entrada para el ADC, correspondiente al monitoreo de las baterías (Tabla 3.5).

MUX [3:0]	Entrada de terminación única
0000	ADC0
0001	ADC1
0010	ADC2
0011	ADC3
0100	ADC4
0101	ADC5
0110	ADC6
0111	ADC7
1000	Sensor de temperatura
1001	Reservado
1010	Reservado
1011	Reservado
1100	Reservado
1101	Reservado
1110	1.1 V (V_{BG})
1111	GND

Tabla 3.5 Selección del canal de entrada.
(Fuente: http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-42735-8-bit-AVR-Microcontroller-ATmega328-328P_Datasheet.pdf. Último acceso: 25 de junio de 2018)

Para que el ADC funcione correctamente, es necesario ajustar la frecuencia del reloj del microcontrolador a una frecuencia de entre 50 kHz y 200 kHz, para ello se utiliza el preescalador que contiene el módulo del convertidor.

Si la frecuencia del reloj del microcontrolador es de 8 MHz, las posibles frecuencias que se podrían obtener utilizando el preescalador se presentan en la Tabla 3.6.

ADPS [2:0]	Factor de división	ADC_{CLK} [kHz]
000	2	4000
001	2	4000
010	4	2000
011	8	1000
100	16	500
101	32	250
110	64	125
111	128	62.5

Tabla 3.6 Frecuencias posibles provenientes del preescalador del ADC, dentro del registro ADCSRA.
(Fuente: http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-42735-8-bit-AVR-Microcontroller-ATmega328-328P_Datasheet.pdf. Último acceso: 25 de junio de 2018)

Como se puede observar en la Tabla 3.6, la opción que mejor se ajusta para la frecuencia del ADC es la de 125 kHz, correspondiente a un preescalamiento de 64.

Para obtener una mayor precisión durante el cálculo del voltaje de la batería, se tomarán 100 muestras cada vez que se ejecute el método de monitoreo y se calculará el promedio de los valores leídos, para ello, es necesario que el ADC se dispare automáticamente tras realizar una conversión. Cambiando el bit ADATE del registro ADCSRA a 1 activamos el disparo automático (Tabla 3.7).

7	6	5	4	3	2	1	0
ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0

Tabla 3.7 Registro A de control y estado del ADC.
(Fuente: http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-42735-8-bit-AVR-Microcontroller-ATmega328-328P_Datasheet.pdf. Último acceso: 25 de junio de 2018)

Monitoreo del Posicionamiento del Motor

Para monitorear en qué posición se encuentra el motor, ya sea para abrir o cerrar la cerradura, se utilizará, al igual que en el monitoreo de las baterías, el ADC del microcontrolador.

La configuración de los registros será prácticamente la misma que la mencionada en el tema anterior, pero con la ligera modificación en el canal de entrada (ver Figura 3.9 y Tabla 3.8).

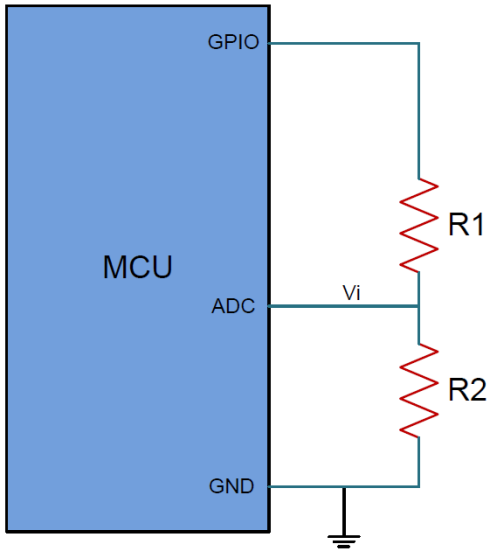


Figura 3.9 Modelo del potenciómetro en el motor.
(Fuente: http://ww1.microchip.com/downloads/en/AppNotes/00002447A.pdf. Último acceso: 25 de junio de 2018)

Además, la alimentación del potenciómetro se llevará a cabo mediante un pin del microcontrolador, con el objetivo de poder controlar cuando esté activo el divisor de voltaje.

MUX [3:0]	Entrada de terminación única
0000	ADC0
0001	ADC1
0010	ADC2
0011	ADC3
0100	ADC4
0101	ADC5
0110	ADC6
0111	ADC7
1000	Sensor de temperatura
1001	Reservado
1010	Reservado
1011	Reservado
1100	Reservado
1101	Reservado
1110	1.1 V (V_{BG})
1111	GND

Tabla 3.8 Selección del canal de entrada.
(Fuente: http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-42735-8-bit-AVR-Microcontroller-ATmega328-328P_Datasheet.pdf. Último acceso: 25 de junio de 2018)

Configuración de los Puertos de Entrada/Salida

Para el sistema se utilizaron los puertos de entrada y salida que se presentan en la Tabla 3.9.

Tipo	Puerto	Características	Función
Entrada	PD0	RX	Recibir datos del Bluetooth
	PD2	Interrupción activa en modo flanco de subida con resistencia de pull-up activa	Determinar si el Bluetooth está vinculado con otro dispositivo
	PD6	Resistencia de pull-up activa	Determinar estado de la puerta
	PC0	Entrada analógica	Leer posición del potenciómetro
Salida	PD1	TX	Transmisión de datos hacia el Bluetooth
	PD7	-----	LED indicador
	PB0	-----	LED indicador
	PB3	-----	Habilitar driver del motor
	PB4	-----	Sentido del giro del motor
	PB5	-----	Sentido del giro del motor

Tabla 3.9 Puertos utilizados y sus funciones.
(Fuente: http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-42735-8-bit-AVR-Microcontroller-ATmega328-328P_Datasheet.pdf. Último acceso: 18 de diciembre de 2017)

Modos Sleep o de Bajo Consumo

EL módulo Bluetooth HM-10 cuenta con un pin de estado que indica, mediante una salida de 3.3 V, cuando está conectado con algún dispositivo y, con 0 V, cuando no lo está.

Con ayuda de este pin se programó al microcontrolador para que entre en uno de los modos sleep cuando no se encuentre conectado con algún dispositivo, esto con el objetivo de reducir el consumo de energía.

El ATMEGA328P-PU cuenta con los modos sleep de la Tabla 3.10.

Modo Sleep	Descripción
Idle	Detiene el CPU, pero permite que los sistemas SPI, USART, comparador analógico, I2C, timers/counters, watchdog e interrupciones continúen operando. Detiene el clk_{CPU} y el clk_{FLASH}.
ADC Noise Reduction	Detiene el CPU, pero permite que los sistemas ADC, I2C, timer/counter2, watchdog e interrupciones externas continúen operando. Detiene el $clk_{CPU,}$ clk_{FLASH} y el $clk_{I/O}$.
Power-down	En este modo el cristal externo es detenido permitiendo al sistema de interrupciones externas, I2C y el watchdog continuar operando. Se detienen todos los relojes generados permitiendo solo la operación de los módulos asíncronos.
Power-save	Esta opción es idéntica al modo power-down con la excepción que, si el timer/counter2 está habilitado, continuará operando durante el modo sleep. Si no está habilitado, este modo es recomendado en lugar del modo power-down.
Standby	Esta opción es idéntica al modo power-down con la excepción que el cristal externo permanece operando.
Extended Standby	Esta opción es idéntica al modo power-save con la excepción que el cristal externo permanece operando.

Tabla 3.10 Modos de bajo consumo del ATMEGA328P-PU.
(Fuente: http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-42735-8-bit-AVR-Microcontroller-ATmega328-328P_Datasheet.pdf. Último acceso: 18 de diciembre de 2017)

Dado que con el microcontrolador no se utilizó un cristal externo, esto también con el objetivo de reducir el consumo de energía, y que en su lugar se utilizó uno de los relojes internos del microcontrolador, el modo sleep que más conviene utilizar es el power-save.

Utilizando una interrupción externa y en conjunto con el pin State del Bluetooth (Figura 3.10), se programó al microcontrolador para que, a través del pin asignado a la interrupción INT0, entrara al modo sleep cuando detectara un cero lógico y para que saliera del modo sleep mediante un flanco de subida.

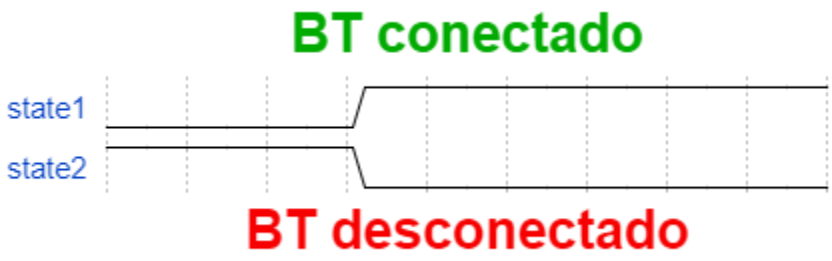


Figura 3.10 Diagrama de comportamiento del pin State.
(Fuente: Elaboración propia)

Cabe mencionar que el módulo Bluetooth HM-10 también cuenta con un modo de bajo consumo que se activa enviando el comando AT+SLEEP. Sin embargo, este modo no será utilizado ya que resulta poco práctico desactivarlo.

Distribución de la memoria EEPROM

El microcontrolador ATMEGA328P-PU cuenta con 1 kb de memoria EEPROM que se utilizó para almacenar los datos de los usuarios y del administrador. Las localidades de memoria se distribuyeron de tal manera que fuera más eficiente la lectura/escritura de los datos a través de funciones itinerantes.

0	1	101	201	301	401	501	601	701
	2	102	202	302	402	502	602	702
	3	103	203	303	403	503	603	703
	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.
	100	200	300	400	500	600	700	709

Tabla 3.11 Distribución de las localidades de la memoria EEPROM.
(Fuente: Elaboración propia)

De acuerdo con la Tabla 3.11, la localidad 0 corresponde a un contador que llevará la cuenta de los usuarios dados de alta y, además, será quien les asigne el ID a los usuarios.

El bloque de memoria correspondiente de la localidad 1 a la 600 se utilizó para guardar los códigos de cada usuario de acuerdo con el patrón de la Figura 3.11.

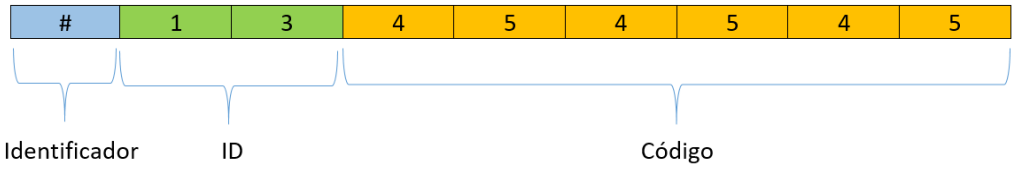


Figura 3.11 Estructura de un paquete de datos recibido por el microcontrolador para solicitar una entrada/salida. (Fuente: Elaboración propia)

Localidad de memoria	14 (ID+1)	114 (ID+101)	214 (ID+201)	314 (ID+301)	414 (ID+401)	514 (ID+501)
Valor almacenado	4	5	4	5	4	5

Tabla 3.12 Almacenamiento de los datos en la memoria EEPROM. (Fuente: Elaboración propia).

El sistema se diseñó para una capacidad máxima de 100 usuarios debido al tamaño de la memoria EEPROM (1kb) y por la estrategia empleada para almacenar los datos en la misma

El bloque de memoria correspondiente de la localidad 601 a 700 se utilizó para almacenar los intentos fallidos de los usuarios y para examinar si un usuario se encuentra bloqueado (ver Tabla 3.11).

La localidad 701 se empleó para almacenar el número de intentos que tienen los usuarios en general para realizar ingresos de emergencia, cada vez que un usuario utiliza el ingreso de emergencia éste contador disminuye en uno (ver Tabla 3.11).

Por último, el bloque correspondiente de la localidad 702 a 709 corresponde a la contraseña que utiliza el administrador para registrar nuevos usuarios (ver Tabla 3.11).

Cabe destacar que se empleó una función específica para alargar la vida útil de escritura de la memoria EEPROM que consiste en comparar el valor almacenado con el valor que se desea escribir, evitando así redundancias en la escritura.

Programa Principal y Métodos

El programa principal que se presenta en la Figura 3.12, se compone esencialmente por sentencias if y else, las cuales cuentan con distintas condiciones para poder ejecutar acciones específicas. Los datos se reciben en un arreglo de tipo char llamado “aux”.

La recepción de datos en el sistema principal se optó que fuera de valores tipo “char” o carácter, esto con el objetivo de poder distinguir fácilmente las operaciones a realizar.

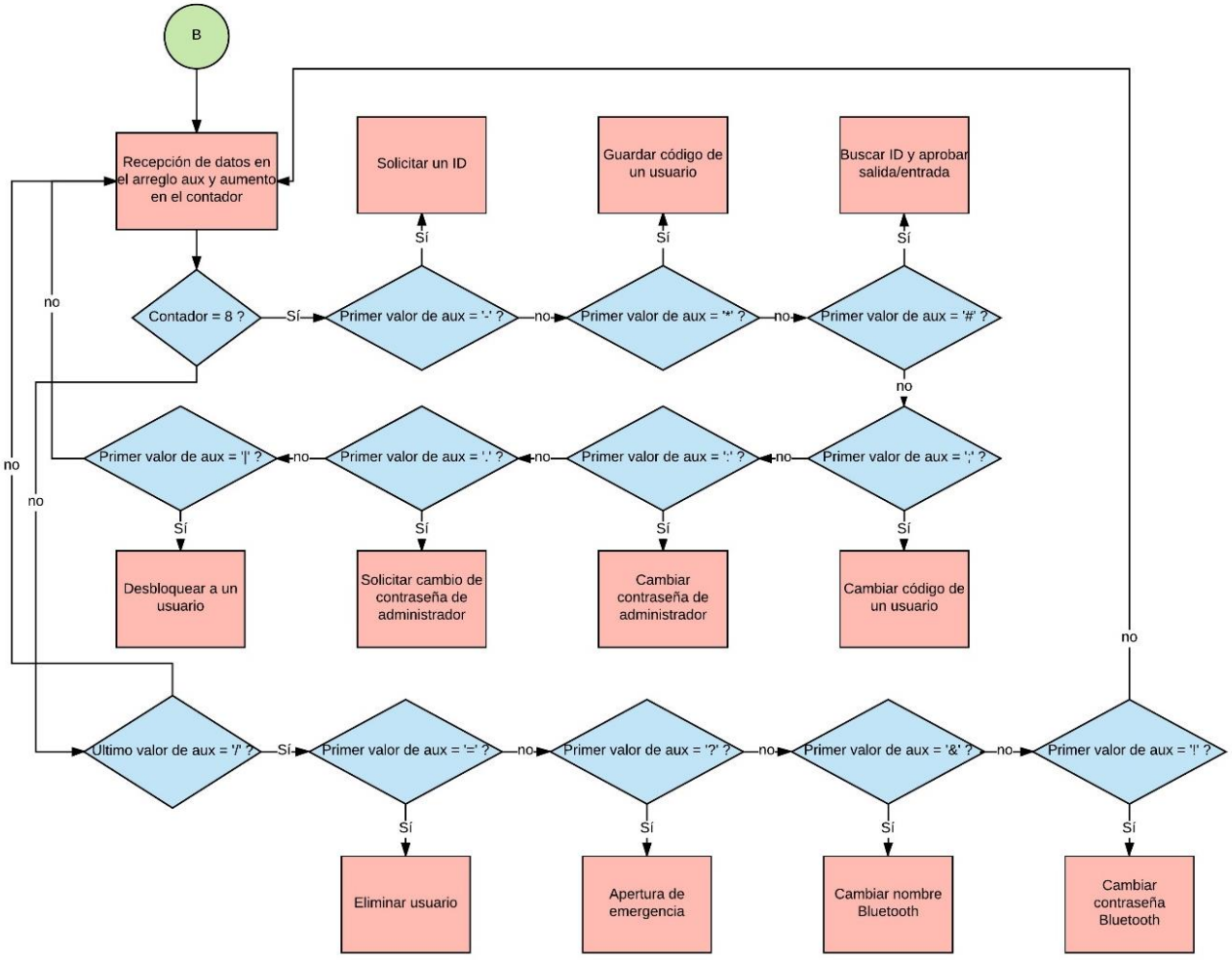


Figura 3.12 Diagrama de flujo del programa principal.
(Fuente: Elaboración propia)

Se desarrollaron distintos arreglos con características específicas para identificar las acciones que el usuario podrá realizar y si los datos ingresados por el mismo cumplen con los requisitos programados. En la Tabla 3.13 se muestran los arreglos, sus características y las funciones que desempeñan.

Identificador	Características	Función
-	El arreglo se compone por el identificador "-" al inicio, seguido por 8 dígitos correspondientes a la contraseña de registro. -01234567	Utilizado para solicitar un ID y registrarlo en la memoria EEPROM del microcontrolador.
*	El arreglo se compone por el identificador "*" al inicio, seguido por 8 dígitos, los dos primeros	Empleado para guardar el código de un usuario en la memoria EEPROM del microcontrolador.

	correspondientes al ID del usuario y los 6 restantes al código elegido. *01334455	
#	El arreglo se compone exactamente como el identificador “*”. #02898956	Utilizado cuando el usuario ha solicitado una salida o entrada y ha ingresado su código en la aplicación. Se compara su código con el guardado en la EEPROM del microcontrolador y se realiza la operación correspondiente.
;	El arreglo se compone exactamente como el identificador “*”. ;05334455	Empleado para cambiar el código de un usuario. Primero se ingresa el código actual y se compara con el almacenado en la memoria EEPROM permitiendo, o no, el cambio de este.
=	El arreglo se compone por el identificador “=” al inicio, seguido por dos dígitos correspondientes a un ID. =15	Función disponible sólo para el administrador. Empleada para eliminar a un usuario del sistema.
?	El arreglo se compone de la misma manera que el identificador “=”. ?23	Función empleada para una apertura de emergencia en caso de que el usuario haya olvidado su código.
&	El arreglo se compone por el identificador “&” al inicio, seguido por un número indefinido de caracteres. &CerraduraCasa	Función disponible sólo para el administrador. Utilizada para cambiar el nombre al módulo Bluetooth.
!	El arreglo se compone por el identificador “!” al inicio, seguido por 6 dígitos. !123456	Función disponible sólo para el administrador. Empleada para cambiar la contraseña de vinculación del módulo Bluetooth.
.	El arreglo se compone por el identificador “.” al inicio, seguido por 8 dígitos correspondientes a la contraseña para registro de nuevos usuarios. .11111111	Función disponible sólo para el administrador. Utilizada para solicitar el cambio de la contraseña con la que se les asigna un ID a los usuarios.
:	El arreglo se compone por el identificador “:” al inicio, seguido de 8 dígitos. :00000000	Función disponible sólo para el administrador. Empleada para guardar la nueva contraseña que asigna los ID en la memoria EEPROM.
|	El arreglo se compone por el identificador “|” al inicio, seguido por 10 dígitos, los primeros 8 corresponden a la contraseña de asignación de ID y los 2 restantes a un ID. |0000000018	Función utilizada para desbloquear al usuario que ha superado el número de intentos disponibles para ingresar.

Tabla 3.13 Estructura y funciones de los paquetes programados para la ejecución de los métodos. (Fuente: Elaboración propia)

Solicitud de un ID o Registro de un Nuevo Usuario

En el diagrama de flujo de la Figura 3.13, se muestra el método desarrollado para el registro de un nuevo usuario.

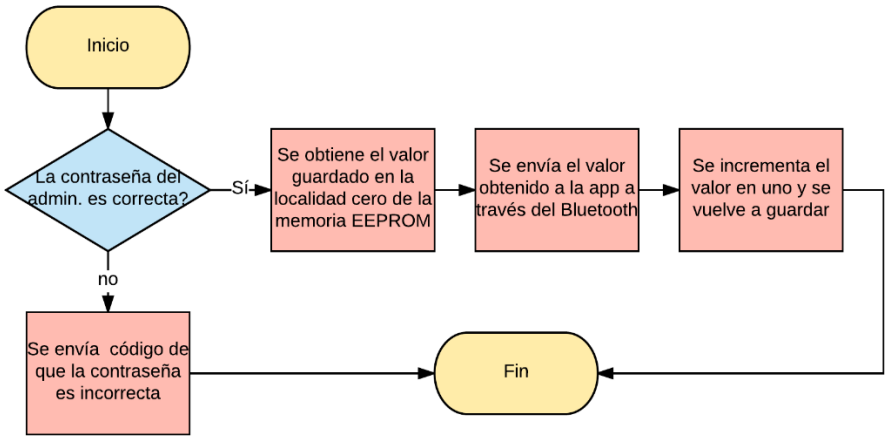


Figura 3.13 Diagrama de flujo para registrar un usuario.
(Fuente: Elaboración propia)

Almacenamiento del Código de un Usuario

Este método, mostrado en el diagrama de la Figura 3.14, se utiliza para dos casos:

- Almacenar el código de un nuevo usuario
- Cambiar el código de un usuario ya registrado

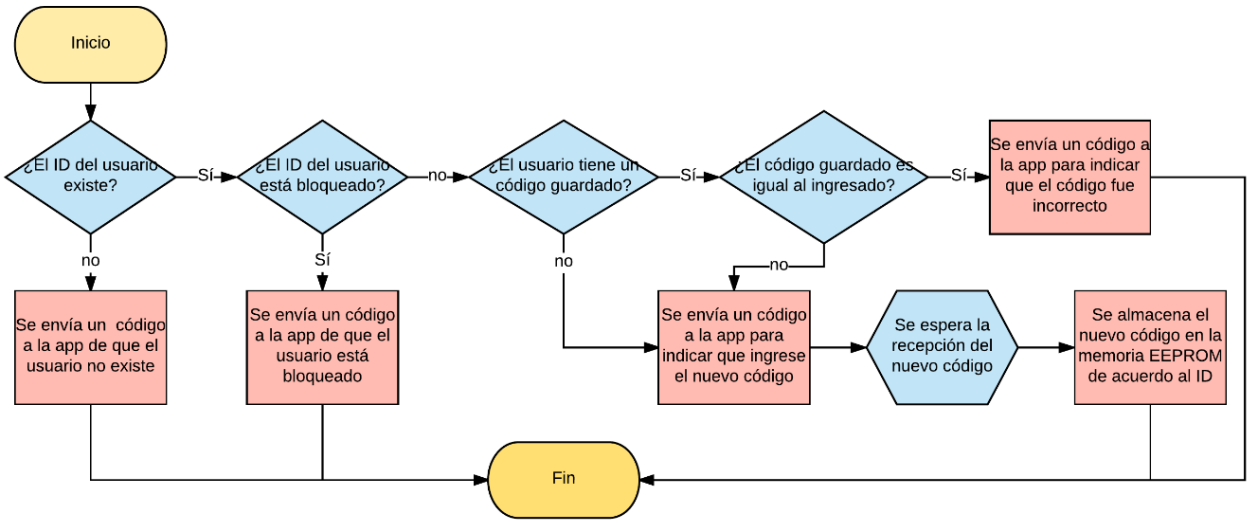


Figura 3.14 Diagrama de flujo para guardar o cambiar el código de un usuario.
(Fuente: Elaboración propia)

Entrada o Salida de un Usuario

Como se observa en el diagrama de la Figura 3.15, para que un usuario pueda realizar una entrada o salida se deberá seguir el siguiente procedimiento:

- Elegir en la aplicación cuál de las dos acciones desea realizar
- Ingresar su código

El microcontrolador comparará el código recibido con el almacenado en la memoria EEPROM y ejecutará las acciones correspondientes.

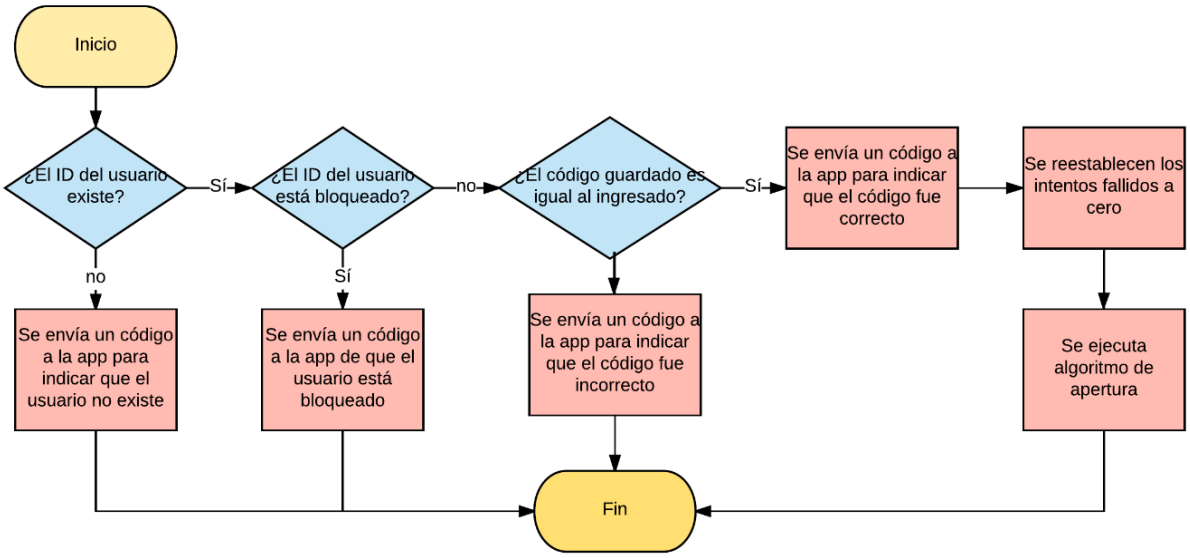


Figura 3.15 Diagrama de flujo para solicitar una entrada/salida.
(Fuente: Elaboración propia)

Método de Apertura

Este proceso está constituido por una serie de pasos, mostrado en la Figura 3.16, para asegurar que la cerradura no se abra o se cierre en momentos no deseados.

- Estado inicial: puerta cerrada.
- Si el código ingresado por el usuario es correcto se habilita el driver del motor y comienza la apertura de la cerradura.
- Se habilita el ADC y se monitorea el giro del motor a través del potenciómetro hasta completar la apertura.
- Se abre la cerradura y permanece así hasta que la puerta sea abierta.
- Si la puerta no se abre en los próximos 30 segundos, la cerradura se cierra automáticamente.
- La puerta es abierta.
- Se espera a que la puerta vuelva a cerrarse.

- La puerta es cerrada y se vuelve a habilitar el driver del motor realizar el cierre de la cerradura.
- Se vuelve a habilitar el ADC y se monitorea el giro hasta completar el cierre.

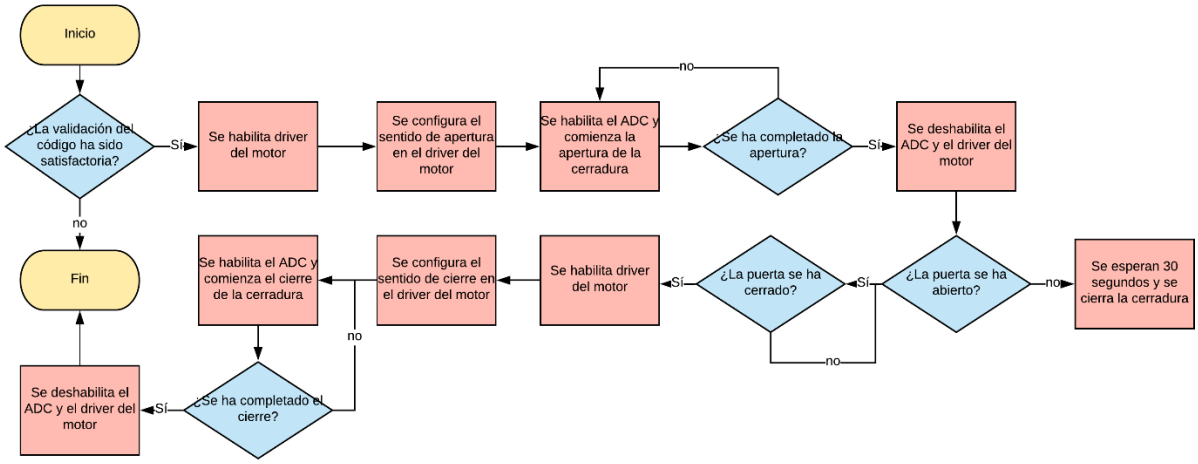


Figura 3.16 diagrama de flujo para la apertura de la cerradura.
(Fuente: Elaboración propia)

Actualizar Contraseña del Administrador

A pesar de que la contraseña del administrador se encuentra ya preestablecida en la memoria EEPROM del microcontrolador, el administrador es capaz de cambiarla en cualquier momento de acuerdo con el algoritmo de la Figura 3.17.

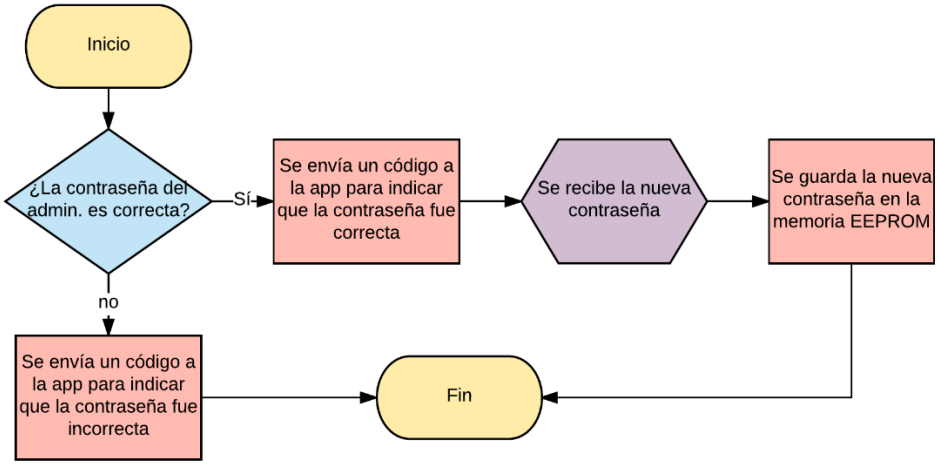


Figura 3.17 Diagrama de flujo para cambiar la contraseña del administrador.
(Fuente: Elaboración propia)

Desbloqueo de un Usuario

El bloqueo de un usuario puede ser causado por dos situaciones:

- Exceder el número de intentos disponibles para ingresar el código
- Ser bloqueado por el administrador

Para el desbloqueo de un usuario es necesario que el administrador ingrese su contraseña junto con el ID a desbloquear, como se observa en la Figura 3.18.

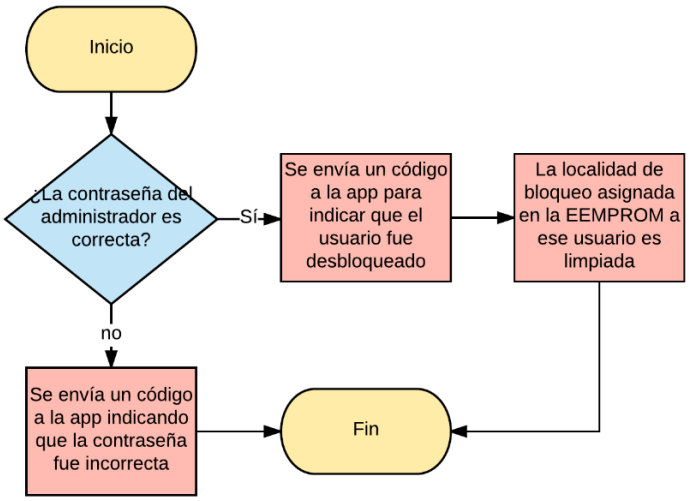


Figura 3.18 Diagrama de flujo para el desbloqueo de un usuario.
(Fuente: Elaboración propia)

Eliminación de un Usuario

De manera similar al bloqueo de un usuario, el administrador debe ingresar su contraseña y el ID del usuario a eliminar (Figura 3.19).

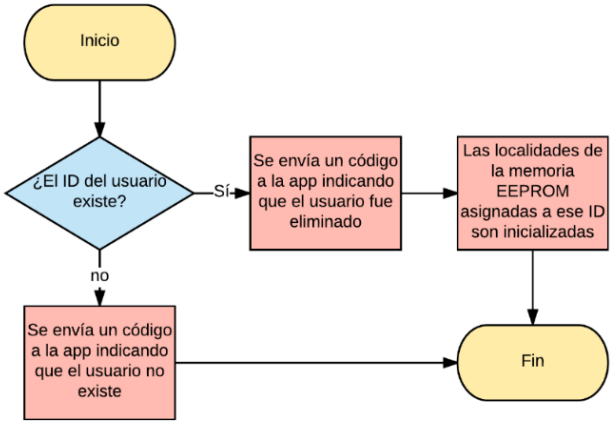


Figura 3.19 Diagrama de flujo para eliminación de un usuario.
(Fuente: Elaboración propia)

Apertura de Emergencia

Esta función, mostrada en la Figura 3.20, se desarrolló como solución a las distintas situaciones en las que los usuarios podrían verse afectados como:

- Extravío o robo del dispositivo que albergaba la aplicación.
- Olvidar el código de entrada/salida

Se programaron sólo 5 accesos de emergencia, una vez excedido este número los usuarios no podrán hacer uso de esta función. Cabe mencionar que esta no es una función que el administrador pueda manipular.

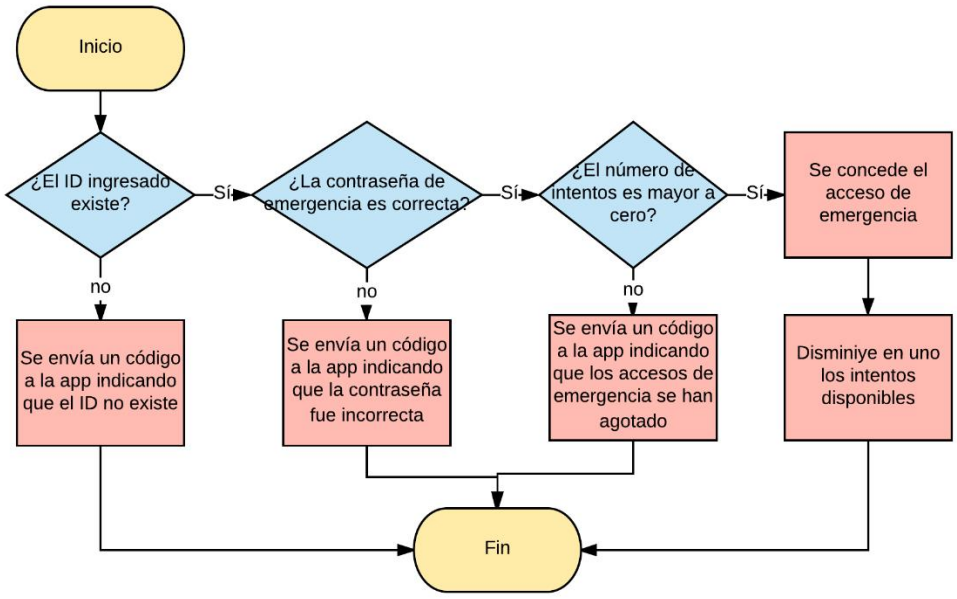


Figura 3.20 Diagrama de flujo para realizar una apertura de emergencia.
(Fuente: Elaboración propia)

Apertura de Emergencia desde el Interior

Esta función se desarrolló con el fin de realizar una apertura rápida en caso de que el usuario necesite salir repentinamente.

Este proceso se activa a través de un botón que, al ser oprimido, genera una interrupción en el microcontrolador e inicia el proceso de apertura inmediatamente.

3.2.2 Programación de la Aplicación Android

Para la programación de la aplicación se desarrollaron distintas funciones de acuerdo con las planteadas en el capítulo anterior:

- Establecer código de entrada/salida
- Cambiar código de entrada/salida
- Limitar número de intentos de acceso
- Identificación de usuarios
- Creación de un administrador
- Registro de cada usuario a través de contraseña
- Bloqueo de algún usuario
- Eliminación de algún usuario
- Cambio de contraseña de registro
- Cambio del nombre del módulo Bluetooth
- Cambio del pin de vinculación del módulo Bluetooth
- Registro de entrada/salida de los usuarios
- Utilización de clave de emergencia

La aplicación Android será el medio por el cual se manipule el sistema, por lo tanto, resulta coherente que ambos sistemas, el microcontrolador y la aplicación, posean casi las mismas funciones.

La aplicación se dividió en 4 secciones agrupando en cada una de ellas las distintas funciones antes mencionadas (ver Figura 3.21).

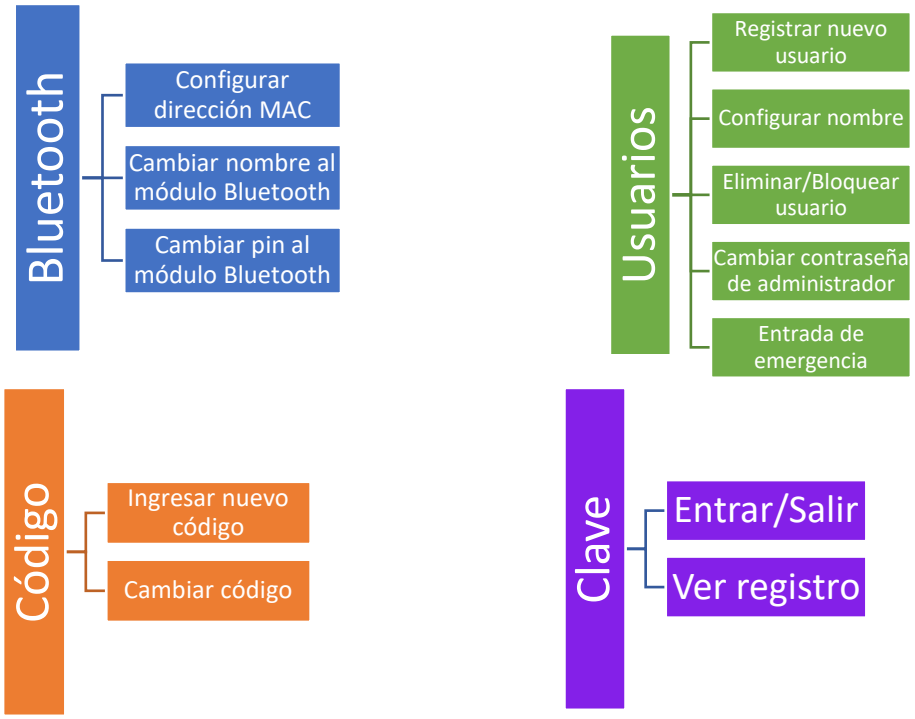


Figura 3.21 Secciones que conforman la aplicación Android.
(Fuente: Elaboración propia)

A continuación, se explicarán con más detalle el diseño de las funciones, además de otras características necesarias para el correcto funcionamiento de la aplicación.

Desarrollo de la Pantalla Bluetooth

Al iniciar la aplicación por primera vez, el usuario deberá configurar la dirección del Bluetooth de la cerradura, esto con el objetivo de lograr una conexión automática entre dispositivos cada vez que se inicie la aplicación (Figura 3.22).

Si el usuario resulta ser el administrador, podrá cambiar también el nombre del Bluetooth de la cerradura y el pin de vinculación.

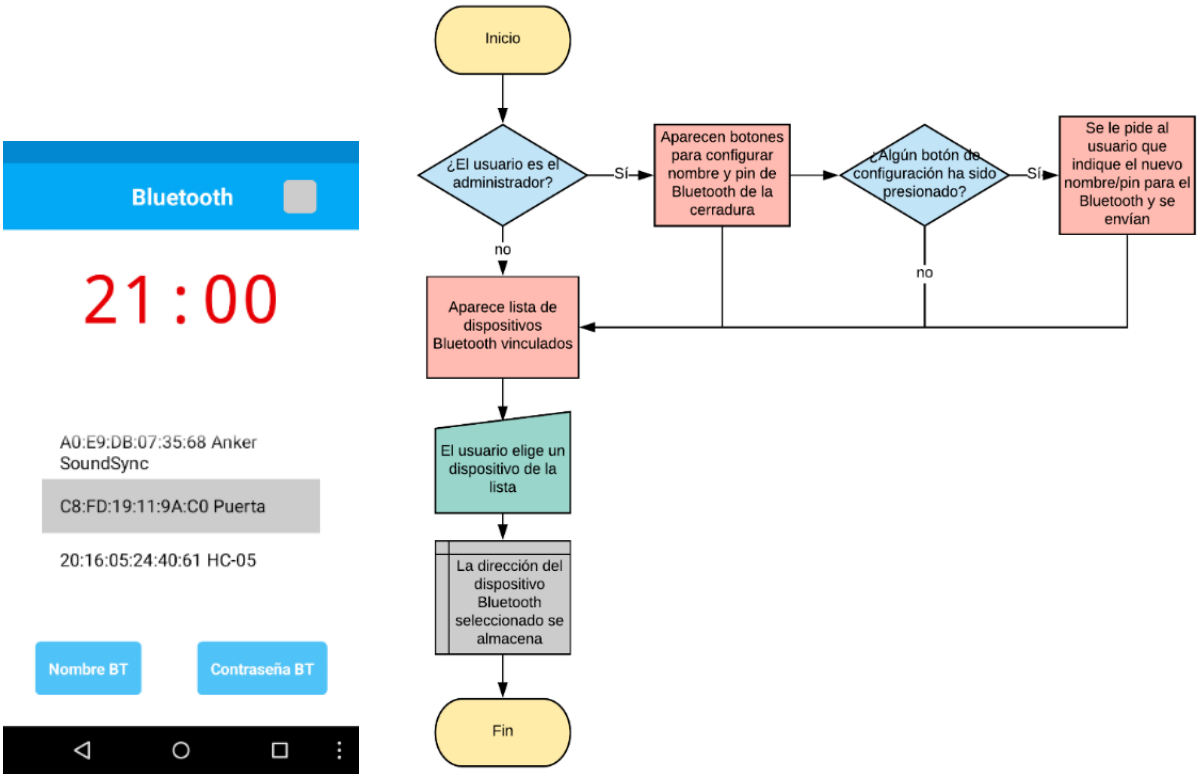


Figura 3.22 En la figura de la izquierda se muestra la configuración de la dirección Bluetooth y en la figura de la derecha el diagrama de flujo del funcionamiento de la pestaña Bluetooth.
(Fuente: Elaboración propia)

Una vez configurada la dirección Bluetooth, la cual se guarda en la memoria interna del smartphone, cada vez que se inicie la aplicación se llevará a cabo la conexión automática con el módulo Bluetooth del microcontrolador; se le indicará al usuario que la conexión ha sido establecida mediante el cambio del color de un recuadro de color gris a verde.

Desarrollo de la Pantalla Usuarios

En esta pantalla se configuran el ID y el nombre con el que se reconocerá al usuario, una vez configurados estos datos esta pantalla permanece en blanco a menos que el usuario sea el administrador (Figura 3.23).

Si el usuario es el administrador, y después de haber configurado su ID y su nombre, las configuraciones adicionales que podrá llevar a cabo son la eliminación de algún usuario y el cambio de la contraseña de administrador.

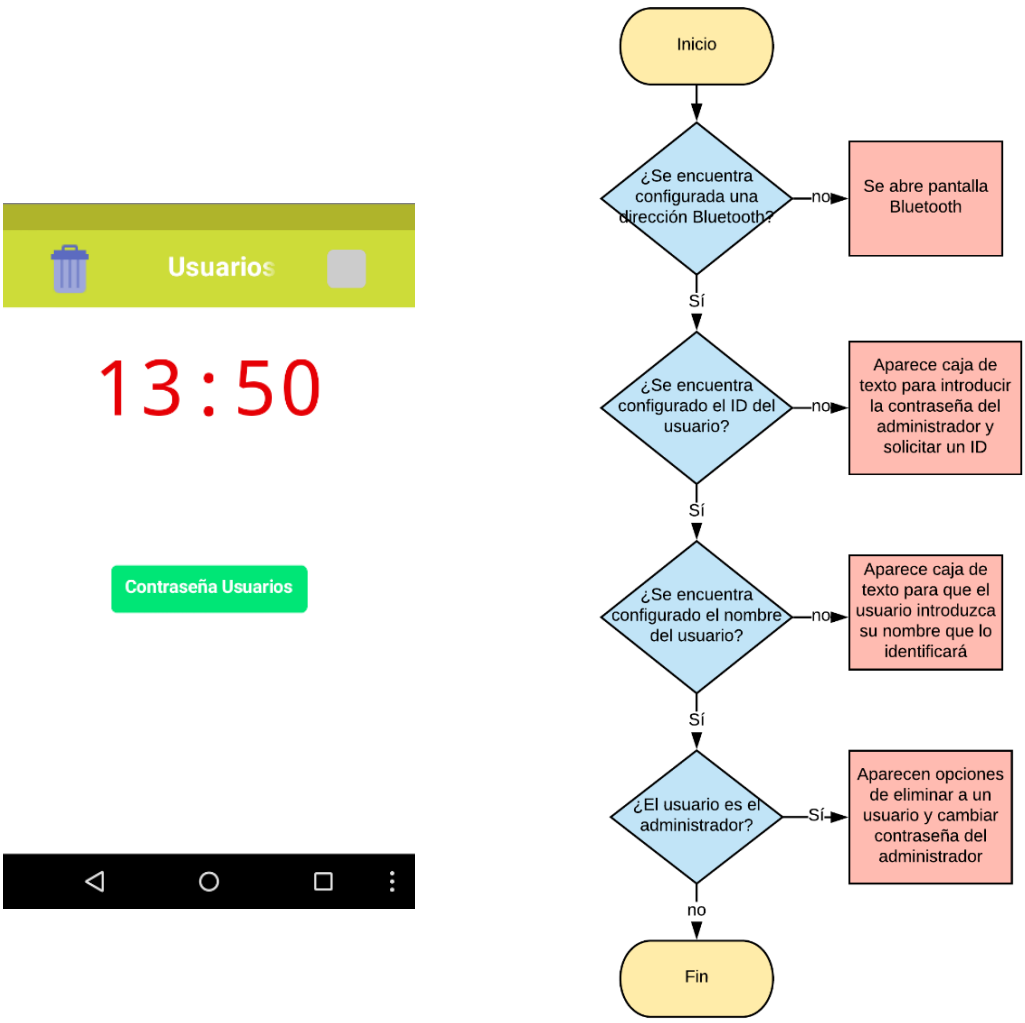


Figura 3.23 En la figura de la izquierda se muestra la pantalla de configuración de usuarios y en la figura de a derecha el diagrama de flujo de la pantalla de usuarios.
(Fuente: Elaboración propia)

El ID recibido por el microcontrolador y el nombre de usuario que elige el usuario son almacenados en la memoria interna del smartphone.

Desarrollo de la Pantalla Código

Después que la dirección Bluetooth, el ID y el nombre de usuario han sido configurados, el usuario debe configurar su código de acceso.

El código de acceso está compuesto por una combinación de seis colores a partir de cuatro colores disponibles: rojo, verde, azul y amarillo (ver Figura 3.24). El usuario puede crear combinaciones entre estos cuatro colores o simplemente elegir un color seis veces.

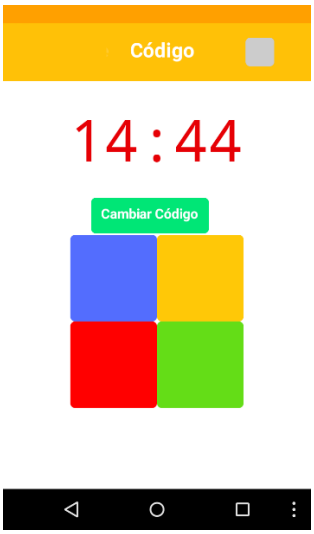


Figura 3.24 Pantalla de configuración del código de acceso.
(Fuente: Elaboración propia)

Al presionar la combinación de seis colores estos se envían al microcontrolador y son almacenados en el bloque de memoria EEPROM correspondiente al ID del usuario, no se almacenan en la memoria del teléfono como el ID o la dirección del Bluetooth, esto con el objetivo de no guardar información esencial en el dispositivo y reducir la posibilidad del robo de datos.

Al ser más práctico y eficiente el enviar un solo número que una palabra, por ejemplo, enviar el número “1” en lugar de enviar la palabra “rojo” o “amarillo”, a cada color se le asignó un número que lo identificaría de los otros colores, por ejemplo, Rojo = 1, Azul = 2, etc.

Sin embargo, que cada color tuviera siempre el mismo identificador no hacía un gran cambio en la seguridad del código, así que se desarrolló un método para que los identificadores asignados a los colores cambiaran cada vez que se establecía un nuevo código, es decir, si en un principio el color rojo correspondía al número uno y el usuario cambiara su código, aunque siga siendo la misma combinación de colores anterior, el color rojo tendría un nuevo identificador, como el número cuatro o seis, de la misma manera que el resto de los colores.

Cuando el usuario ha establecido su código, los identificadores de los colores sí se guardan en la memoria interna del smartphone.

En conjunto con el cambio aleatorio de los identificadores de los colores, se agregó otro método para reforzar la seguridad el cual consiste en cambiar de lugar y de forma aleatoria los colores cada vez que se inicia la aplicación (ver Figura 3.25).

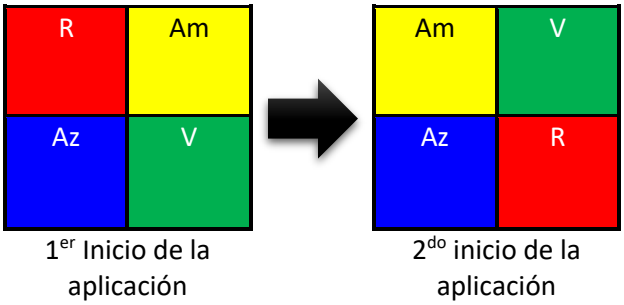


Figura 3.25 Cambio de la posición de los colores.
(Fuente: Elaboración propia)

Cabe resaltar que para que un usuario pueda cambiar su código, primero debe ingresar su código actual y, si es correcto, podrá configurar un nuevo código como se muestra en el diagrama de la Figura 3.26.

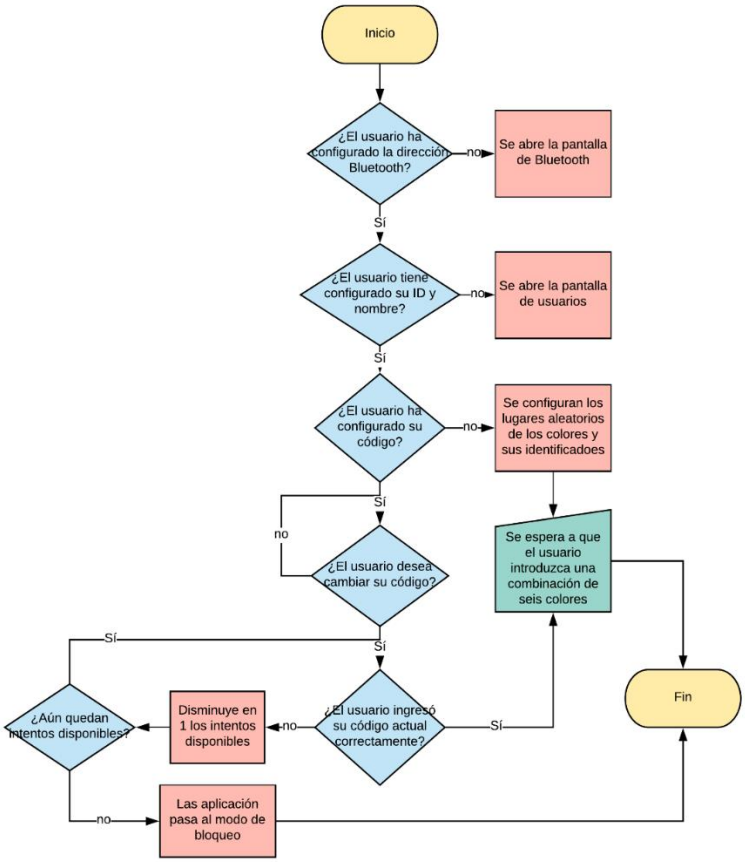


Figura 3.26 Diagrama de bloques de la pantalla código.
(Fuente: Elaboración propia)

Desarrollo de la Pantalla Clave

Ésta se convierte en la pantalla principal de la aplicación una vez configurada (Figura 3.27), a través de ella, se podrán realizar los accesos y revisar el registro de entrada o salida.

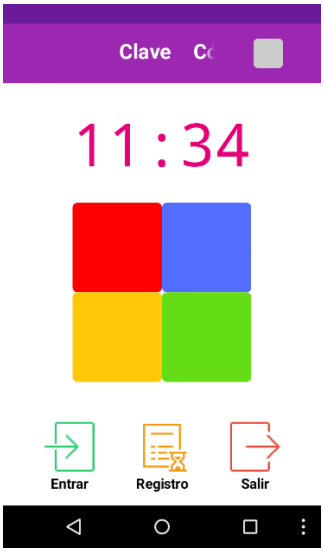


Figura 3.27 Pantalla principal de la aplicación.
(Fuente: Elaboración propia)

Cuando el usuario se ha enlazado con la cerradura, simplemente deberá elegir si realizará una entrada o una salida presionando el botón correspondiente, esto habilitará los botones de colores para que a continuación ingrese su código y se realice la apertura de la cerradura o se le notifique al usuario que su código fue incorrecto.

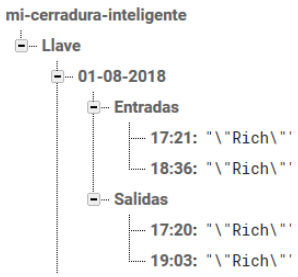


Figura 3.28 Almacenamiento y ramificación del registro en la base de datos.
(Fuente: Consola de la base de datos en tiempo real del proyecto mi-cerradura-inteligente en Firebase)

Cada vez que un usuario ejecuta una entrada o salida se realiza un registro de la acción recopilando datos como la hora, el día, el usuario responsable y la acción. Estos datos se

almacenan en una base de datos en línea suministrada por Firebase[6] de Google (Figura 3.28).

Es necesario tener una conexión a internet para que el registro se pueda actualizar o consultar, si no se cuenta con una conexión, los datos del registro son almacenados en la memoria del smartphone para después poder transferirlos a la base de datos cuando se tenga de nuevo conexión (ver Figura 3.29).

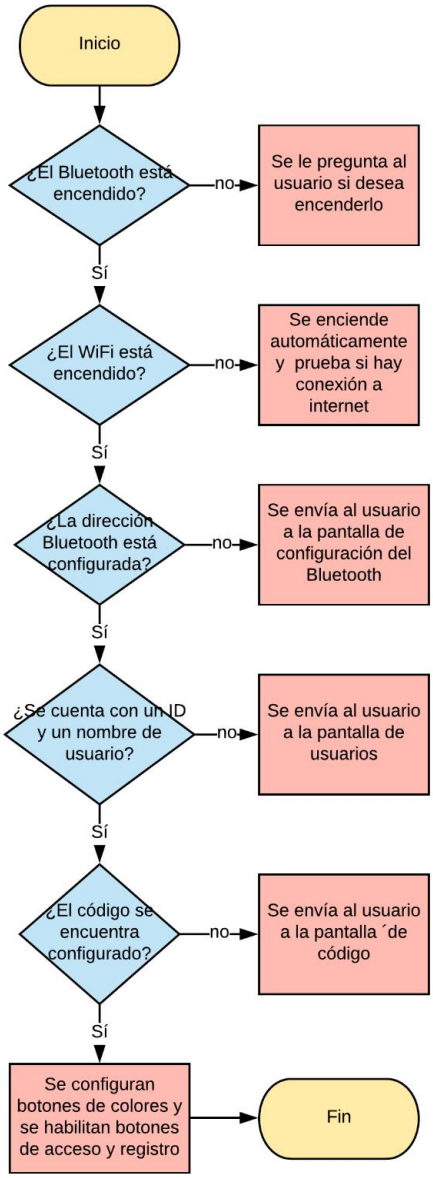


Figura 3.29 Diagrama de bloques de la pantalla clave.
(Fuente: Elaboración propia)

[6] Firebase es una plataforma para el desarrollo de aplicaciones web y aplicaciones móviles con distintos servicios, de los cuáles se utilizó la base de datos en tiempo real para poder almacenar el registro de la aplicación.

4 DISEÑO, CONSTRUCCIÓN Y ENSAMBLADO DEL PROTOTIPO

Para el diseño de la cerradura se buscaron distintos tipos de chapas que fueran de uso común y cuyo funcionamiento radicara en jalar y empujar una barra de seguridad.

De acuerdo con las especificaciones antes mencionadas se optó por la cerradura llave-mariposa de la marca PHILLIPS (Figura 4.1) ya que, además de que su costo no es muy elevado en comparación con otras marcas como SCHLAGE o KWIKSET, se le puede adaptar un motor para su operación.

Figura 4.1 Cerradura llave-mariposa marca PHILLIPS.
(Fuente: https://www.phillips.com.mx/es/site/phillipscommx/productos/linea-economica/b-260-p-/. Último acceso: 24 de enero de 2018)

El siguiente paso fue elegir un motor que tuviera el torque suficiente para poder abrir y cerrar la cerradura y, debido al funcionamiento de apertura y cierre de la cerradura, pueda girar al menos 90° en ambos sentidos. Inicialmente se había optado y utilizado el servomotor S05NF STD (Figura 4.2), que cuenta con un engranaje de metal y un torque de aproximadamente 3 kg-cm operando a 5V, siendo más que suficiente para operar la cerradura.

Figura 4.2 DGServo S05NF STD.
(Fuente: https://media.digikey.com/photos/Sparkfun%20Elec%20%20Photos/ROB-10333.jpg. Último acceso: 23 de mayo de 2018)

Sin embargo, en algunas ocasiones su funcionamiento incorrecto lo volvía impredecible. Como su comportamiento solo se puede controlar a través de una señal PWM no había muchas opciones para modificar y corregir el mismo.

El eje de giro del servomotor se encuentra unido a un potenciómetro que, si se retira el tope físico, puede girar libremente. El potenciómetro funge como un divisor de voltaje que conforme va girando, hacia un lado u otro, va cambiando el valor del voltaje. El motor y el potenciómetro se encuentran conectados al driver o circuito integrado, el driver lee el valor del voltaje que tiene el potenciómetro y lo compara con la señal PWM que se le suministra, así el driver puede saber en qué posición se encuentra el eje y hacia qué lado debe girar el motor para alcanzar los grados correspondientes a la señal PWM recibida (ver Figura 4.3).

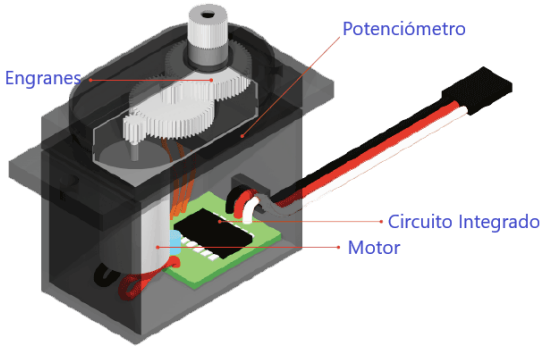


Figura 4.3 Estructura de un servomotor.
(Fuente: https://brainy-bits.com/wp-content/uploads/2018/01/Servo-Exploded.png. Último acceso: 23 de mayo de 2018)

Teniendo en cuenta que la estructura y cualidades físicas del servomotor eran las deseadas y conociendo más a fondo su funcionamiento, se decidió retirar el driver y el tope físico para poder manipular con mayor libertad su forma de operar. Para ello, el divisor de voltaje generado por el potenciómetro se conectó a uno de los puertos analógicos del ATMEGA328P-PU para monitorear la posición del eje (Figura 4.4).

Para disminuir el consumo de corriente por parte del potenciómetro, se propuso que fuera alimentado por uno de los pines del microcontrolador con tal de elegir en qué momento debía de estar activo el divisor de voltaje, para ello se tomaron las siguientes consideraciones.

De acuerdo con la hoja de especificaciones del ATMEGA328P-PU, cada pin puede suministrar una corriente máxima de 40 mA, y además cada pin analógico tiene una resistencia de entrada de 100 MΩ, por lo que:

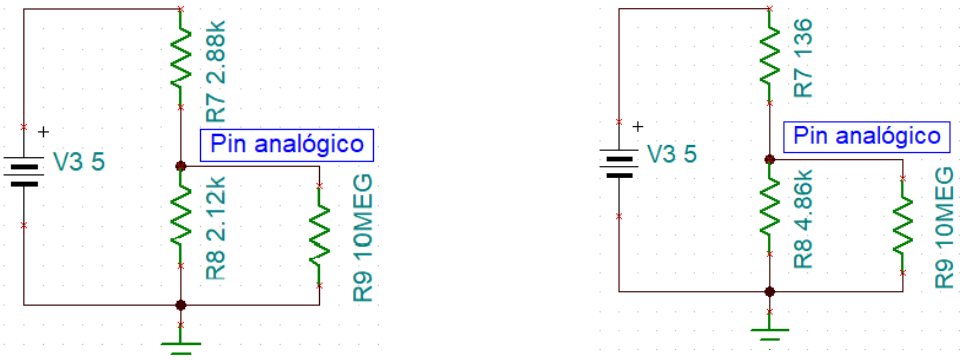


Figura 4.4 Circuitos equivalentes cuando la cerradura se encuentra cerrada y abierta respectivamente. (Fuente: Elaboración propia)

La resistencia de entrada del pin analógico, al ser muy grande, se puede asumir como un circuito abierto, resultando en el modelo de la Figura 4.5.

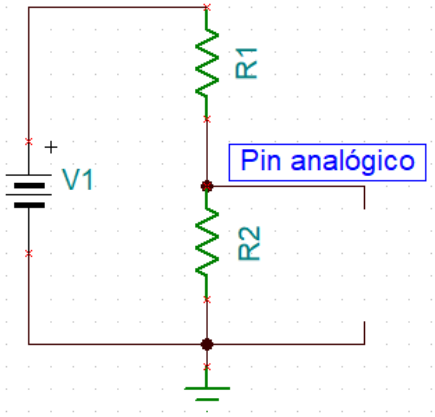


Figura 4.5 Modelo aproximado de entrada al pin analógico. (Fuente: Elaboración propia)

Donde, a través de la Ley de Ohm, se puede calcular la corriente que necesitará el potenciómetro:

$$V_1 = 5\ [V]$$

$$I_{pot} = \frac{V_1}{R_1 + R_2} \qquad R_1 + R_2 = R_{pot} = 5\ [k\Omega]$$

$$I_{pot} = \frac{V_1}{R_{pot}} = \frac{5}{5000} = 1\ [mA]$$

Por lo tanto, alimentar el potenciómetro a través de un pin analógico para disminuir el consumo de corriente es posible, ya que, aunque el potenciómetro varíe la fuente siempre tendrá como carga 5 kΩ.

Para controlar el motor se eligió el circuito integrado L293D (Figura 4.6) ya que con este componente se puede manipular el sentido de giro del motor por medio del

microcontrolador y, además el rango de voltaje de alimentación (4.5 V a 36 V) se ajusta muy bien al voltaje de trabajo de los demás componentes (5 V).

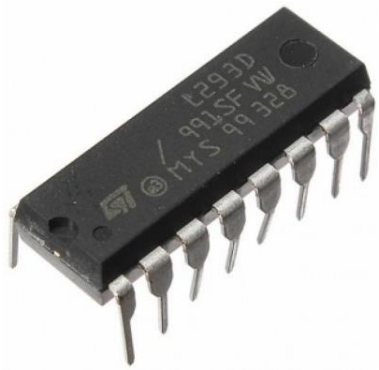


Figura 4.6 Circuito integrado L293D.
(Fuente: https://nomada-e.com/store/461-large_default/driver-para-motores-l293d.jpg. Último acceso: 24 de mayo de 2018)

Una vez resuelto el problema del motor, el siguiente punto a solucionar fue encontrar una manera para que el microcontrolador supiera en qué momento la puerta se encontraba abierta o cerrada, esto principalmente para que la cerradura no se cerrara en momentos que no correspondían.

Primeramente, se había utilizado y adaptado un push-button, que sería presionado cuando la puerta se cerrara avisando así al micro la acción realizada, sin embargo, su adaptación a la puerta era un poco complicada y no siempre era presionado cuando se cerraba la puerta, además que resultaba impráctico si se utilizaba en alguna puerta en la vida real.

Investigando, se encontraron los interruptores magnéticos que mediante un campo magnético pueden abrirse o cerrarse. Constan de dos piezas, un imán y el interruptor, una pieza se coloca en el marco de la puerta y la otra sobre la puerta a la misma altura, así cuando la puerta se cierre o se abra el interruptor se abrirá o se cerrará haciéndoselo saber al microcontrolador (ver Figura 4.7).

Figura 4.7 Interruptor magnético.
(Fuente: http://www.arduinosaltillo.denivel.com/wp-content/uploads/2017/10/proba4-1.jpg. Último acceso: 23 de mayo de 2018)

Por último, habría que resolver el problema de la alimentación. La mayoría de las cerraduras electrónicas actuales son alimentadas a través de baterías con una autonomía que puede alcanzar hasta 1 año, como el usuario no sabe en qué momento exacto va a necesitar recargar las baterías o cambiarlas, los fabricantes de estas cerraduras dejaron abierta la opción de poder utilizar las llaves para entrar o salir, resolviendo así el problema de impedir al usuario la entrada si las baterías se han agotado.

Como la cerradura presentada en este proyecto no puede ser manipulada físicamente desde el exterior, se debía garantizar el funcionamiento de esta en todo momento.

Inicialmente, se había pensado utilizar baterías recargables con cierta capacidad para que el sistema pudiera durar aproximadamente 3 meses antes de que necesitasen ser recargadas, no obstante, durante las pruebas de duración las baterías sólo lograron alimentar al sistema por 3 días antes de agotarse.

Finalmente, se decidió mantener una alimentación fija a través de un eliminador de 5 V y, a través de un relevador, cambiar la alimentación fija por baterías cuando por alguna razón no hubiera electricidad, con el propósito de mantener la cerradura en funcionamiento.

El relevador elegido para cumplir esta tarea fue el SRD-05VDC-SL-C de SONGLE (Figura 4.8), ya que el voltaje de funcionamiento de este se ajusta al voltaje de funcionamiento del microcontrolador y el motor.

Figura 4.8 Relevador SONGLE de 5 V.
(Fuente: http://teslabem.com/media/catalog/product/cache/1/image/636x/602f0fa2c1f0d1ba5e241f914e856ff9/r/e/relevador_de_5v_250vac_10a_teslabem.jpg. Último acceso: 24 de mayo de 2018)

Como se muestra en el circuito de la Figura 4.9, cuando haya electricidad, el relevador se encontrará funcionando gracias al eliminador de 5 V y a su vez, con el mismo eliminador, alimentará al microcontrolador, el Bluetooth y el motor.

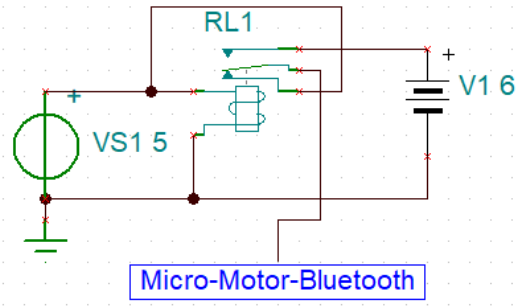


Figura 4.9 Relevador encendido, alimentación a través del eliminador.
(Fuente: Elaboración propia)

Como se muestra en la Figura 4.10, cuando no haya electricidad, el relevador se encontrará apagado y, aprovechando el pin normalmente cerrado (NC por sus siglas en inglés), el microcontrolador, el Bluetooth y el motor ahora serán alimentados por las baterías.

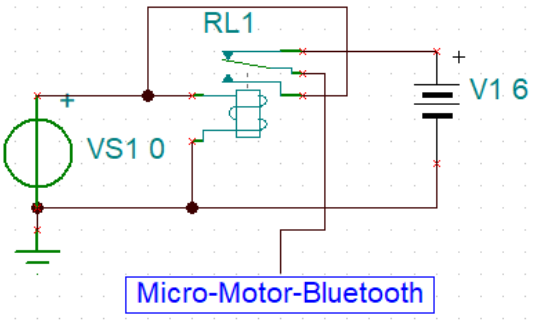


Figura 4.10 Relevador apagado, alimentación a través de las baterías.
(Fuente: Elaboración propia)

4.1 Desarrollo y Construcción de la PCB

Para el diseño del PCB se utilizó KiCad, un paquete de software de código abierto para la Automatización de Diseño Electrónico (EDA por sus siglas en inglés). Al ser un software de código abierto, un gran número de componentes se pueden encontrar en la web desarrollados por la comunidad que utiliza este programa, si el componente no se encuentra se puede crear o modificar fácilmente.

4.1.1 Creación del Diagrama Esquemático

El primer paso realizado para la construcción del PCB fue generar el diagrama esquemático (Figura 4.11), el cual es una representación gráfica de los circuitos eléctricos y electrónicos

constituido por símbolos correspondientes a los componentes y líneas que representan las conexiones entre ellos.

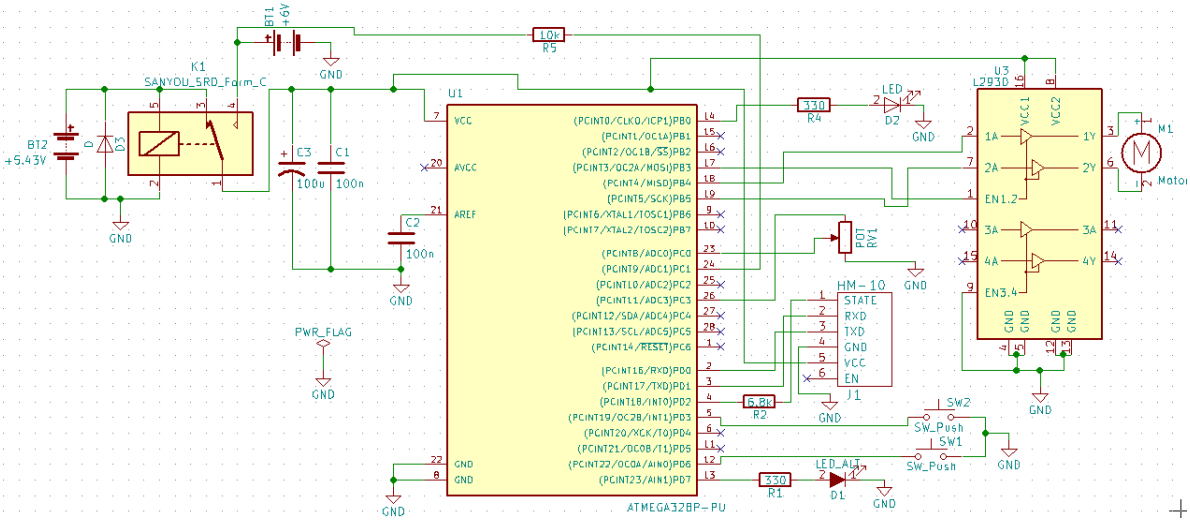


Figura 4.11 Diagrama esquemático de la cerradura electrónica.
(Fuente: Elaboración propia)

Una vez terminado el esquemático, a cada componente se le asoció una huella (Figura 4.12). Una huella o footprint es un boceto correspondiente a la ubicación y forma de los pines de cada componente.

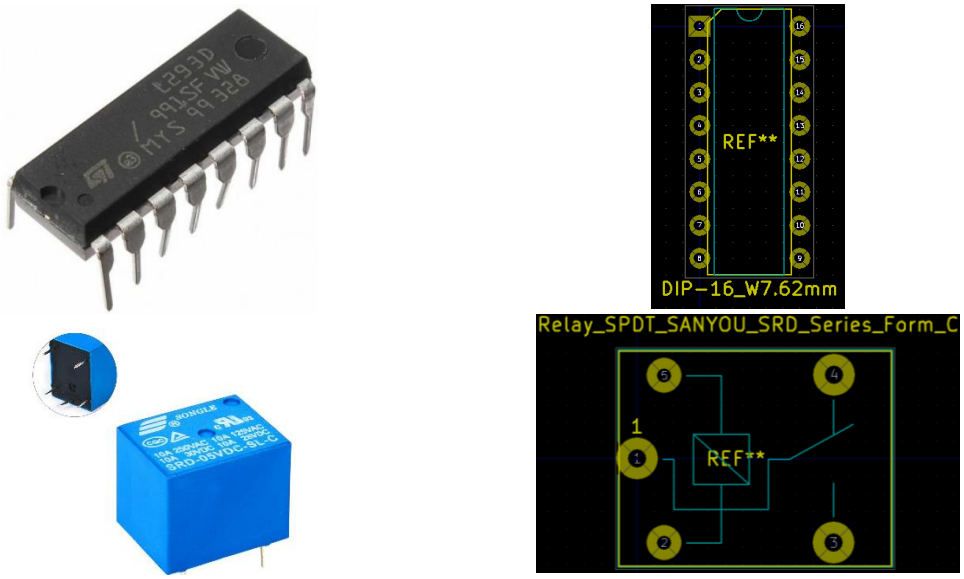


Figura 4.12 Ejemplos de huellas asociadas al driver L293D y relevador utilizados.
(Fuente: https://megaeshop.pk/media/catalog/product/cache/1/image/7dfa28859a690c9f1afbf103da25e678/1/0/10pcs-relay-5v-srd-5vdc-sl-c-t73-5v-font-b-songle-b-font-power-relay_1_.jpg. Último acceso: 24 de mayo de 2018)

4.1.2 Diseño de la PCB

Una vez asignadas las huellas a cada componente, se llevó a cabo el posicionamiento y conexión de cada uno de ellos, para ello se empleó la cara inferior de la placa de cobre (ver Figura 4.13).

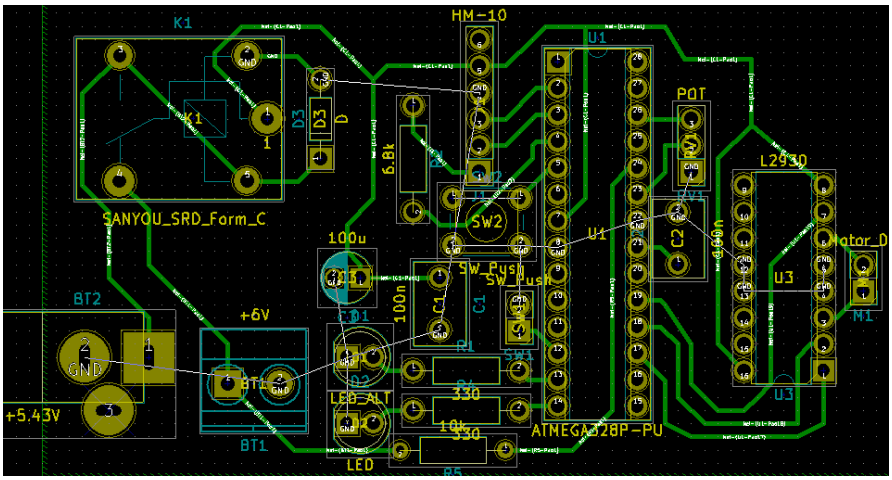


Figura 4.13 Posicionamiento y conexión de los componentes.
(Fuente: Elaboración propia)

Posteriormente, se trazó un dibujo incluyendo a todos los componentes y conexiones correspondientes, éste dibujo corresponderá a las medidas que deberá tener la placa de cobre.

Una vez trazado el dibujo, se procedió a conectar todas las tierras en una misma superficie de cobre, para ello fue necesario trazar otro dibujo dentro del ya elaborado, como se muestra en la Figura 4.14.

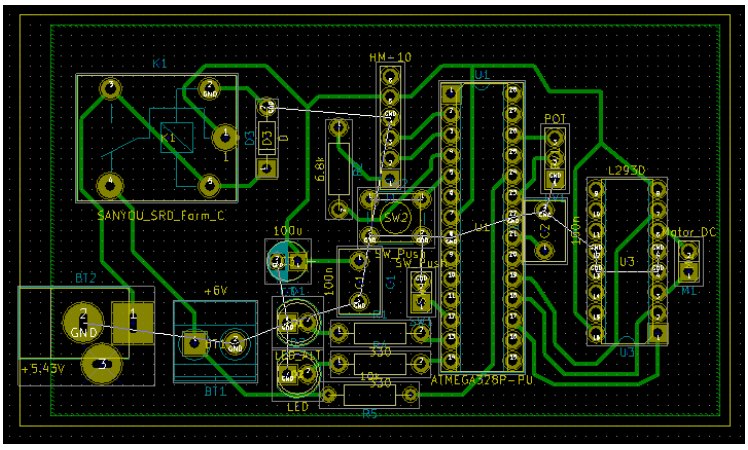


Figura 4.14 La línea amarilla corresponde al dibujo que tendrá la placa de cobre y la línea verde al área que cubrirá y conectará todas las tierras.
(Fuente: Elaboración propia)

Por último, se rellenó el área correspondiente a la tierra, terminando de esa manera el diseño de la PCB (Figura 4.15).

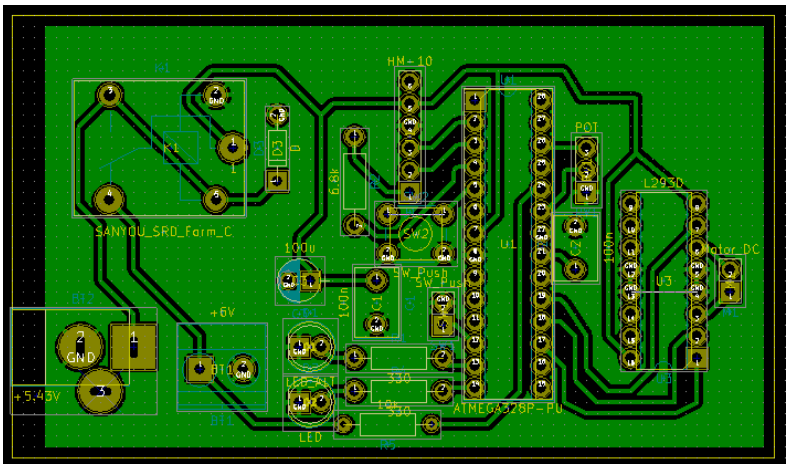


Figura 4.15 Diseño de las pistas y posicionamiento de los componentes en la PCB.
(Fuente: Elaboración propia)

Figura 4.16 Comparación modelo 3D y placa real vista desde arriba.
(Fuente: Elaboración propia)

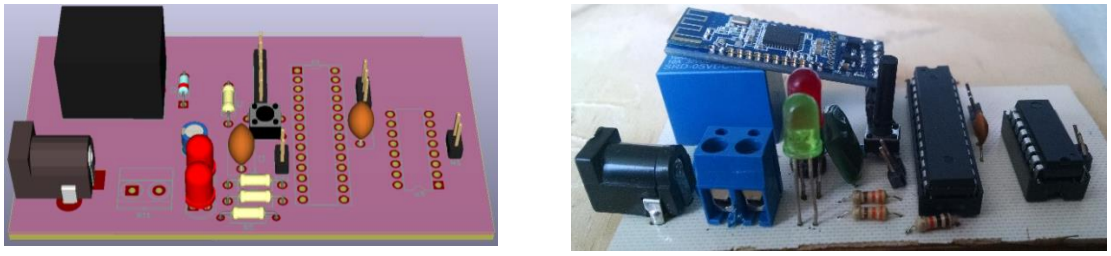


Figura 4.17 Comparación modelo 3D y placa real vista en perspectiva.
(Fuente: Elaboración propia)

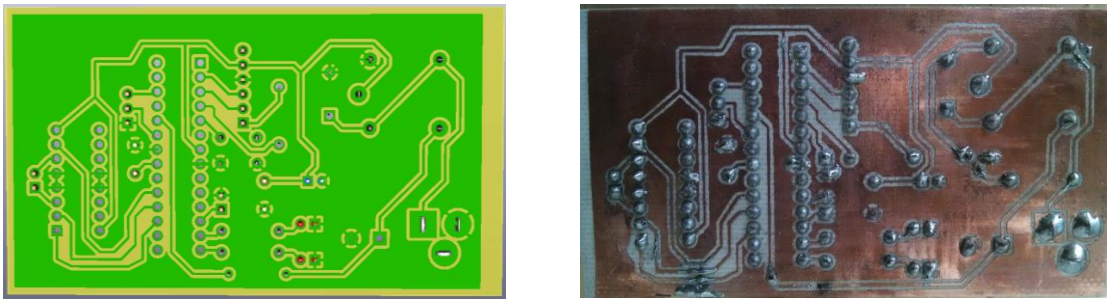


Figura 4.18 Comparación modelo 3D y placa real vista desde abajo.
(Fuente: Elaboración propia)

Entre el modelo 3D y la imagen real (ver Figuras 4.16 y 4.17) correspondiente a la Figura 4.18 se pueden apreciar ciertas diferencias con respecto al trazo de algunas de las pistas, esto se debe a que al final hubo algunas modificaciones en cuanto a algunos componentes, se decidió utilizar la placa que ya se tenía y no fabricarla de nuevo ya que las modificaciones eran mínimas. Sin embargo, ambos modelos cumplen la misma función.

4.2 Construcción de la Puerta y Montaje del Sistema Electrónico

Para poder probar la cerradura electrónica se construyó una pequeña puerta de madera, mostrada en la Figura 4.19, la cual pudiera simular una puerta de tamaño real.

Figura 4.19 Cara trasera de la puerta prototipo.
(Fuente: Elaboración propia)

De la cerradura llave-mariposa elegida sólo se utilizó la parte que contenía la barra de metal móvil, como se muestra en la Figura 4.20.

Figura 4.20 Barra metálica móvil de la cerradura.

La cerradura elegida se montó dentro de la puerta, mediante un orificio realizado a una altura media. A continuación, se hizo lo mismo con el marco con el propósito de que la barra lograra cerrar completamente (ver Figura 4.21).

Figura 4.21 Colocación de la barra metálica de la cerradura.
(Fuente: Elaboración propia)

Posteriormente, se hizo una pequeña perforación en la cara delantera de la puerta, a la altura de donde se encontraría el anclaje para el motor.

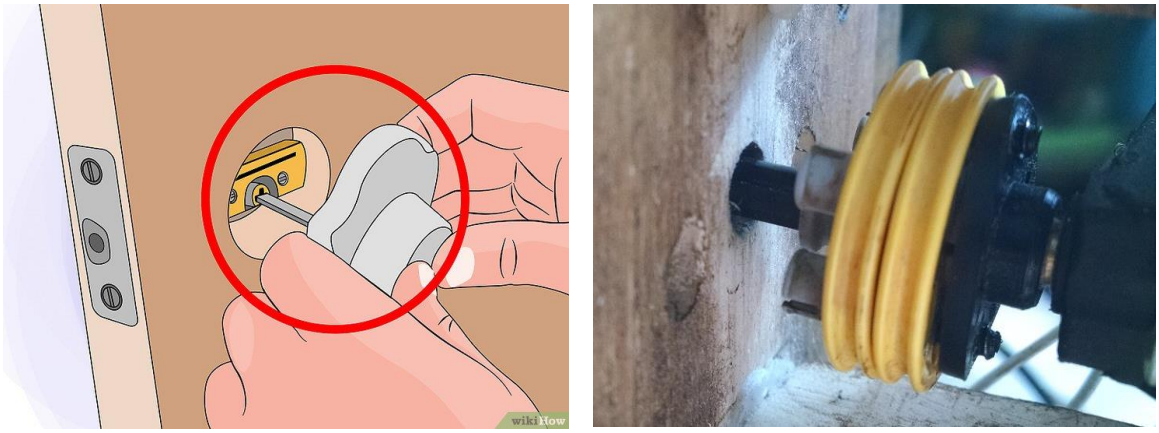


Figura 4.22 Orificio de anclaje de la cerradura y en la puerta prototipo.
(Fuente: https://www.wikihow.com/images_en/thumb/f/fc/Change-a-Lock-Step-16-Version-2.jpg/v4-728px-Change-a-Lock-Step-16-Version-2.jpg. Último acceso: 01 de julio de 2018)

Como se puede observar en la Figura 4.22, el acoplamiento entre la cerradura y el motor se realizó a través de una barra de plástico empotrada en dos discos reforzados con seguros del mismo material, estos se atornillaron a un cabezal el cual se acopla perfectamente a la

forma del eje del motor. Cabe mencionar que la barra, los discos y los seguros fueron piezas desarrolladas por LEGO[7].

Para mantener fijo el motor a la puerta se construyó un armazón de madera al cual se le realizó una perforación para permitir el paso de los cables. Este armazón fue pegado a la puerta empleando pegamento blanco (ver Figura 4.23).

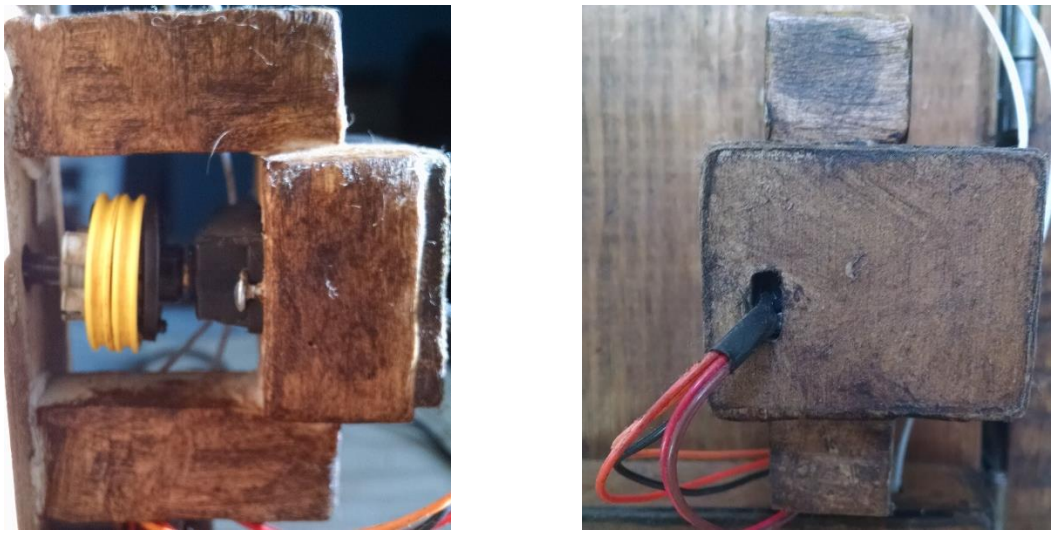


Figura 4.23 Armazón de madera del motor.
(Fuente: Elaboración propia)

La colocación del interruptor magnético se llevó a cabo como se describió en el Capítulo 3. Como se observa en la Figura 4.24, se instaló en la parte superior de la puerta, ya que, el espacio que existía entre ambas partes resultaba menor en comparación con el costado de la puerta.

Figura 4.24 Interruptor magnético colocado en la parte superior de la puerta.
(Fuente: Elaboración propia)

[7] LEGO es una empresa y marca de juguetes danesa reconocida principalmente por sus bloques de plástico interconectables.

Por último, para resguardar y mantener la PCB se construyó la caja mostrada en la Figura 4.25, utilizando papel batería; se empleó este material ya que es muy fácil de manejar y su alta porosidad permite unirlo mejor mediante el pegamento blanco.

La caja también fue pegada a la puerta para mantenerla fija cuando se abriera o cerrara.

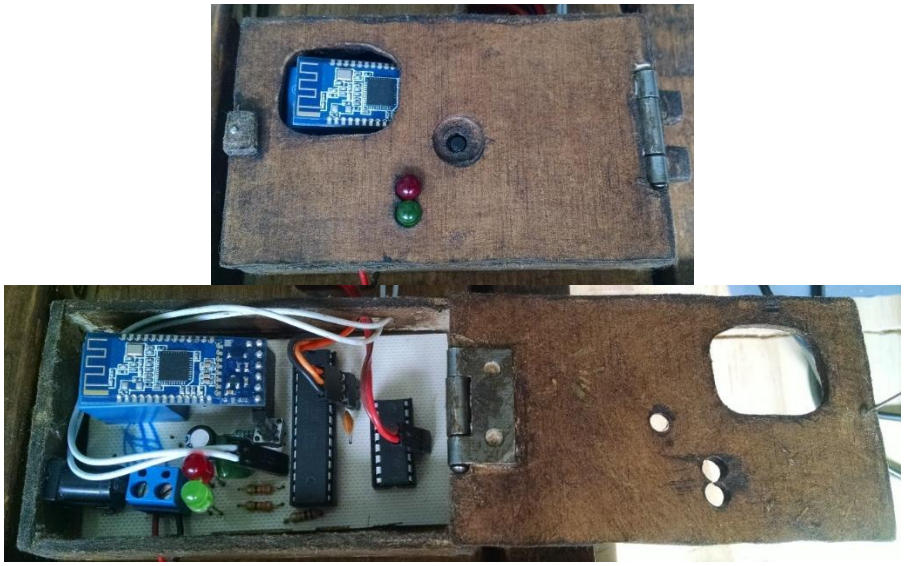


Figura 4.25 Caja contenedora de la PCB.
(Fuente: Elaboración propia)

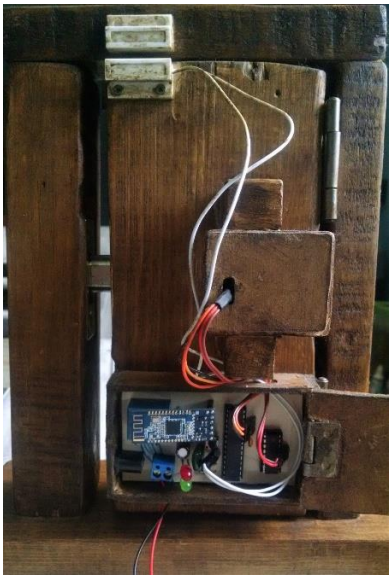


Figura 4.26 Prototipo final.
(Fuente: Elaboración propia)

En la Figura 4.26 se muestra el prototipo final de cerradura, desarrollado en este trabajo de tesis.

5 PRUEBAS Y RESULTADOS

En este capítulo se detallarán las pruebas realizadas y los resultados obtenidos a partir de ellas y tras la fabricación y construcción del prototipo.

5.1 CONFIGURACIÓN Y USO DE LA APLICACIÓN

Para que se pueda establecer la comunicación entre el dispositivo Android y el módulo Bluetooth se debe agregar el módulo a los dispositivos emparejados del dispositivo Android, con el objetivo de poder configurar la dirección del módulo en la aplicación. Esto se realiza sin mayor problema, sin embargo, una vez configurada la dirección en la aplicación, no se establece la conexión entre ambos dispositivos inmediatamente y es necesario el reinicio de la aplicación para que esta acción se produzca.

Una vez configurada y reiniciada la aplicación, se lleva a cabo una función de conexión automática entre el módulo y el dispositivo Android, si la conexión ha sido satisfactoria, un botón cambia de color gris a color verde, como se presenta en la Figura 5.1.

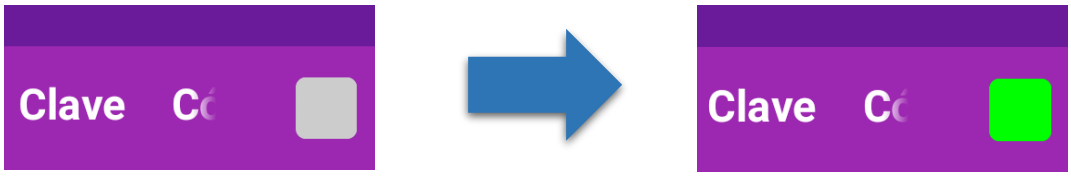


Figura 5.1 Confirmación de conexión establecida.
(Fuente: Elaboración propia)

La mayoría de los casos en que se abre la aplicación y se trata de establecer la conexión entre ambos dispositivos es satisfactoria, no obstante, hay casos en que se producen falsos positivos y es necesario reiniciar la aplicación para que se vuelva a intentar la conexión.

Una vez establecida la conexión, el paso siguiente es obtener un ID para poder utilizar la cerradura. Esto se logra ingresando una contraseña de 8 caracteres que solo el administrador conoce. Para que la asignación del ID se produzca al primer intento, es necesario que inmediatamente después que la dirección Bluetooth es configurada se cierre la aplicación y se destruya la actividad, ya que, si no se realiza de esa manera, la asignación del ID no se producirá hasta que se reinicie por segunda vez, la primera ocasión se produce durante la configuración del Bluetooth, ocasionando que el contador de ID siga aumentando sin asignar verdaderamente a nadie esos ID.

Este problema se le atribuye a la extensión provista por la misma plataforma de MIT App Inventor, ya que, a través de otra aplicación de la Google Play Store [8] llamada Serial Bluetooth Terminal[9], se comprobó que la asignación de ID se realizaba siempre al primer intento.

Posteriormente a la asignación del ID, se elige un nombre de usuario con el cual se registrarán sus entradas y salidas y se configura su código de acceso; estos pasos se realizan sin mayor inconveniente.

La parte correspondiente a la configuración de la dirección Bluetooth y de la asignación de ID son las únicas partes que causan problemas a la hora de configurar la aplicación, una vez superada la configuración, el uso de la aplicación para realizar entradas o salidas y para realizar el cambio del código de acceso resultan altamente fiables.

Concerniente al registro de entradas y salidas, como depende de la conexión a internet para actualizarse y aunque se implementó un método para guardar localmente el registro cuando no se hallara esta conexión, la actualización del registro se lleva a cabo de forma aleatoria, subiendo los datos algunas veces y algunas veces no.

Por último, las acciones especiales que puede realizar el administrador, entre las cuales están eliminar a un usuario, desbloquearlo, cambiar el nombre del módulo Bluetooth, así como el pin de emparejamiento, se ejecutan de manera correcta a excepción del desbloqueo; esta función no se logró corregir.

5.2 Desempeño de la Cerradura y sus Componentes

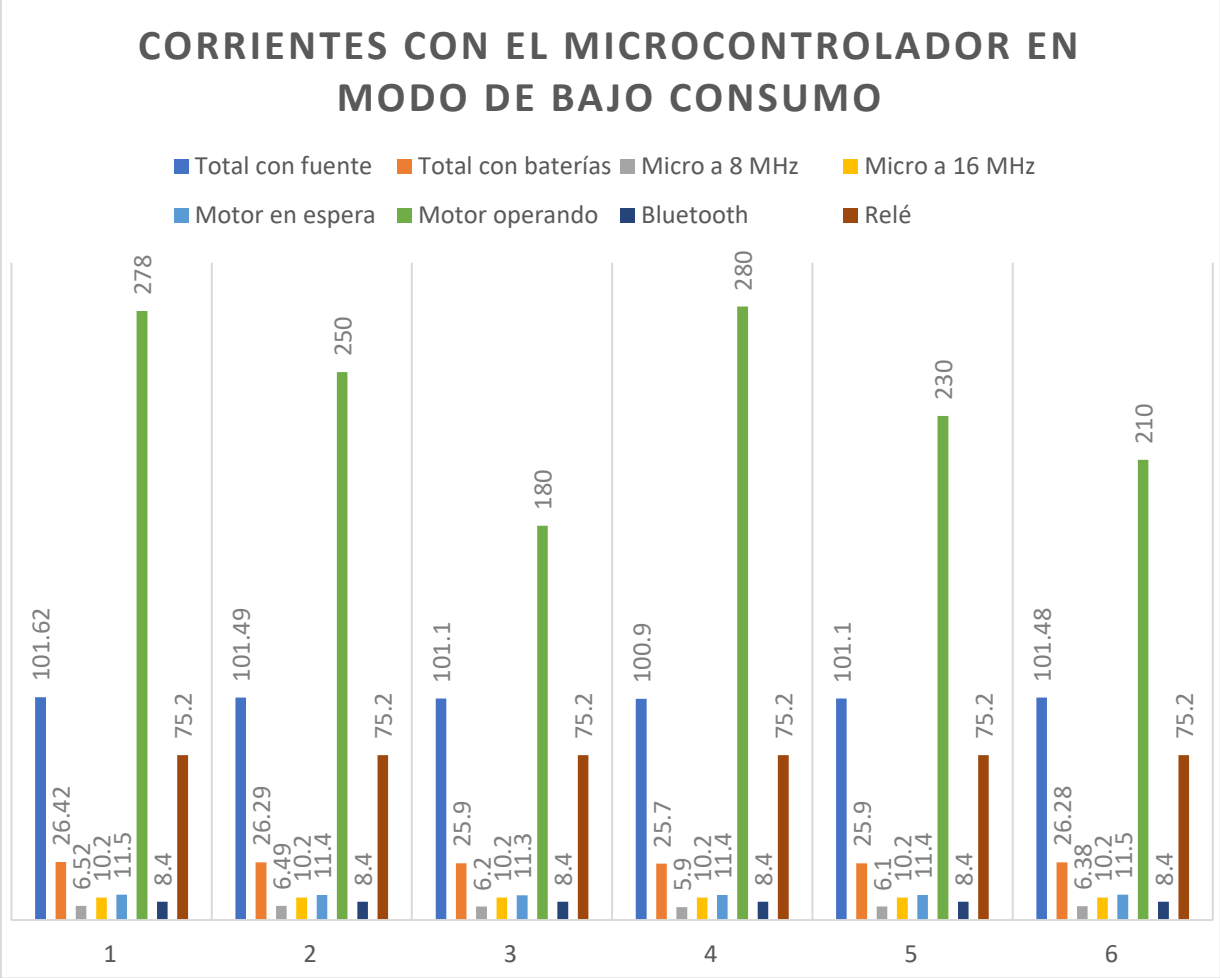
Gracias a las modificaciones que se le hicieron al motor, se tiene un mejor control de este y por ende un mejor desempeño; abre y cierra la cerradura sin intentar girar de más o girar menos de lo que debería.

El interruptor magnético también cumple su función de manera correcta, el único detalle que presenta es que, debido al campo magnético del imán, se necesitan separar ambas partes aproximadamente 2 cm para que se abra el interruptor y, de igual manera, estando a 2 cm del imán cuando la puerta está cerrándose se cierra el interruptor y se detecta que la puerta ha sido cerrada.

[8] La Google Play Store es una plataforma de distribución digital de aplicaciones móviles para los dispositivos con sistema operativo Android.
[9] Desarrollada por Kai Morich.

5.3 DESEMPEÑO ELÉCTRICO

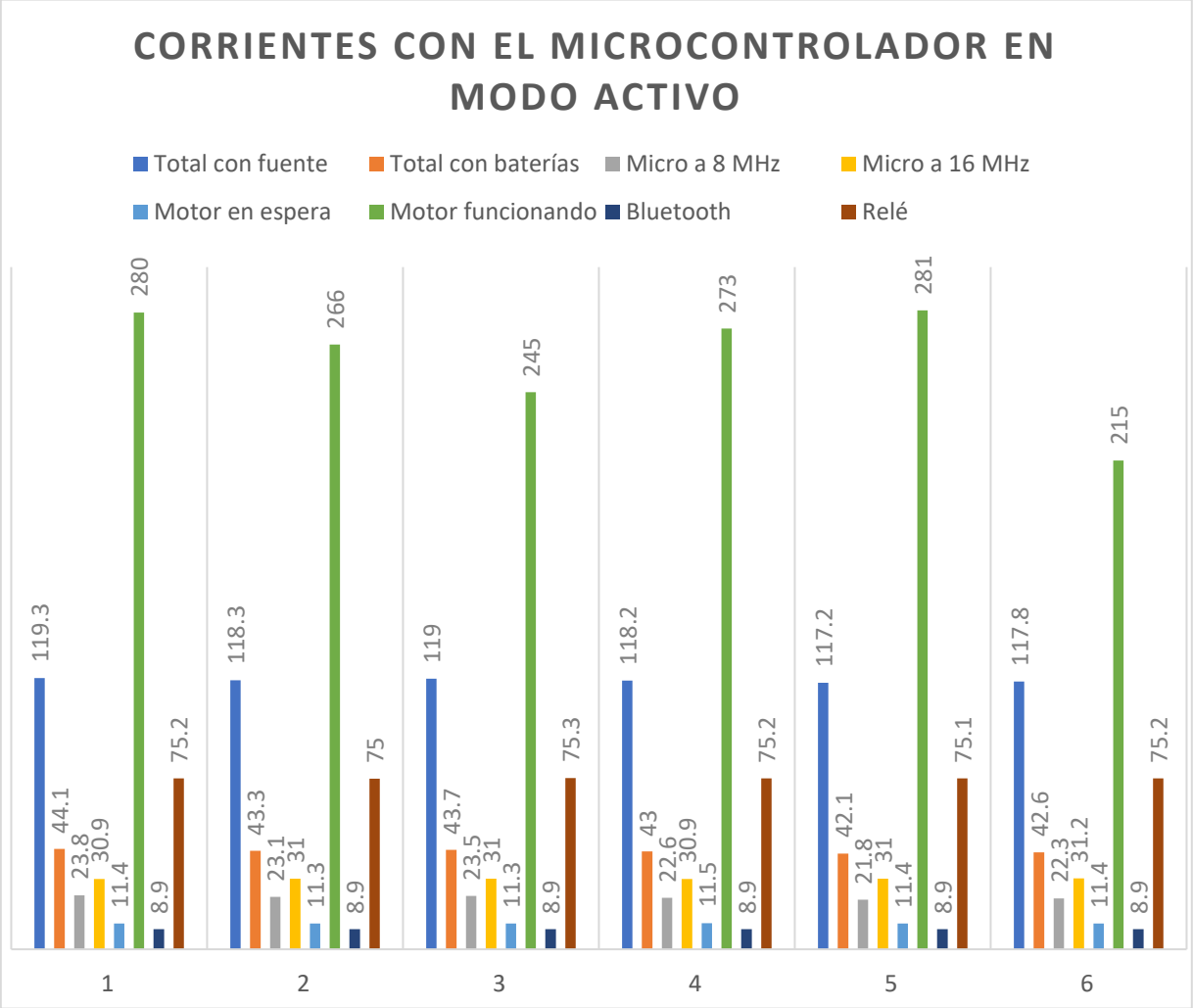
Debido a que el sistema dependerá de baterías (4 baterías AA de 1.2 V cada una) para su funcionamiento cuando la fuente principal no se encuentre en operación, fue necesario optimizar el sistema para consumir la menor corriente posible y con ello, extender la autonomía de éstas.



Gráfica 5.1 Corriente consumida, medida en miliamperes con intervalos de 10 minutos entre cada muestra. (Fuente: Elaboración propia)

En la gráfica 5.1 se pueden observar la corriente consumida por cada uno de los dispositivos que constituyen el sistema electrónico cuando el microcontrolador se encuentra en modo de bajo consumo, así como también el total consumido por todo el sistema cuando se encuentra alimentado por la fuente fija (primera barra de izquierda a derecha) y cuando se encuentra alimentado por las baterías (segunda barra). Cabe señalar que cuando el sistema

se encuentra alimentado por las baterías, el relevador se encuentra apagado y no consume corriente.



Gráfica 5.2 Corriente consumida, medida en miliamperes con intervalos de 10 minutos entre cada muestra. (Fuente: Elaboración propia)

En la gráfica 5.2 se muestran las mediciones de corriente de los dispositivos mencionados anteriormente, pero con el microcontrolador en modo activo. Es necesario destacar que la corriente consumida cuando el motor se encuentra en operación (sexta barra) no se considera dentro del total de corriente consumida tanto para el caso de la fuente fija como de las baterías, ya que esa corriente solo es consumida cuando se realiza una apertura o cierre de la cerradura y no es una corriente que se esté consumiendo constantemente.

Por último, se llevaron a cabo las pruebas para determinar la autonomía útil que las baterías pueden brindar al sistema.

Se utilizaron, como se mencionó anteriormente, 4 baterías AA de Ni-MH de 1.2 V de la marca Steren con una capacidad de 1300 mAh cada una por las siguientes razones:

- Para las pruebas fue necesario utilizar baterías recargables ya que, si se hubieran utilizado baterías alcalinas, por ejemplo, habría sido necesario comprar varios paquetes, lo cual resultaría en algo caro y contaminante para el ambiente.
- Las baterías de Níquel-Metal Hidruro resultan ser mejores en varios aspectos que las baterías de Níquel-Cadmio (otro tipo de batería recargable), como en la capacidad, el efecto memoria, etc.
- Las baterías de Steren resultan ser más accesibles a comparación de otras marcas como Duracell o Energizer.

Suponiendo que el sistema se encontrase en todo momento en modo de bajo consumo, es decir consumiendo un total de 26.12 mA, la autonomía de las baterías sería:

$$t_1 = \frac{1300}{26.12} = 49.77\ [h]$$

Lo que equivale a casi un día y medio de duración. Ahora, asumiendo que el sistema se hallase en modo activo en todo momento consumiendo un total de 43.21 mA, tenemos:

$$t_2 = \frac{1300}{43.21} = 30.08\ [h]$$

Equivalente a un día y dos horas de duración. Estos resultados teóricos podrían acercarse bastante a los reales si el sistema completo funcionara a niveles de voltaje más bajos. Sin embargo, debido principalmente al motor, ya que el módulo Bluetooth es capaz de operar hasta los 3.5 V y el microcontrolador hasta los 3.3 V, el tiempo de operación útil del sistema se ve mermado por el voltaje que necesita el motor para girar y mover la cerradura.

Estado	Resultado teórico [h]	Resultado práctico [h]
Modo bajo consumo	49.77	32
Modo activo	30.08	17

Tabla 5.1 Comparación de la autonomía de las baterías.
(Fuente: Elaboración propia)

Como se puede apreciar en la Tabla 5.1, los resultados prácticos difieren en gran medida y de forma negativa con respecto a los valores teóricos. Una posible solución temporal a este problema sería el de emplear baterías con una mayor capacidad de miliamperes por hora.

6 CONCLUSIONES

En este apartado se pretende hacer una recopilación de los objetivos propuestos y especificar el grado de efectividad logrado en su ejecución. Para ello, se listan de nuevo dichos objetivos con el fin de poder realizar dicha valoración:

- Diseñar e implementar un sistema electrónico que sea capaz de manipular una cerradura convencional.

Este objetivo se cumplió de manera satisfactoria, ya que el prototipo diseñado es capaz de accionar la cerradura elegida, ejecutando los ciclos de cierre y apertura de manera correcta y sin fallos mediante una aplicación basada en Android.

- Diseñar e implementar una aplicación basada en el sistema operativo Android, la cual sea capaz de manipular el sistema electrónico de la cerradura.

La aplicación diseñada, una vez configurada, cumple sin mayor problema su función, sin embargo, y debido a la plataforma elegida para desarrollarla, no era posible manejar de manera más precisa o a mayor conveniencia las funciones que el entorno provee, esto debido al estilo que posee la plataforma para programar. Además, conforme la aplicación fue creciendo en cuanto a funciones, el código también lo fue haciendo y como todo se desarrolla en una sola pantalla, entorpecía bastante la depuración del programa o cuando se buscaba corregir algún error de programación.

- Implementar un medio inalámbrico por el cual ambos sistemas, la cerradura y la aplicación Android, puedan comunicarse.

Establecer la comunicación a través de Bluetooth entre ambos dispositivos se logró de manera satisfactoria, sin embargo, y como se había mencionado en el apartado 4.1, la recepción de datos provenientes del microcontrolador al momento de configurar la aplicación no siempre era la necesaria, al igual que la conexión automática entre el módulo Bluetooth y el dispositivo Android. Estas fallas se le adjudican principalmente a la extensión utilizada para la implementación del Bluetooth 4.0 y a la nula posibilidad de manipular sus funciones. Cabe resaltar que para poder utilizar la aplicación, solo se puede conseguir mediante un dispositivo Android que posea la versión de Bluetooth 4.0 o superior.

- Diseñar y programar los métodos de seguridad necesarios para la manipulación de todo el sistema.

La utilización de una contraseña y un perfil de administrador, la implementación de un posicionamiento dinámico de los botones y la asignación aleatoria de números a estos botones cada vez que se cambia la contraseña, el cambio del nombre y pin de emparejamiento del módulo Bluetooth, la implementación de 5 oportunidades de acceso

previos a que el usuario sea bloqueado y la posibilidad de eliminar a un usuario enriquecen y fortalecen en gran medida la seguridad de la cerradura, a pesar de que el método de desbloqueo no se encuentre funcional y la actualización del registro no se realice el 100% de las ocasiones.

Cabe mencionar que aun así el sistema no resulta inviolable, ya que si se conoce el pin de emparejamiento o la dirección es posible enlazarse con el módulo y enviarle instrucciones para intentar lograr una apertura ilícita, aunque no sería una tarea sencilla conseguirlo ya que se tendrían que conocer también las estructuras de los paquetes de datos.

- Optimizar el sistema para que sea de bajo consumo energético.

Como se observó en las gráficas 4.1 y 4.2, hay una diferencia de aproximadamente 14 mA entre el modo activo y el modo de bajo consumo del microcontrolador, lo cual implica un gran ahorro en cuanto al consumo de corriente, sin embargo, el motor en espera, que concierne también al L293D, es el dispositivo que más consume corriente con 19.5 mA. Si se desarrollara una estrategia la cual alimentase al driver solo en los momentos requeridos, el consumo de corriente podría disminuir significativamente, afectando de manera positiva la autonomía de las baterías.

Otro punto importante es el modo de bajo consumo que posee el módulo Bluetooth, se intentó implementar este modo para que ambos dispositivos, el microcontrolador y el módulo, entraran en modo de bajo consumo cuando no se utilizara la cerradura, sin embargo, los métodos para activar el módulo, que consisten en enviarle una cadena de 20 caracteres o mediante un comando AT, necesitaban que el micro estuviera en modo activo y al estar también en modo de bajo consumo, no iba ser posible activar el módulo. También se intentó que cuando el módulo estuviera en modo de bajo consumo, el celular tratara de establecer la conexión con el mismo y así pudiera salir del modo de bajo consumo, algunas veces era exitosa la conexión y se activaba el módulo y en otras la aplicación marcaba un falso positivo. Por estas razones y porque el módulo consume una menor corriente en comparación con el microcontrolador, se decidió mantener el módulo siempre activo y que solo el microcontrolador entrara en modo de bajo consumo.

- Diseñar y construir el prototipo de la cerradura integrando ambos sistemas.

El prototipo desarrollado, como se ha podido comprobar a lo largo del presente trabajo, cumple en su gran mayoría con los objetivos propuestos y, en cuanto a las funciones principales de apertura y cierre, las tareas se llevan a cabo de manera casi perfecta.

Se trató que la cerradura poseyera un gran número de funciones con el objetivo de convertirla en un prototipo que estuviera al nivel de las cerraduras electrónicas de las grandes corporaciones, lo cual y en mi opinión, no creo que fuese una mala estrategia para comenzar a desarrollarla, no obstante y después de horas y horas de trabajo, considero que el mejor plan habría sido el de establecer las funciones principales y trabajar sobre ellas

hasta el punto en que quedaran perfectamente funcionales y, en un futuro, seguir desarrollando las demás funciones pero ya sobre un prototipo estable, en lugar de implementar un gran número de ellas dejando algunas a medio funcionar.

7 TRABAJO A FUTURO

- Eliminar el uso de dispositivos adicionales para el manejo del sistema y utilizar en su lugar sistemas biométricos, con el propósito de evitar los problemas que con estos se contraen.

- Hacer más eficientes los métodos de registro y eliminación de usuarios del sistema.

- Realizar un estudio más detallado con el objetivo de poder determinar el torque necesario para abrir la cerradura y, con base en los resultados, diseñar un sistema de engranaje que implique un motor más compacto.

- Incluir un sistema de recarga para las baterías con el fin de que estas puedan recuperarse mientras no son utilizadas u optimizar el sistema para depender sólo de baterías garantizando una autonomía de al menos 6 meses.

- Diseñar el sistema para ocupar el menor tamaño posible y que todas las partes se encuentren en un solo gabinete.

8 REFERENCIAS

[1] C. Franklin and J. Layton, "howstuffworks," [Online]. Available: https://electronics.howstuffworks.com/bluetooth.htm. [Accessed 12 Junio 2017].

[2] Jimb0, «sparkfun,» [En línea]. Available: https://learn.sparkfun.com/tutorials/serial-communication/serial-intro. [Último acceso: 4 Diciembre 2017].

[3] «Android Open Source Project,» [En línea]. Available: https://source.android.com/setup/. [Último acceso: 8 Junio 2017].

[4] «open handset alliance,» [En línea]. Available: https://www.openhandsetalliance.com/android_overview.html. [Último acceso: 8 Junio 2017].

[5] «Mc Graw Hill Education,» [En línea]. Available: https://www.mheducation.es/bcv/guide/capitulo/8448147227.pdf. [Último acceso: 12 Junio 2017].

[6] M. I. o. Technology, «MIT App Inventor,» [En línea]. Available: http://appinventor.mit.edu/explore/about-us.html. [Último acceso: 4 Enero 2018].

[7] T. H. Depot, «The Home Depot,» [En línea]. Available: http://www.homedepot.com.mx/ferreteria/cerraduras. [Último acceso: 6 Junio 2017].

[8] Samsung, «Samsung,» [En línea]. Available: https://www.samsungdigitallife.com. [Último acceso: 6 Junio 2017].

[9] Kickstarter, «Kickstarter,» [En línea]. Available: https://www.kickstarter.com/projects/candyhouse/sesame-your-key-reinvented?ref=nav_search&result=project&term=sesame. [Último acceso: 6 Junio 2017].

[10] A. Corporation, «Microchip,» 11 2016. [En línea]. Available: http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-42735-8-bit-AVR-Microcontroller-ATmega328-328P_Datasheet.pdf. [Último acceso: 25 Junio 2018].

[11] J.-M. Chung, «Coursera,» [En línea]. Available: https://www.coursera.org/lecture/wireless-communication-technologies/bluetooth-xbvzP. [Último acceso: 8 Junio 2017].

[12] T. Instruments, «Texas Instruments,» Junio 2013. [En línea]. Available: http://www.ti.com/lit/ds/symlink/cc2541.pdf. [Último acceso: 14 Junio 2017].

[13] T. Instruments, «Texas Instruments,» Enero 2016. [En línea]. Available: http://www.ti.com/lit/ds/symlink/l293.pdf. [Último acceso: 8 Mayo 2018].

[14] M. Inc., «Microchip Technology,» 2017. [En línea]. Available: http://ww1.microchip.com/downloads/en/AppNotes/00002447A.pdf. [Último acceso: 22 Junio 2018].

[15] A. Corporation, «Microchip Technology,» Abril 2015. [En línea]. Available: https://www.microchip.com/webdoc/AVRLibcReferenceManual/index.html. [Último acceso: 14 Marzo 2018].

[16] R. Irani, «Romin Irani's Blog,» 9 Septiembre 2016. [En línea]. Available: https://rominirani.com/tutorial-mit-app-inventor-firebase-4be95051c325. [Último acceso: 22 12 2017].

[17] D. Jahshan, P. Hutchinson, F. Tappero, C. Jarron y M. Van den Berg, «KiCad EDA,» 24 Agosto 2017. [En línea]. Available: http://docs.kicad-pcb.org/stable/es/getting_started_in_kicad.pdf. [Último acceso: 12 Mayo 2018].

9 ANEXOS

9.1 CÓDIGO DEL MICROCONTROLADOR ATMEGA328P-PU

```
#define F_CPU 8000000   //Velocidad del reloj del microcontrolador
#define BAUD 9600                     //Número de bits por segundo
#define BRC ((F_CPU/16/BAUD) - 1)           //Cálculo del Baud Rate

#include <avr/io.h>
#include <util/delay.h>
#include <avr/eeprom.h>
#include <stdbool.h>
#include <avr/wdt.h>
#include <avr/sleep.h>
#include <avr/interrupt.h>

//Declaración de variables y arreglos

char aux[14];
uint8_t cont = 0;
uint8_t auxInt[9];
bool error = false;
bool newPass = false;
bool sleepFlag = false;
bool nomBT = false;
bool passBT = false;
char help[10]   = {'@','j','>',']','2','S','p','h','J','m'};
char nombre[15] = {'A','T','+','N','A','M','E'};
char pass[13]   = {'A','T','+','P','I','N'};
float Vcc_value = 0;
bool battery_flag = false;
uint16_t pot = 0;


//*************************************************************
//                    Declaración de funciones
//*************************************************************

//Función para enviar datos tipo char
void USARTWriteChar(unsigned char data)
{
  while(!(UCSR0A & ( 1 << UDRE0)))    //Esperamos hasta que el
                                      buffer de transmisión esté
                                      vacío
  {
  }
```

```
  UDR0 = data;                          //Se escriben los datos en el
                                        buffer del USART
}

//Función para cambiar el nombre del Bluetooth
void nombreBT()
{
  uint8_t i,j;

  for (i = 1;i <= (cont - 2);i++)
  {
   nombre[i + 6] = aux[i];              //Se guarda en un arreglo el
                                        nombre enviado por el
                                        administrador
  }

  for (j = 0;j <= (i + 5);j++)
  {
   USARTWriteChar(nombre[j]);           //Se envía el nombre junto el
                                        comando AT correspondiente
                                        para cambiarlo
  }

  USARTWriteChar('\r');
  USARTWriteChar('\n');
  cont = 0;
  error = false;
}

//Función para cambiar el pin de vinculación del Bluetooth
void passwordBT()
{
  uint8_t i;

  for (i = 1;i <= (cont - 2);i++)
  {
   pass[i + 5] = aux[i];                //Se guarda en un arreglo el
                                        pin eviado por el
                                        administrador
  }

  for (i = 0;i <= 11;i++)
  {
   USARTWriteChar(pass[i]);             //Se envía el pin junto el
                                        comando AT correspondiente
                                        para cambiarlo
  }

  USARTWriteChar('\r');
  USARTWriteChar('\n');

   cont = 0;
```

```
}

//Función modo sleep
void goToSleep()
{
  if (!(PIND & (1 << PD2)))      //Si el Bluetooth no está
                                 conectado a un dispositivo
  {
   if (nomBT)
   {
    nombreBT();
    _delay_ms(100);
    nomBT = false;
   }
   if (passBT)
   {
    passwordBT();
    _delay_ms(100);
    passBT = false;
   }

   sleepFlag = true;

   uint8_t adcsra = ADCSRA;      //Se guarda el registro A de
                                 control y estado del ADC
   PORTB &= ~(1 << PORTB0);
   PORTD &= ~(1 << PORTD7);

   Battery_flag = false;

   ADCSRA = 0;                   //Se deshabilita el ADC
   SMCR = (1 << SE);             //Se habilita el modo sleep
   sleep_bod_disable();          //Se deshabilita el Brown Out
                                 Detector
   sei();                        //Se habilita la bandera de
                                 interrupciones globales
   sleep_cpu();                  //El dispositivo entra en modo
                                 sleep
   SMCR = (0 << SE);
   ADCSRA = adcsra;
  }
}

//Función para enviar datos tipo byte
void USARTWriteByte(uint8_t data)
{
  while(!(UCSR0A & (1 << UDRE0)))     //Esperamos hasta que el
  {                                   buffer de transmisión esté
  }                                   vacío

  UDR0 = data;                        //Se extraen los datos en el
                                      buffer del USART
```

```
}

//Función para el monitoreo de las baterías
void battery_sampling()
{
  //Se configure el registro del ADC para que tome como referencia
el voltaje AVcc, y que el canal de entrada sea el puerto ADC1)
  ADMUX |= (1 << MUX0);
  //Se habilita el ADC e indica que comience la conversión
  ADCSRA |= (1 << ADEN) | (1 << ADSC);

  uint8_t cont = 0;
  Vcc_value = 0;

  while (cont <= 100)                //Ciclo para tomar un total
                                     de 100 muestras
  {
   if (ADCSRA & (0x01 << ADIF))      //Si la conversión se ha
                                     completado
   { //Cálculo para saber el nivel de voltaje
    VCC_value += (ADCL + (ADCH * 255));
    cont++;
    _delay_ms(10);
   }
  }
Vcc_value = Vcc_value / 100;         //Se calcula el promedio de
                                     las conversiones leídas

if (Vcc_value >= 820)                //Si el voltaje llega a un
                                     nivel crítico
{
  PORTB |= (1 << PORTB0);            //Se enciende LED rojo
  PORTD &= ~(1 <<PORTD7);
}
else
{
  PORTB &= ~(1 << PORTB0);           //Se enciende LED verde
  PORTD |= (1 <<PORTD7);
}

ADCSRA &= ~(1 <<ADEN);               //Se deshabilita el ADC
ADCSRA &= ~(1 <<ADSC);

battery_flag = true;
}

//Función para abrir la cerradura
void abrir()
{
  uint8_t tiempo = 0;
```

```
  PORTC |= (1 << PORTC3);              // Se activa el puerto para
                                       que se pueda realizar el
                                       divisor de voltaje en el
                                       pot

  //Se cambia el registro del ADC para que la entrada sea a través
del pin asignado al ADC0;
  ADMUX &= ~(1 << MUX0);
  ADCSRA |= (1 << ADEN) | (1 << ADSC);

  PORTB |= (1 << PORTB3);             //Se habilita el driver del
                                      motor

  while (1)
  {
   _delay_ms(10);

   if (ADCSRA & (0x01 << ADIF))
   {
    pot = ADCL + (ADCH * 256);        //Se lee el nivel de voltaje
                                      correspondiente del
                                      potenciometro
   }

   if (pot < 970)                     //Si el valor leído es menor
                                      a 970, se abre la cerradura
   {
    PORTB |= (1 << PORTB5);
   }
   else                               //De lo contrario, la
                                      cerradura ya se encuentra
                                      abierta y no se hace nada
   {
    PORTB &= ~(1 << PORTB5);
    PORTB &= ~(1 << PORTB3);
    break;
   }
  }

  ADCSRA &= ~(1 << ADEN);
  PORTC &= ~(1 << PORTC3);            //Se desactiva el divisor de
                                      voltaje del pot

  while (!(PIND & (1 << PIND6)))      //Mientras la puerta
                                      permanezca cerrada los LEDs
                                      parpadearán a cierta
                                      velocidad
  {
   _delay_ms(500);
   PORTD |= (1 << PORTD7);
   PORTB &= ~(1 << PORTB0);
   _delay_ms(500);
```

```
   PORTD &= ~(1 << PORTD7);
   PORTB |= (1 << PORTB0);

   tiempo++;                          //Se toma el tiempo que ha
                                      transcurrido desde que se
                                      abrió la cerradura

   if (tiempo == 30)                  //Si el tiempo excede los 30
                                      segundos se termina el método
                                      de apertura
   {
    break;
   }
  }
  _delay_ms(50);
}

//Función para cerrar la cerradura
void cerrar()
{
  while (PIND & (1 << PIND6))         //Mientras la puerta se
                                      encuentre abierta los LEDs
                                      parpadearán a mayor velocidad
  {
   _delay_ms(200);
   PORTD |= (1 << PORTD7);
   PORTB &= ~(1 << PORTB0);
   _delay_ms(200);
   PORTD &= ~(1 << PORTD7);
   PORTB |= (1 << PORTB0);
  }

  _delay_ms(1000);


  PORTC |= (1 << PORTC3);             //Se activa el divisor de
                                      voltaje en el pot
  ADCSRA |= (1 << ADEN) | (1 << ADSC);     //Se activa el ADC para
                                           leer el valor del
                                           potenciometro
  PORTB |= (1 << PORTB3);             //Se active el driver del
                                      motor

  while (1)
  {
   _delay_ms(10);

   if (ADCSRA & (0x01 << ADIF))
   {
    pot = ADCL + (ADCH * 256);        //Se lee el valor del ADC
   }
```

```
   if (pot > 550)                    //Si es mayor a 550 se active
                                     el motor para que cierre la
                                     cerradura
   {
    PORTB |= (1 << PORTB4);
   }
   else
   {
    PORTB &= ~(1 << PORTB4);
    PORTB &= ~(1 << PORTB3);
    break;
   }
  }

  ADCSRA &= ~(1 << ADEN);            //Se desactiva el ADC
  PORTC &= ~(1 << PORTC3);           //Se desactiva el divisor de
                                     voltaje del pot

  PORTB &= ~(1 << PORTB0);           //Se desactiva el driver

  _delay_ms(500);

  battery_sampling();                //Se hace un análisis del
                                     estado de las baterias
}

//Función para inicializar el USART
void USARTInit(uint16_t ubrr_value)
{
  UBRR0H = (unsigned char)(ubrr_value >> 8);    //Se guardan los 4
                                                bits más
                                                significativos de
                                                ubbr_value
  UBRR0L = (unsigned char)ubrr_value;           //Se guardan los 8
                                                bits menos
                                                significativos de
                                                ubbr_value
  UCSR0B = (1 << RXEN0) | (1 << TXEN0);         //Se habilitan el
                                                transmisor y
                                                receptor del USART
  UCSR0C = (1 << USBS0) | (3 << UCSZ00);        //Se configura el
                                                número de bits de
                                                parada que debe
                                                insertar el
                                                transmisor
}

//Función para recibir datos tipo char
unsigned char USARTReadChar(void)
{
```

```
  while(!(UCSR0A & (1 << RXC0)))      //Ciclo que espera a que la
                                      bandera de datos sin leer se
                                      levante
  {
   if (!battery_flag)
   {
    battery_sampling();               //Se realiza un muestreo del
                                      estado de las baterias
                                      momentos después que un
                                      usuario se desvincula del
                                      módulo Bluetooth
   }

   goToSleep();                       //Mientras se esperan los
                                      datos se ejecuta el método
                                      para entrar al modo sleep
  }

  return UDR0;                        //Se retorna el dato
                                      almacenado en el buffer del
                                      USART
}

//Función para asignar ID
void asignarId()
{
  uint8_t val = eeprom_read_byte((uint8_t*)0);  //Se lee el valor
                                                guardado en la
                                                localidad "0" de
                                                la memoria EEPROM

  USARTWriteByte(val);                //Se envía el valor leído
  val++;                              //Se incrementa el valor
                                      leído en 1
  eeprom_update_byte((uint8_t*)0,val);     //Se almacena el nuevo
                                           valor en la EEPROM
}

//Función para convertir valores tipo char a entero
void convertir()
{
  uint8_t i;

  //Se compara el identificador
  if ((aux[0] == '.') || (aux[0] == '|') || (aux[0]) == '-')
  {
   for (i = 1;i <= 8;i++)
   {
    auxInt[i - 1] = aux[i] - 48;      //En un arreglo se almacenan
                                      los nuevos valores restándole
                                      48 a los valores tipo char
   }
```

```
  }
  else
  {
   for (i = 3;i <= 8;i++)
   {
    auxInt[i - 3] = aux[i] - 48;
   }
  }
}

//Función para agregar un nuevo usuario, cambiar contraseña de
administrador y desbloquear usuarios
void nuevoUsuario()
{
  uint8_t i;
  uint8_t no = 101;

  for (i = 0;i <=7;i++)
  {
   //Se compara la contraseña recibida con la almacenada por el
administrador
   if (auxInt[i] != eeprom_read_byte((uint8_t*)(i + 702)))
   {
    error = true;
   }
  }

  if (!error && (aux[0] == '.'))        //Si la contraseña es
                                        correcta
  {
   no = 7;
   newPass = true;                    //Se configura la variable
                                      "newPass" para escribir una
                                      nueva contraseña
   USARTWriteByte(no);                //Se envía confirmación
  }
  else if (!error && (aux[0] == '|'))      //Si la contraseña es
                                           correcta
  {
   //Se extrae el ID de la traza recibida
   uint8_t id = ((aux[9] - 48) * 10) + (aux[10] - 48);
   no = 9;
   USARTWriteByte(no);                //Se envía confirmación
   no = 0;
   //Se configura localidad de la EEPROM correspondiente al ID
recibido para desbloquearlo
   eeprom_update_byte((uint8_t*)(id + 601), no);
  }
  else if (!error && aux[0] == '-')        //Si la contraseña es
                                           correcta
  {
```

```
   asignarId();                          //Se ejecuta la función para
                                         asignar un ID
 }
  else                                   //Si la contraseña es
                                         incorrecta
  {
   USARTWriteByte(no);                   //Se envía código de
                                         incorrecto
  }
}

//Función para guardar los códigos de los usuarios
void grabar()
{
  uint8_t id = ((aux[1] - 48) * 10) + (aux[2] - 48);  //Se extrae
                                                      el ID de la
                                                      traza
                                                      recibida
//Se almacena el código recibido en las localidades
correspondientes al ID recibido
  eeprom_update_byte((uint8_t*)(id + 1), auxInt[0]);
  eeprom_update_byte((uint8_t*)(id + 101), auxInt[1]);
  eeprom_update_byte((uint8_t*)(id + 201), auxInt[2]);
  eeprom_update_byte((uint8_t*)(id + 301), auxInt[3]);
  eeprom_update_byte((uint8_t*)(id + 401), auxInt[4]);
  eeprom_update_byte((uint8_t*)(id + 501), auxInt[5]);
}

//Función para comparar los códigos de los usuarios al entrar o
salir
void comparar(uint8_t id)
{
  uint8_t yes = 1;
  uint8_t no = 0;

  if (!error)                            //Si el código es correcto
  {
   switch (aux[0])
   {
    case '#':
    USARTWriteByte(correct);             //Se envía que fue correcto
//Se limpian los intentos fallidos del usuario en la memoria
EEPROM asignada
    eeprom_update_byte((uint8_t*)(id + 601), no);
    abrir = true;
    abiertoCerrado();                    //Se ejecuta la función para
                                         abrir la cerradura
    break;

    case ';':
    USARTWriteByte(yes);                 //Se envía confirmación
    break;
```

```
   }
  }
  else                                  //Si el código no fue
                                        correcto
  {
   switch (aux[0])
   {
    case '#':
    USARTWriteByte(no);                 //Se envía que fue incorrecto
//Se lee el número de errores que ha cometido el usuario
    no = eeprom_read_byte((uint8_t*)(id + 601));
    no++;                               //Se incrementa la variable
                                        en 1
//Se guarda el número de errores cometidos en la memoria EEPROM
    eeprom_update_byte((uint8_t*)(id + 601),no);
    break;

    case ';':
    USARTWriteByte(no);                 //Se envía que fue incorrecto
    break;
   }
  }
}

//Función para buscar un ID
void buscarID()
{
//Se extrae el ID de la traza recibida
  uint8_t id = ((aux[1] - 48) * 10) + (aux[2] - 48);

//Se lee el número de usuarios registrados de la EEPROM
  uint8_t val = eeprom_read_byte((uint8_t*)0);

//Se lee el número de errores cometidos por el usuario
  uint8_t op = eeprom_read_byte((uint8_t*)(id + 601));

  if ((id >= val))                      //Si el ID recibido es mayor
                                        al número de usuarios
                                        registrados
  {

   USARTWriteByte(4);                   //Se envía que el usuario no
                                        está registrado
  }
  else if (op >= 5)                     //Si el número de errores del
                                        usuario es mayor o igual a 5
  {
   USARTWriteByte(8);                   //Se envía que el usuario
                                        está bloqueado
  }
  else if (eeprom_read_byte((uint8_t*)(id + 1)) == 10)
  {
```

```
   USARTWriteByte(10):                  //Se envía que el usuario fue
                                        eliminado
  }
  else                                  //Si el usuario está
                                        registrado y no se encuentra
                                        bloqueado
  {
   convertir();                         //Se ejecuta la función
                                        convertir


//Se compara el código recibido del usuario con el código guardado
   if (auxInt[0] == eeprom_read_byte((uint8_t*)(id + 1)))
   {
    if (auxInt[1] == eeprom_read_byte((uint8_t*)(id + 101)))
    {
     if (auxInt[2] == eeprom_read_byte((uint8_t*)(id + 201)))
     {
      if (auxInt[3] == eeprom_read_byte((uint8_t*)(id + 301)))
      {
       if (auxInt[4] == eeprom_read_byte((uint8_t*)(id + 401)))
       {
        if (auxInt[5] == eeprom_read_byte((uint8_t*)(id + 501)))
        {
        }
        else
        {
         error = true;
        }
       }
       else
       {
        error = true;
       }
      }
      else
      {
       error = true;
      }
     }
     else
     {
      error = true;
     }
    }
    else
    {
```

```
    error = true;
   }

     comparar(id);                          //Se ejecuta la función
                                            comparar
  }
}

//Función para borrar a un usuario
void borrar()
{
  uint8_t id = ((aux[1] - 48) * 10) + (aux[2] - 48);  //Se extrae
                                                       el ID de la
                                                       traza
                                                       recibida

  if ((id >= eeprom_read_byte((uint8_t*)0)))     //Si el ID
                                                 recibido es mayor
                                                 al número de
                                                 usuarios
                                                 registrados
  {
   USARTWriteByte(4);                 //Se envía que el usuario no
                                      está registrado
  }
  else                                //Si el usuario está
                                      registrado
  {
   res = 5;
//Se resetean las localidades de la EEPROM asignadas al ID
recibido
   eeprom_update_byte((uint8_t*)(id + 1), 10);
   eeprom_update_byte((uint8_t*)(id + 101), 10);
   eeprom_update_byte((uint8_t*)(id + 201), 10);
   eeprom_update_byte((uint8_t*)(id + 301), 10);
   eeprom_update_byte((uint8_t*)(id + 401), 10);
   eeprom_update_byte((uint8_t*)(id + 501), 10);

   USARTWriteByte(5);                 //Se envía que el usuario ha
                                      sido borrado
  }
}

//Función para realizar una entrada de emergencia
void emergencia()
{
  uint8_t id = ((aux[1] - 48)*10) + (aux[2] - 48);    //Se extrae
                                                       el ID de la
                                                       traza
                                                       recibida
  uint8_t i, res;
```

```
  uint8_t emer = eeprom_read_byte((uint8_t*)701);     //Se lee el
                                                      número de
                                                      accesos de
                                                      emergencia
                                                      disponibles

  if ((id >= eeprom_read_byte((uint8_t*)0)))     //Si el ID
                                                 recibido es mayor
                                                 al número de
                                                 usuarios
                                                 registrados
  {
   res = 4;
   USARTWriteByte(res);                  //Se envía que el usuario no
                                         está registrado
  }
  else if (emer > 0)                     //Si aún hay intentos
                                         disponibles
  {
   for (i = 0;i <= 9;i++)
   {
    if (help[i] != aux[i + 3])           //Si la contraseña de
                                         emergencia introducida es
                                         diferente a la almacenada
    {
     error = true;                       //Se establece la variable
                                         "error" como true
    }
   }
   if (!error)                           //Si no hubo error
   {
    res = 6;
    eeprom_update_byte((uint8_t*)701, (emer - 1));    //Se resta 1
                                                      a los
                                                      intentos de
                                                      emergencia
    USARTWriteByte(res);                 //Se envía que la contraseña
                                         de emergencia fue correcta
    abrir = true;
    abrir();                             //Se ejecuta función para
    cerrar();                            abrir la cerradura
   }
  }
  else
  {
  }
}

//Función para guardar la contraseña del administrador
void guardarPass()
{
```

```
  if (newPass)                          //Si la variable "newPass" es
                                        verdadera
  {
//Se guarda en la memoria EEPROM la nueva contraseña del
administrador
   eeprom_update_byte((uint8_t*)702, (aux[1] - 48));
   eeprom_update_byte((uint8_t*)703, (aux[2] - 48));
   eeprom_update_byte((uint8_t*)704, (aux[3] - 48));
   eeprom_update_byte((uint8_t*)705, (aux[4] - 48));
   eeprom_update_byte((uint8_t*)706, (aux[5] - 48));
   eeprom_update_byte((uint8_t*)707, (aux[6] - 48));
   eeprom_update_byte((uint8_t*)708, (aux[7] - 48));
   eeprom_update_byte((uint8_t*)709, (aux[8] - 48));
  }

  newPass = false;
}

//Función para inicializar el ADC
void adc_init()
{
  ADMUX = (1 << REFS0) | (1 << MUX0);
  ADCSRA = (1 << ADATE) | (1 << ADPS2) | (1 << ADPS1);
}

//Función para resetear todos los valores
void reset_all()
{
  cont = 0;
  error = false;
  newPass = false;
  sleepFlag = false;
  nomBT = false;
  passBT = false;
  Vcc_value = 0;
  battery_flag = false;
  pot = 0;
  wdt_disable();
  USARTInit(BRC);
  adc_init();
  DDRD = 0b10000000;
  DDRB = 0b00111011;
  PORTD = 0b01001000;
  EIMSK = (1 << INT1) | (1 << INT0);
  EICRA = (1 << ISC01) | (1 << ISC00);
  SMCR = (1 << SM1);
  sei();
}

int main(void)
{
  wdt_disable();                        //Se desactiva el Watchdog
```

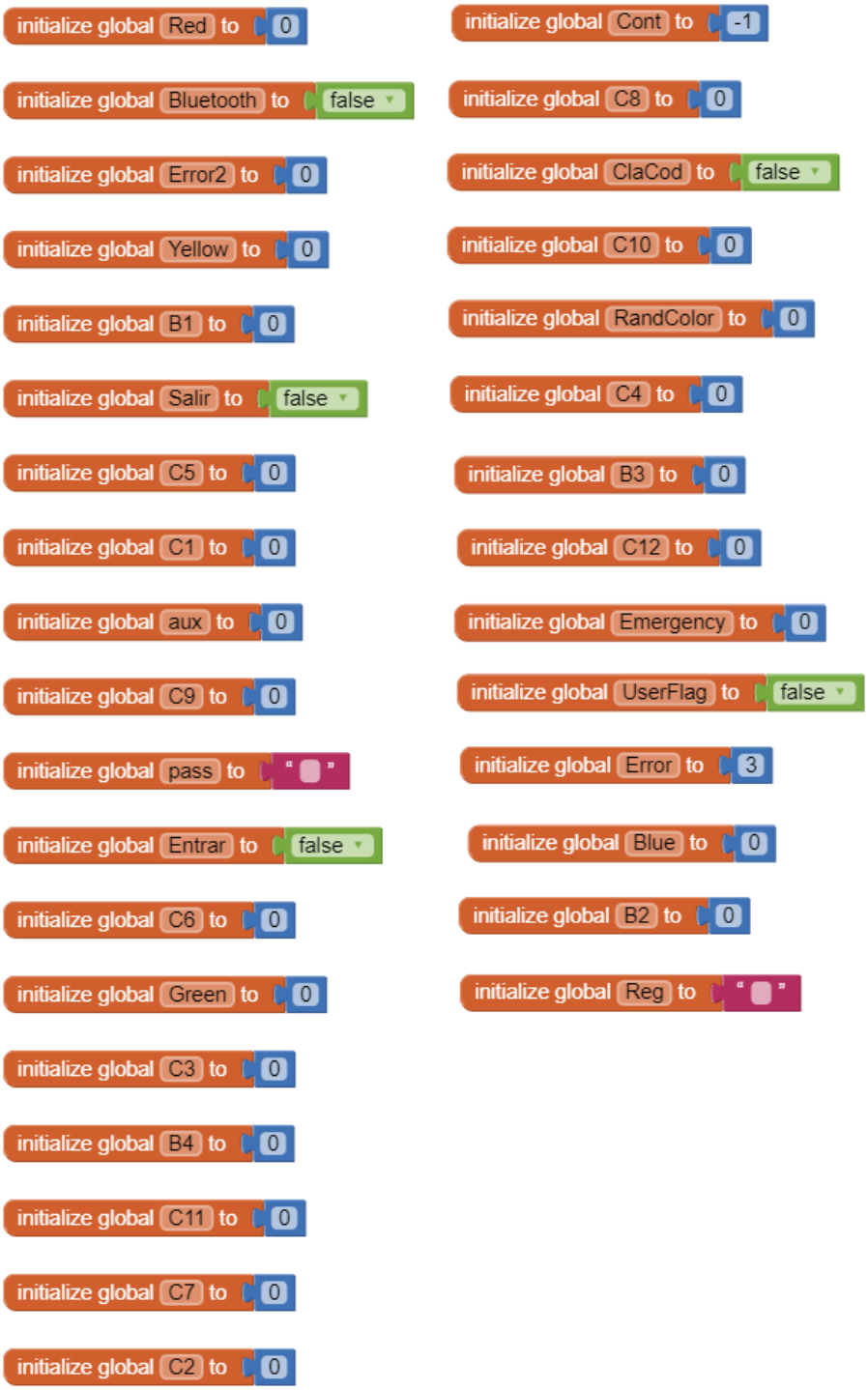
```
  USARTInit(BRC);                       //Se inicializa el USART
  adc_init();                           //Se inicializa el ADC

  DDRD = 0b10000000;                    //Se configuran el puerto PD7
                                        como salida y los demás
                                        puertos como entradas

  DDRB = 0b00111011;                    // Se configuran los puertos
                                        PB0 PB3, PB4, PB5 como
                                        salidas
  DDRC = 0b00001000;                    // Se configura el puerto PC3
                                        como salida

  PORTD = 0b01001000;                   //Se activa la resistencia de
                                        pull-up en el puerto PD6 y
                                        PD3

  EIMSK = (1 << INT1) | (1 << INT0);  //Se activa la interrupción
                                        externa INT0 e INT1
  EICRA = (1 << ISC11) | (1 << ISC01) | (1 << ISC00); //Se
                                             configura el modo de
                                             activación rising edge
                                             para la interrupción
                                             INT0 y falling Edge para
                                             INT1

  SMCR = (1 << SM1);                    //Se configura el sleep mode
                                        1 (power-down)
  sei();                                //Se habilita la bandera de
                                        interrupciones globales


  while (1)
  {

   aux[cont] = USARTReadChar();         //Se leen los datos recibidos
                                        y se almacenan en un arreglo
   cont++;                              //Se incrementa el contador
                                        del arreglo aux

//Método para solicitar un ID
   if (aux[0] == '-' && cont > 8)
   {
    convertir();
    nuevoUsuario();
    cont = 0;
    error = false;
   }

//Método para guardar código de un usuario
   else if (aux[0] == '*' && cont > 8)
   {
```

```
    convertir();
    grabar();
    cont = 0;
    error = false;
   }

//Método para buscar un ID y aprobar o no la entrada/salida de un
usuario
   else if (aux[0] == '#' && cont > 8)
   {
    buscarID();
    cont = 0;
    error = false;
   }

//Método para cambiar el código de un usuario
   else if (aux[0] == ';' && cont > 8)
   {
    buscarID();
    cont = 0;
    error = false;
   }

//Método para cambiar la contraseña del administrador
   else if (aux[0] == ':' && cont > 8)
   {
    guardarPass();
    cont = 0;
    error = false;
   }

//Método para eliminar a un usuario
   else if (aux[0] == '=' && aux[cont -1] == '/')
   {
    borrar();
    cont = 0;
    error = false;
   }

//Método para una apertura de emergencia
   else if (aux[0] == '?' && aux[cont - 1] == '/')
   {
    emergencia();
    cont = 0;
    error = false;
   }

//Método para cambiar el nombre del módulo Bluetooth
   else if (aux[0] == '&' && aux[cont - 1] == '/')
   {
    nomBT = true;
   }
```

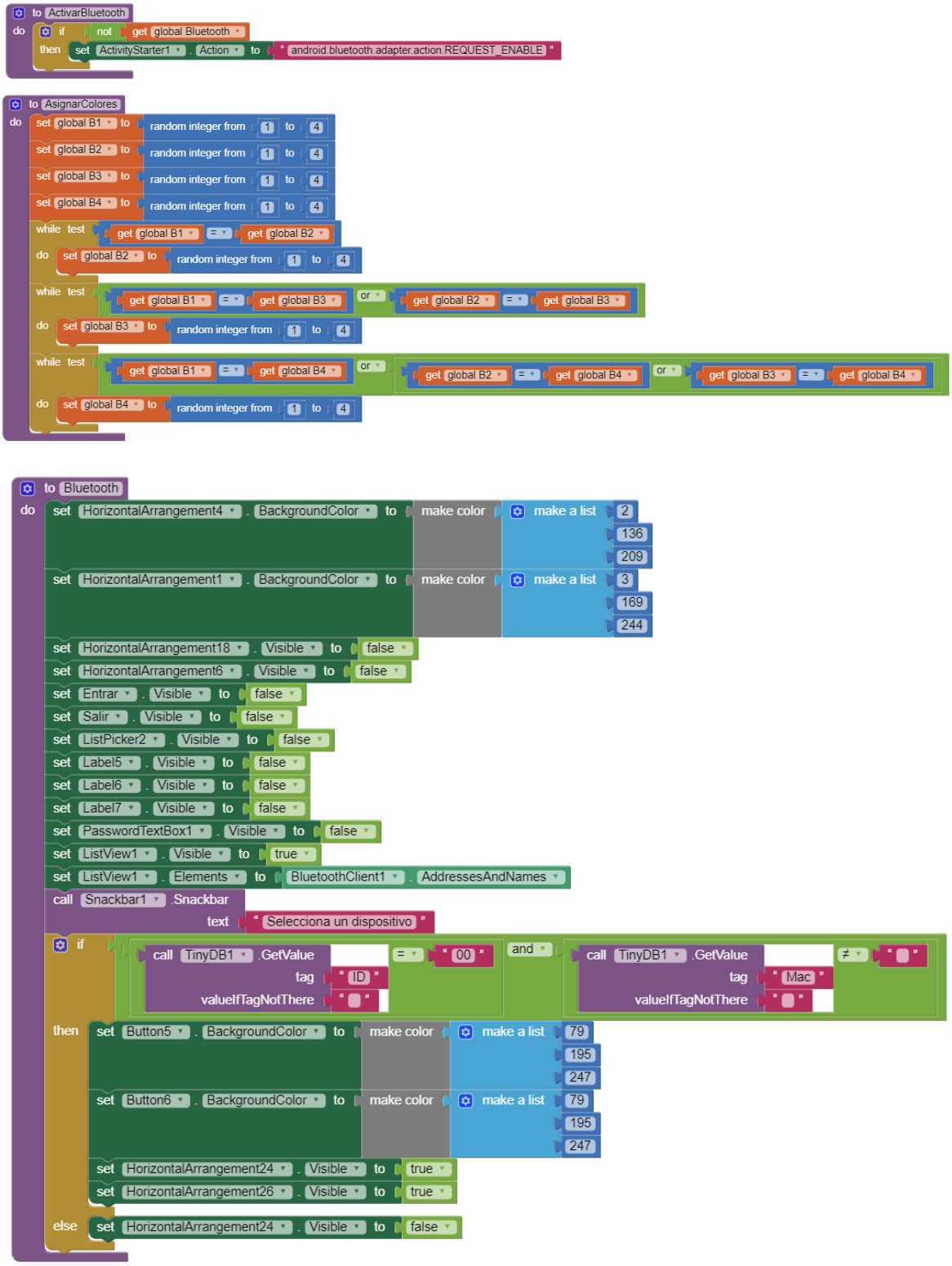
```
//Método para cambiar la contraseña de vinculación del Bluetooth
   else if (aux[0] == '!' && aux[cont - 1] == '/')
   {
    passBT();
   }

//Método para solicitar el cambio de contraseña del administrador
   else if (aux[0] == '.' && cont > 8)
   {
    convertir();
    nuevoUsuario();
    cont = 0;
    error = false;
   }

//Método para desbloquear a un usuario
   else if (aux[0] == '|' && cont > 8)
   {
    convertir();
    nuevoUsuario();
    cont = 0;
    error = false;
   }
  }
}

//Manejo de interrupciones
ISR(INT0_vect){

  if (sleepFlag)                    //Si la variable "sleepFlag"
                                    es verdadera

  {
   sleepFlag = false;               //Se configura en falso la
                                    variable
   reset_all();                     //Se resetean todos los
                                    valores
  }

  EIFR |= (1 << INTF0);             //Se limpia la bandera de la
                                    interrupción generada
}

ISR (INT1_vect){                    // Si el botón en el puerto
                                    PD3 es presionado se ejecuta
                                    interrupción e inicia
                                    apertura

 reset_all();

 _delay_ms(200);

 abrir();
```

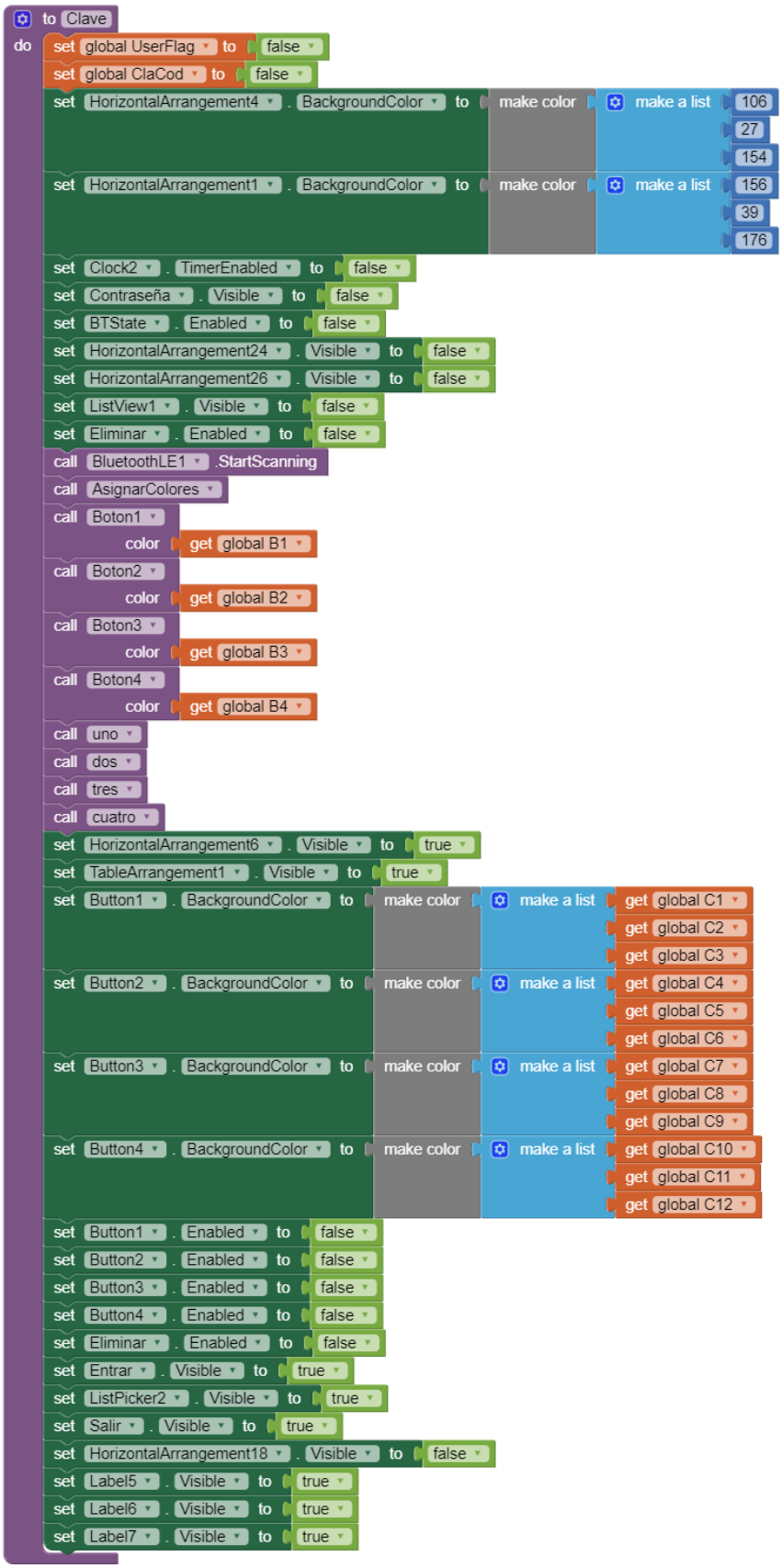
```
 cerrar();
 goToSleep();
  EIFR |= (1 << INTF1);
}
```

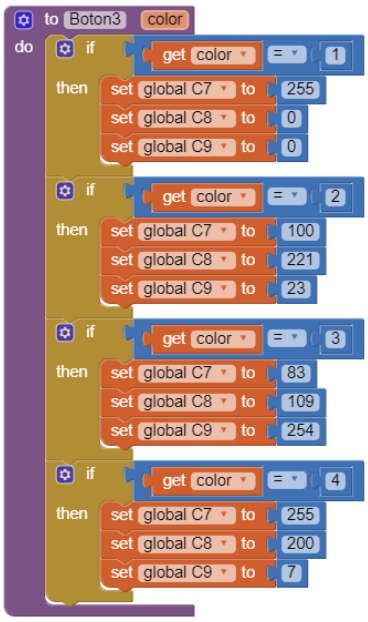
9.2 CÓDIGO DE LA APLICACIÓN ANDROID

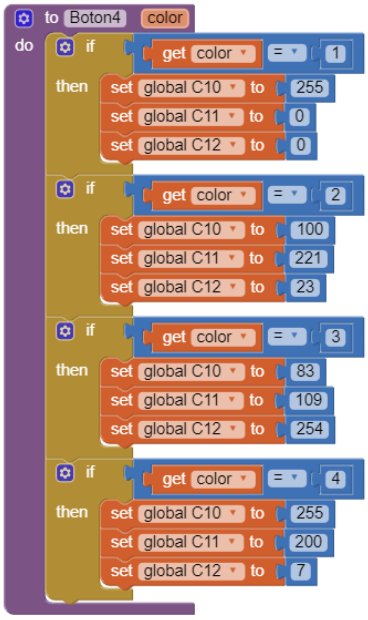



to Arcoiris
do
set global RandColor to random integer from 1 to 13
if get global RandColor = 1
then set Hora . TextColor to make color make a list 230 0 0
else if get global RandColor = 2
then set Hora . TextColor to make color make a list 230 120 0
else if get global RandColor = 3
then set Hora . TextColor to make color make a list 230 218 0
else if get global RandColor = 4
then set Hora . TextColor to make color make a list 46 230 0
else if get global RandColor = 5
then set Hora . TextColor to make color make a list 0 230 132
else if get global RandColor = 6
then set Hora . TextColor to make color make a list 0 230 230
else if get global RandColor = 7
then set Hora . TextColor to make color make a list 0 132 230
else if get global RandColor = 8
then set Hora . TextColor to make color make a list 0 0 230
else if get global RandColor = 9
then set Hora . TextColor to make color make a list 138 0 230
else if get global RandColor = 10
then set Hora . TextColor to make color make a list 213 0 230
else if get global RandColor = 11
then set Hora . TextColor to make color make a list 230 0 120
else if get global RandColor = 12
then set Hora . TextColor to make color make a list 230 0 5


to ActivarBluetooth
do if not get global Bluetooth
then set ActivityStarter1 . Action to " android.bluetooth.adapter.action.REQUEST_ENABLE "
to AsignarColores
do set global B1 to random integer from 1 to 4
set global B2 to random integer from 1 to 4
set global B3 to random integer from 1 to 4
set global B4 to random integer from 1 to 4
while test get global B1 = get global B2
do set global B2 to random integer from 1 to 4
while test get global B1 = get global B3 or get global B2 = get global B3
do set global B3 to random integer from 1 to 4
while test get global B1 = get global B4 or get global B2 = get global B4 or get global B3 = get global B4
do set global B4 to random integer from 1 to 4
to Bluetooth
do set HorizontalArrangement4 . BackgroundColor to make color make a list 2 136 209
set HorizontalArrangement1 . BackgroundColor to make color make a list 3 169 244
set HorizontalArrangement18 . Visible to false
set HorizontalArrangement6 . Visible to false
set Entrar . Visible to false
set Salir . Visible to false
set ListPicker2 . Visible to false
set Label5 . Visible to false
set Label6 . Visible to false
set Label7 . Visible to false
set PasswordTextBox1 . Visible to false
set ListView1 . Visible to true
set ListView1 . Elements to BluetoothClient1 . AddressesAndNames
call Snackbar1 .Snackbar
text " Selecciona un dispositivo "
if call TinyDB1 .GetValue tag " ID " valueIfTagNotThere " " = " 00 " and call TinyDB1 .GetValue tag " Mac " valueIfTagNotThere " " ≠ " "
then set Button5 . BackgroundColor to make color make a list 79 195 247
set Button6 . BackgroundColor to make color make a list 79 195 247
set HorizontalArrangement24 . Visible to true
set HorizontalArrangement26 . Visible to true
else set HorizontalArrangement24 . Visible to false

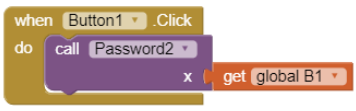

to Clave
do
set global UserFlag to false
set global ClaCod to false
set HorizontalArrangement4 . BackgroundColor to make color make a list 106 27 154
set HorizontalArrangement1 . BackgroundColor to make color make a list 156 39 176
set Clock2 . TimerEnabled to false
set Contraseña . Visible to false
set BTState . Enabled to false
set HorizontalArrangement24 . Visible to false
set HorizontalArrangement26 . Visible to false
set ListView1 . Visible to false
set Eliminar . Enabled to false
call BluetoothLE1 .StartScanning
call AsignarColores
call Boton1
color get global B1
call Boton2
color get global B2
call Boton3
color get global B3
call Boton4
color get global B4
call uno
call dos
call tres
call cuatro
set HorizontalArrangement6 . Visible to true
set TableArrangement1 . Visible to true
set Button1 . BackgroundColor to make color make a list get global C1 get global C2 get global C3
set Button2 . BackgroundColor to make color make a list get global C4 get global C5 get global C6
set Button3 . BackgroundColor to make color make a list get global C7 get global C8 get global C9
set Button4 . BackgroundColor to make color make a list get global C10 get global C11 get global C12
set Button1 . Enabled to false
set Button2 . Enabled to false
set Button3 . Enabled to false
set Button4 . Enabled to false
set Eliminar . Enabled to false
set Entrar . Visible to true
set ListPicker2 . Visible to true
set Salir . Visible to true
set HorizontalArrangement18 . Visible to false
set Label5 . Visible to true
set Label6 . Visible to true
set Label7 . Visible to true

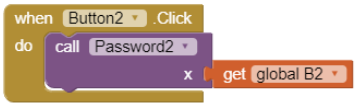

to Boton3 color
do
if get color = 1
then set global C7 to 255
set global C8 to 0
set global C9 to 0
if get color = 2
then set global C7 to 100
set global C8 to 221
set global C9 to 23
if get color = 3
then set global C7 to 83
set global C8 to 109
set global C9 to 254
if get color = 4
then set global C7 to 255
set global C8 to 200
set global C9 to 7


to Boton4 color
do
if get color = 1
then set global C10 to 255
set global C11 to 0
set global C12 to 0
if get color = 2
then set global C10 to 100
set global C11 to 221
set global C12 to 23
if get color = 3
then set global C10 to 83
set global C11 to 109
set global C12 to 254
if get color = 4
then set global C10 to 255
set global C11 to 200
set global C12 to 7

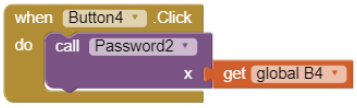

to tres
do
if get global B3 = 1
then set global B3 to call TinyDB1 .GetValue
tag "Red"
valueIfTagNotThere ""
else if get global B3 = 2
then set global B3 to call TinyDB1 .GetValue
tag "Green"
valueIfTagNotThere ""
else if get global B3 = 3
then set global B3 to call TinyDB1 .GetValue
tag "Blue"
valueIfTagNotThere ""
else if get global B3 = 4
then set global B3 to call TinyDB1 .GetValue
tag "Yellow"
valueIfTagNotThere ""

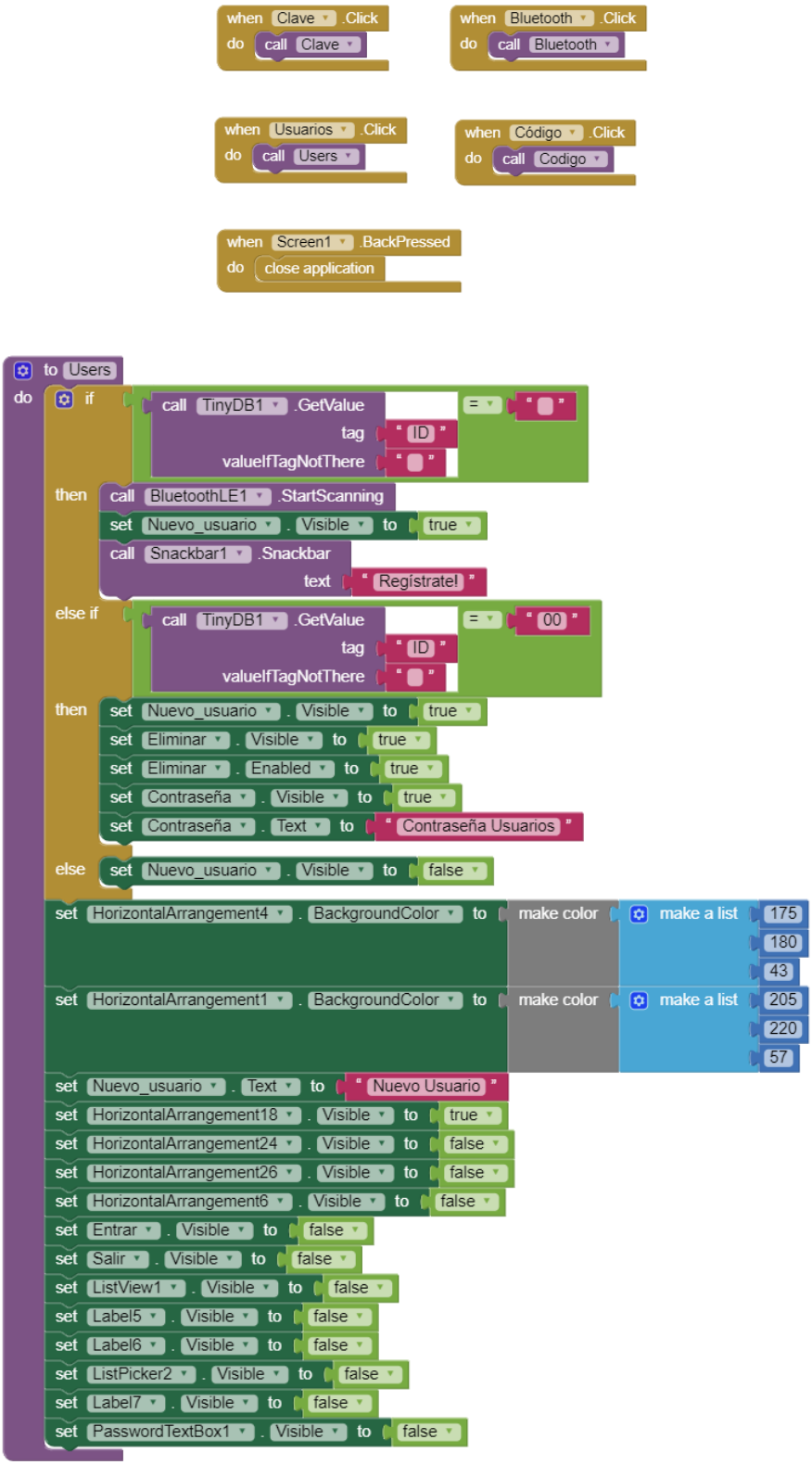

to uno
do
if get global B1 = 1
then set global B1 to call TinyDB1 .GetValue
tag "Red"
valueIfTagNotThere ""
else if get global B1 = 2
then set global B1 to call TinyDB1 .GetValue
tag "Green"
valueIfTagNotThere ""
else if get global B1 = 3
then set global B1 to call TinyDB1 .GetValue
tag "Blue"
valueIfTagNotThere ""
else if get global B1 = 4
then set global B1 to call TinyDB1 .GetValue
tag "Yellow"
valueIfTagNotThere ""


when Button1 .Click
do call Password2
x get global B1


when Button2 .Click
do call Password2
x get global B2

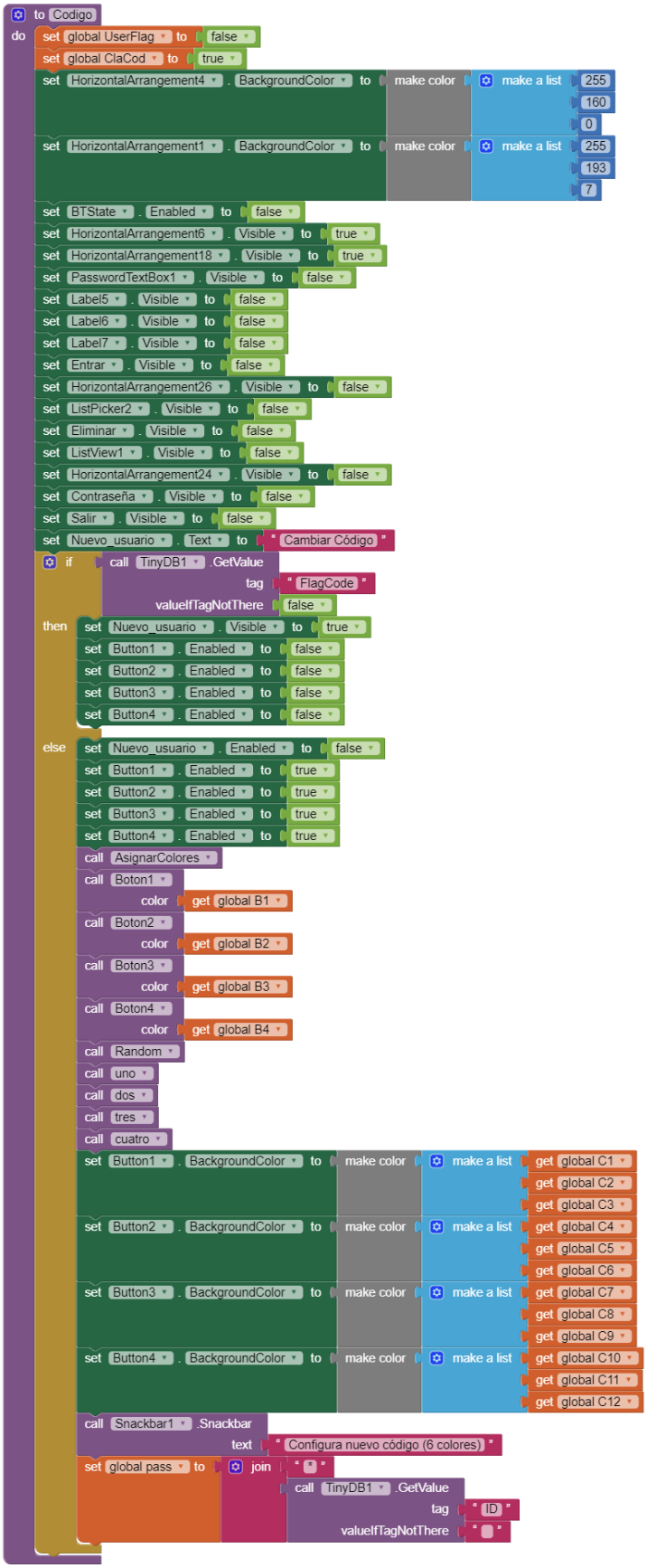

when Button3 .Click
do call Password2
x get global B3

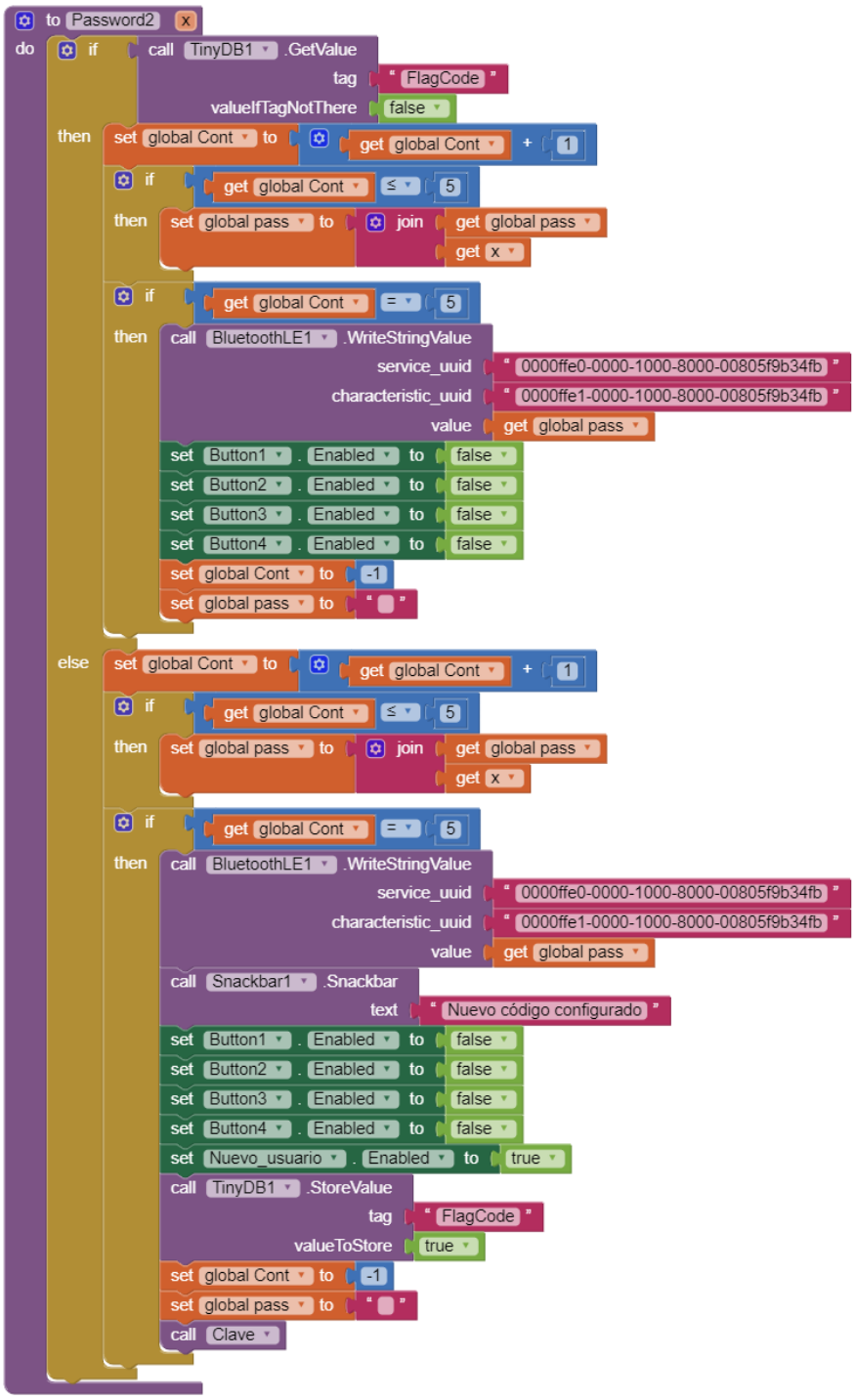

when Button4 .Click
do call Password2
x get global B4


when Clave .Click
do call Clave
when Bluetooth .Click
do call Bluetooth
when Usuarios .Click
do call Users
when Código .Click
do call Codigo
when Screen1 .BackPressed
do close application
to Users
do if call TinyDB1 .GetValue tag "ID" valueIfTagNotThere "" = ""
then call BluetoothLE1 .StartScanning
set Nuevo_usuario . Visible to true
call Snackbar1 .Snackbar text "Regístrate!"
else if call TinyDB1 .GetValue tag "ID" valueIfTagNotThere "" = "00"
then set Nuevo_usuario . Visible to true
set Eliminar . Visible to true
set Eliminar . Enabled to true
set Contraseña . Visible to true
set Contraseña . Text to "Contraseña Usuarios"
else set Nuevo_usuario . Visible to false
set HorizontalArrangement4 . BackgroundColor to make color make a list 175 180 43
set HorizontalArrangement1 . BackgroundColor to make color make a list 205 220 57
set Nuevo_usuario . Text to "Nuevo Usuario"
set HorizontalArrangement18 . Visible to true
set HorizontalArrangement24 . Visible to false
set HorizontalArrangement26 . Visible to false
set HorizontalArrangement6 . Visible to false
set Entrar . Visible to false
set Salir . Visible to false
set ListView1 . Visible to false
set Label5 . Visible to false
set Label6 . Visible to false
set ListPicker2 . Visible to false
set Label7 . Visible to false
set PasswordTextBox1 . Visible to false

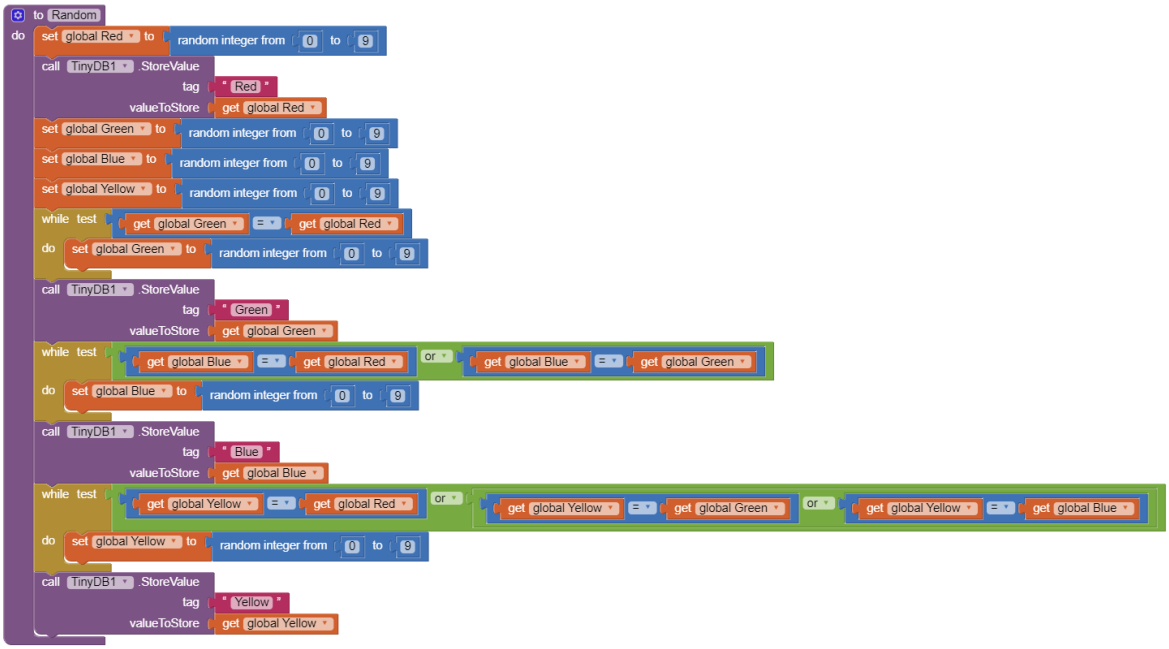

to cuatro
do
if get global B4 = 1
then set global B4 to call TinyDB1 .GetValue tag "Red" valueIfTagNotThere ""
else if get global B4 = 2
then set global B4 to call TinyDB1 .GetValue tag "Green" valueIfTagNotThere ""
else if get global B4 = 3
then set global B4 to call TinyDB1 .GetValue tag "Blue" valueIfTagNotThere ""
else if get global B4 = 4
then set global B4 to call TinyDB1 .GetValue tag "Yellow" valueIfTagNotThere ""

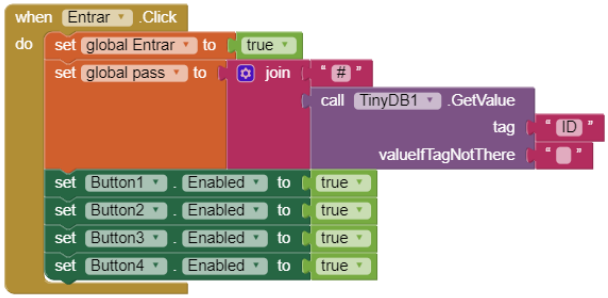

to dos
do
if get global B2 = 1
then set global B2 to call TinyDB1 .GetValue tag "Red" valueIfTagNotThere ""
else if get global B2 = 2
then set global B2 to call TinyDB1 .GetValue tag "Green" valueIfTagNotThere ""
else if get global B2 = 3
then set global B2 to call TinyDB1 .GetValue tag "Blue" valueIfTagNotThere ""
else if get global B2 = 4
then set global B2 to call TinyDB1 .GetValue tag "Yellow" valueIfTagNotThere ""

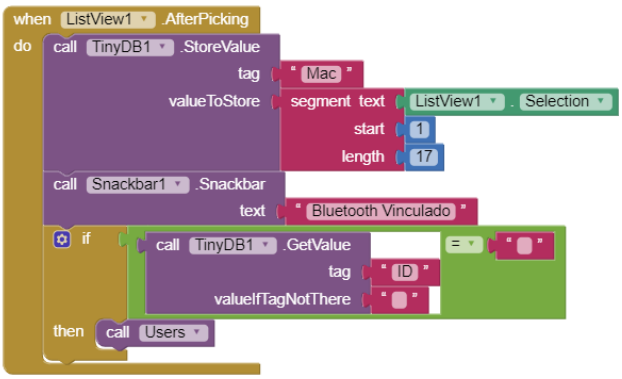

to Codigo
do
set global UserFlag to false
set global ClaCod to true
set HorizontalArrangement4 . BackgroundColor to make color make a list 255 160 0
set HorizontalArrangement1 . BackgroundColor to make color make a list 255 193 7
set BTState . Enabled to false
set HorizontalArrangement6 . Visible to true
set HorizontalArrangement18 . Visible to true
set PasswordTextBox1 . Visible to false
set Label5 . Visible to false
set Label6 . Visible to false
set Label7 . Visible to false
set Entrar . Visible to false
set HorizontalArrangement26 . Visible to false
set ListPicker2 . Visible to false
set Eliminar . Visible to false
set ListView1 . Visible to false
set HorizontalArrangement24 . Visible to false
set Contraseña . Visible to false
set Salir . Visible to false
set Nuevo_usuario . Text to " Cambiar Código "
if call TinyDB1 .GetValue tag " FlagCode " valueIfTagNotThere false
then set Nuevo_usuario . Visible to true
set Button1 . Enabled to false
set Button2 . Enabled to false
set Button3 . Enabled to false
set Button4 . Enabled to false
else set Nuevo_usuario . Enabled to false
set Button1 . Enabled to true
set Button2 . Enabled to true
set Button3 . Enabled to true
set Button4 . Enabled to true
call AsignarColores
call Boton1 color get global B1
call Boton2 color get global B2
call Boton3 color get global B3
call Boton4 color get global B4
call Random
call uno
call dos
call tres
call cuatro
set Button1 . BackgroundColor to make color make a list get global C1 get global C2 get global C3
set Button2 . BackgroundColor to make color make a list get global C4 get global C5 get global C6
set Button3 . BackgroundColor to make color make a list get global C7 get global C8 get global C9
set Button4 . BackgroundColor to make color make a list get global C10 get global C11 get global C12
call Snackbar1 .Snackbar text " Configura nuevo código (6 colores) "
set global pass to join " " call TinyDB1 .GetValue tag " ID " valueIfTagNotThere " "

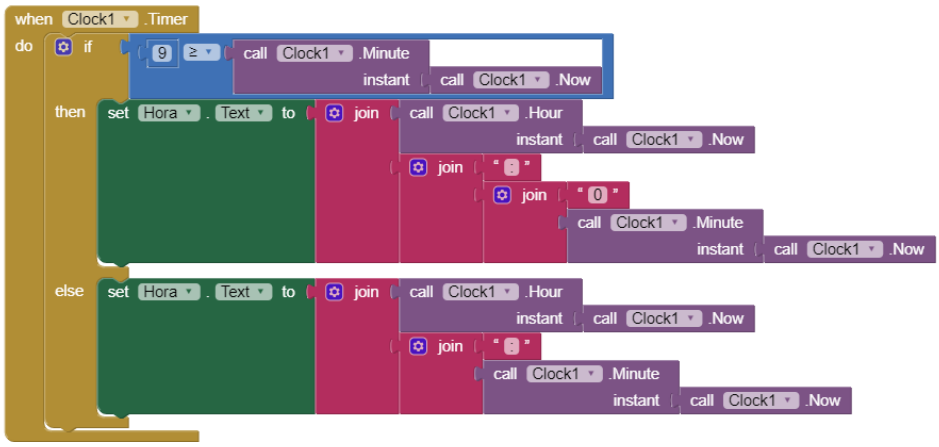

to Password2 x
do if call TinyDB1 .GetValue
tag " FlagCode "
valueIfTagNotThere false
then set global Cont to get global Cont + 1
if get global Cont ≤ 5
then set global pass to join get global pass
get x
if get global Cont = 5
then call BluetoothLE1 .WriteStringValue
service_uuid " 0000ffe0-0000-1000-8000-00805f9b34fb "
characteristic_uuid " 0000ffe1-0000-1000-8000-00805f9b34fb "
value get global pass
set Button1 . Enabled to false
set Button2 . Enabled to false
set Button3 . Enabled to false
set Button4 . Enabled to false
set global Cont to -1
set global pass to " "
else set global Cont to get global Cont + 1
if get global Cont ≤ 5
then set global pass to join get global pass
get x
if get global Cont = 5
then call BluetoothLE1 .WriteStringValue
service_uuid " 0000ffe0-0000-1000-8000-00805f9b34fb "
characteristic_uuid " 0000ffe1-0000-1000-8000-00805f9b34fb "
value get global pass
call Snackbar1 .Snackbar
text " Nuevo código configurado "
set Button1 . Enabled to false
set Button2 . Enabled to false
set Button3 . Enabled to false
set Button4 . Enabled to false
set Nuevo_usuario . Enabled to true
call TinyDB1 .StoreValue
tag " FlagCode "
valueToStore true
set global Cont to -1
set global pass to " "
call Clave


to Boton1 color
do
if get color = 1
then set global C1 to 255
set global C2 to 0
set global C3 to 0
if get color = 2
then set global C1 to 100
set global C2 to 221
set global C3 to 23
if get color = 3
then set global C1 to 83
set global C2 to 109
set global C3 to 254
if get color = 4
then set global C1 to 255
set global C2 to 200
set global C3 to 7

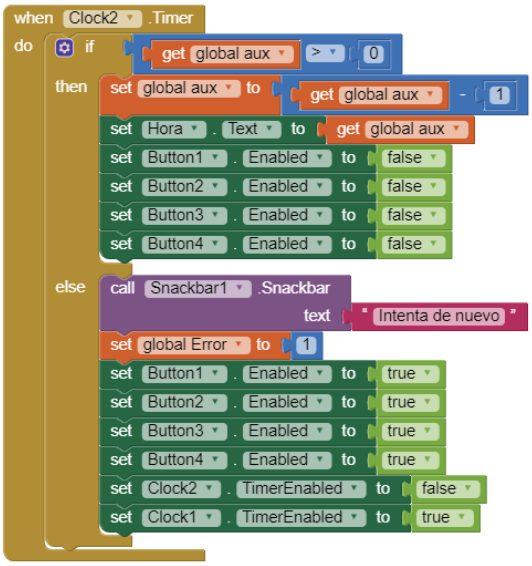

to Random
do
set global Red to random integer from 0 to 9
call TinyDB1 .StoreValue
tag "Red"
valueToStore get global Red
set global Green to random integer from 0 to 9
set global Blue to random integer from 0 to 9
set global Yellow to random integer from 0 to 9
while test get global Green = get global Red
do set global Green to random integer from 0 to 9
call TinyDB1 .StoreValue
tag "Green"
valueToStore get global Green
while test get global Blue = get global Red or get global Blue = get global Green
do set global Blue to random integer from 0 to 9
call TinyDB1 .StoreValue
tag "Blue"
valueToStore get global Blue
while test get global Yellow = get global Red or get global Yellow = get global Green or get global Yellow = get global Blue
do set global Yellow to random integer from 0 to 9
call TinyDB1 .StoreValue
tag "Yellow"
valueToStore get global Yellow

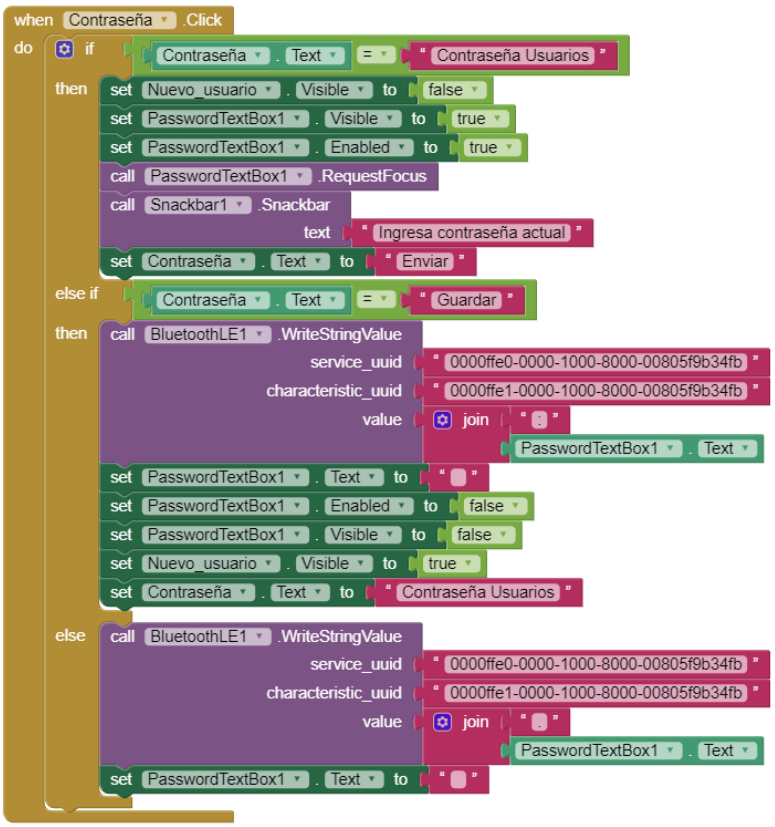

when Entrar .Click
do
set global Entrar to true
set global pass to join "#"
call TinyDB1 .GetValue
tag "ID"
valueIfTagNotThere ""
set Button1 . Enabled to true
set Button2 . Enabled to true
set Button3 . Enabled to true
set Button4 . Enabled to true

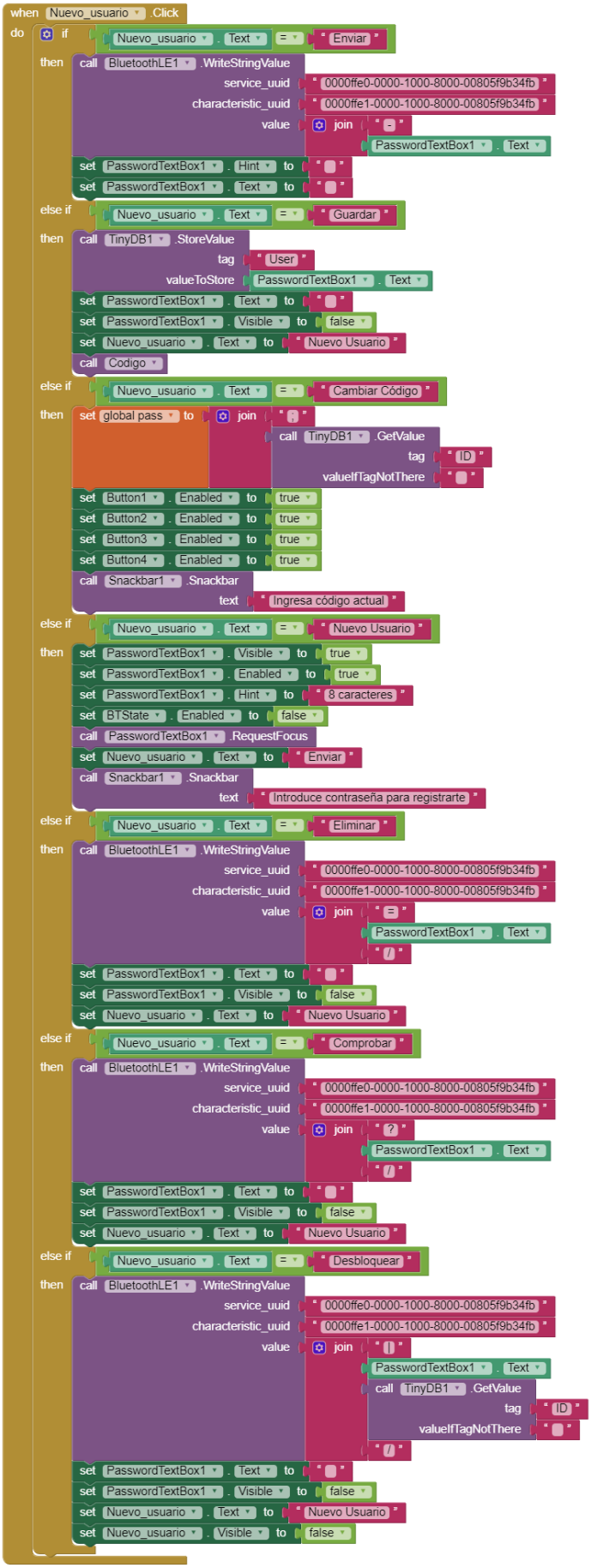

when ListView1 .AfterPicking
do
call TinyDB1 .StoreValue
tag
" Mac "
valueToStore
segment text
ListView1 . Selection
start
1
length
17
call Snackbar1 .Snackbar
text
" Bluetooth Vinculado "
if
call TinyDB1 .GetValue
tag
" ID "
valueIfTagNotThere
" "
=
" "
then
call Users

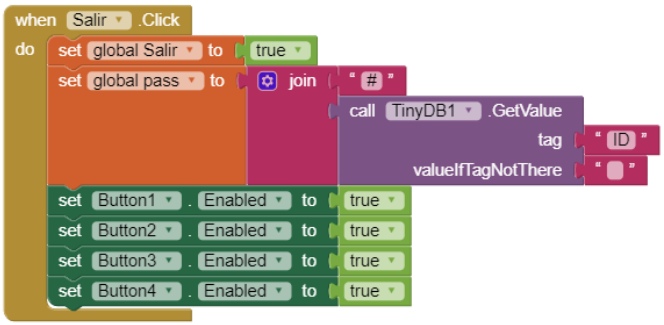

when Clock1 .Timer
do
if
9
≥
call Clock1 .Minute
instant
call Clock1 .Now
then
set Hora . Text to
join
call Clock1 .Hour
instant
call Clock1 .Now
join
" : "
join
" 0 "
call Clock1 .Minute
instant
call Clock1 .Now
else
set Hora . Text to
join
call Clock1 .Hour
instant
call Clock1 .Now
join
" : "
call Clock1 .Minute
instant
call Clock1 .Now

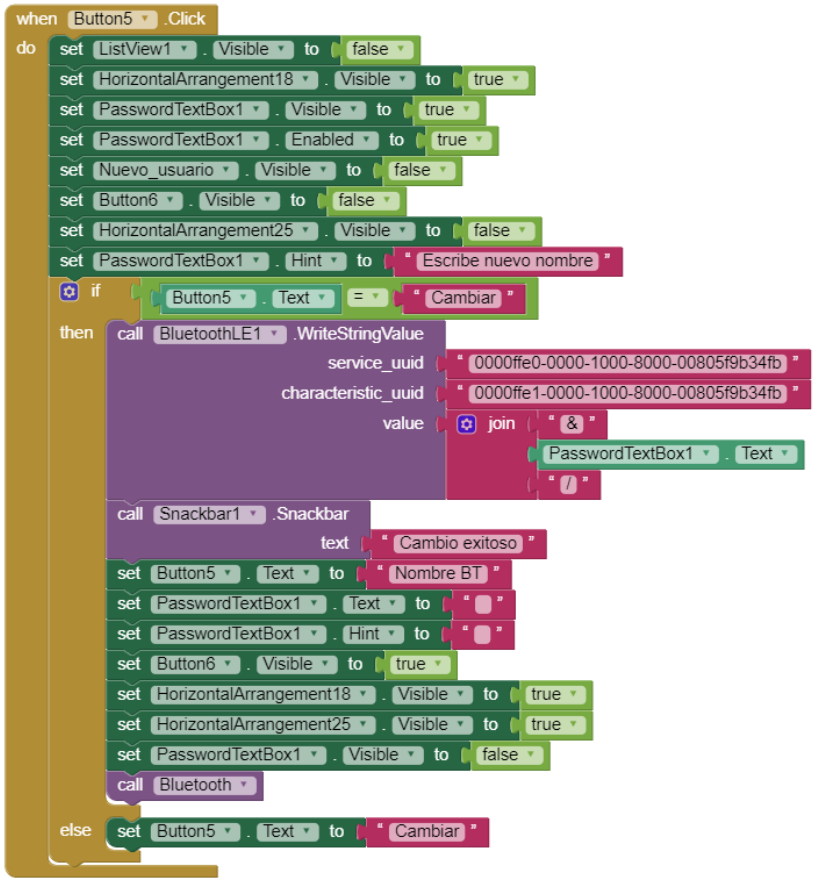

when BTState .Click
do set global Emergency to 1 + get global Emergency
if get global Emergency = 15
then call Users
set Nuevo_usuario . Text to " Comprobar "
call Snackbar1 .Snackbar
text " Escribe tu ID junto con la contraseña de emergen... "
set PasswordTextBox1 . Visible to true
set PasswordTextBox1 . Enabled to true
call PasswordTextBox1 .RequestFocus

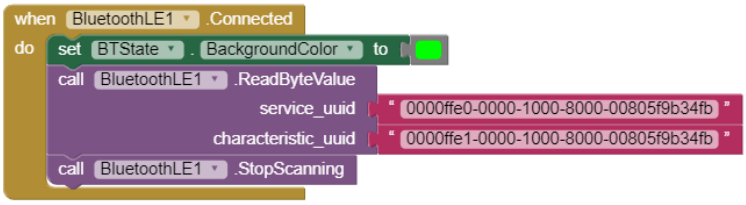

when Clock2 .Timer
do if get global aux > 0
then set global aux to get global aux - 1
set Hora . Text to get global aux
set Button1 . Enabled to false
set Button2 . Enabled to false
set Button3 . Enabled to false
set Button4 . Enabled to false
else call Snackbar1 .Snackbar
text " Intenta de nuevo "
set global Error to 1
set Button1 . Enabled to true
set Button2 . Enabled to true
set Button3 . Enabled to true
set Button4 . Enabled to true
set Clock2 . TimerEnabled to false
set Clock1 . TimerEnabled to true

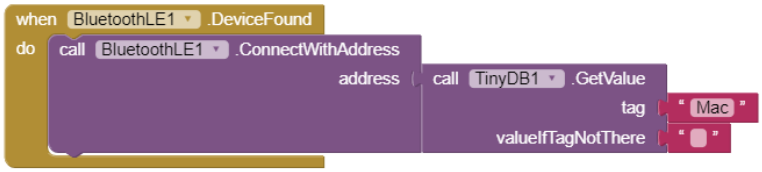

when Contraseña .Click
do if Contraseña . Text = " Contraseña Usuarios "
then set Nuevo_usuario . Visible to false
set PasswordTextBox1 . Visible to true
set PasswordTextBox1 . Enabled to true
call PasswordTextBox1 .RequestFocus
call Snackbar1 .Snackbar
text " Ingresa contraseña actual "
set Contraseña . Text to " Enviar "
else if Contraseña . Text = " Guardar "
then call BluetoothLE1 .WriteStringValue
service_uuid " 0000ffe0-0000-1000-8000-00805f9b34fb "
characteristic_uuid " 0000ffe1-0000-1000-8000-00805f9b34fb "
value join " . "
PasswordTextBox1 . Text
set PasswordTextBox1 . Text to " "
set PasswordTextBox1 . Enabled to false
set PasswordTextBox1 . Visible to false
set Nuevo_usuario . Visible to true
set Contraseña . Text to " Contraseña Usuarios "
else call BluetoothLE1 .WriteStringValue
service_uuid " 0000ffe0-0000-1000-8000-00805f9b34fb "
characteristic_uuid " 0000ffe1-0000-1000-8000-00805f9b34fb "
value join " . "
PasswordTextBox1 . Text
set PasswordTextBox1 . Text to " "

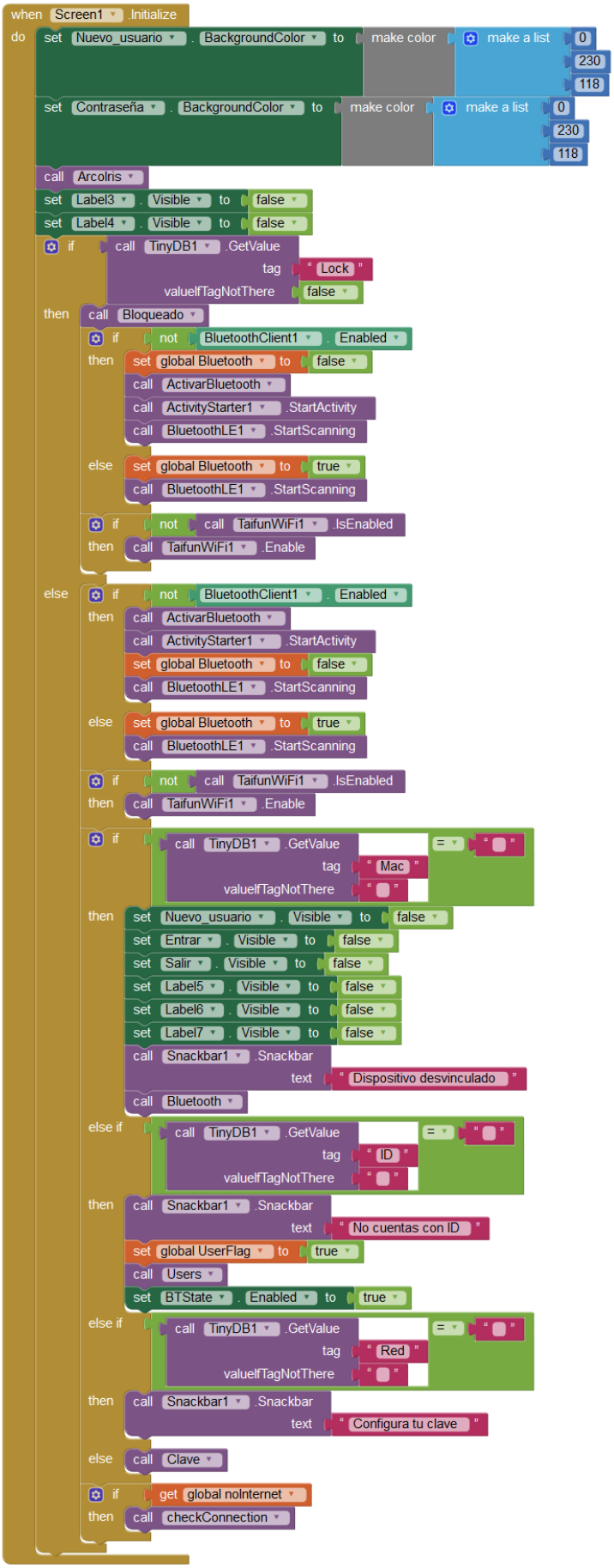

when Nuevo_usuario .Click
do if Nuevo_usuario . Text = " Enviar "
then call BluetoothLE1 .WriteStringValue
service_uuid " 0000ffe0-0000-1000-8000-00805f9b34fb "
characteristic_uuid " 0000ffe1-0000-1000-8000-00805f9b34fb "
value join " - "
PasswordTextBox1 . Text
set PasswordTextBox1 . Hint to " "
set PasswordTextBox1 . Text to " "
else if Nuevo_usuario . Text = " Guardar "
then call TinyDB1 .StoreValue
tag " User "
valueToStore PasswordTextBox1 . Text
set PasswordTextBox1 . Text to " "
set PasswordTextBox1 . Visible to false
set Nuevo_usuario . Text to " Nuevo Usuario "
call Codigo
else if Nuevo_usuario . Text = " Cambiar Código "
then set global pass to join " "
call TinyDB1 .GetValue
tag " ID "
valueIfTagNotThere " "
set Button1 . Enabled to true
set Button2 . Enabled to true
set Button3 . Enabled to true
set Button4 . Enabled to true
call Snackbar1 .Snackbar
text " Ingresa código actual "
else if Nuevo_usuario . Text = " Nuevo Usuario "
then set PasswordTextBox1 . Visible to true
set PasswordTextBox1 . Enabled to true
set PasswordTextBox1 . Hint to " 8 caracteres "
set BTState . Enabled to false
call PasswordTextBox1 .RequestFocus
set Nuevo_usuario . Text to " Enviar "
call Snackbar1 .Snackbar
text " Introduce contraseña para registrarte "
else if Nuevo_usuario . Text = " Eliminar "
then call BluetoothLE1 .WriteStringValue
service_uuid " 0000ffe0-0000-1000-8000-00805f9b34fb "
characteristic_uuid " 0000ffe1-0000-1000-8000-00805f9b34fb "
value join " = "
PasswordTextBox1 . Text
" / "
set PasswordTextBox1 . Text to " "
set PasswordTextBox1 . Visible to false
set Nuevo_usuario . Text to " Nuevo Usuario "
else if Nuevo_usuario . Text = " Comprobar "
then call BluetoothLE1 .WriteStringValue
service_uuid " 0000ffe0-0000-1000-8000-00805f9b34fb "
characteristic_uuid " 0000ffe1-0000-1000-8000-00805f9b34fb "
value join " ? "
PasswordTextBox1 . Text
" / "
set PasswordTextBox1 . Text to " "
set PasswordTextBox1 . Visible to false
set Nuevo_usuario . Text to " Nuevo Usuario "
else if Nuevo_usuario . Text = " Desbloquear "
then call BluetoothLE1 .WriteStringValue
service_uuid " 0000ffe0-0000-1000-8000-00805f9b34fb "
characteristic_uuid " 0000ffe1-0000-1000-8000-00805f9b34fb "
value join " ! "
PasswordTextBox1 . Text
call TinyDB1 .GetValue
tag " ID "
valueIfTagNotThere " "
" / "
set PasswordTextBox1 . Text to " "
set PasswordTextBox1 . Visible to false
set Nuevo_usuario . Text to " Nuevo Usuario "
set Nuevo_usuario . Visible to false

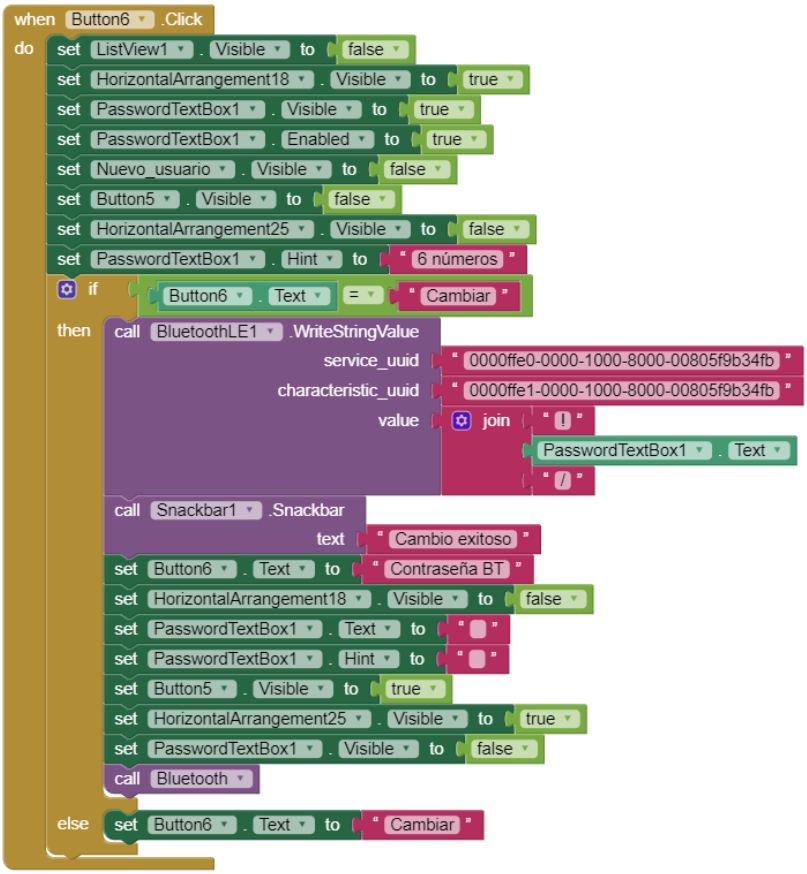

when Salir .Click
do set global Salir to true
set global pass to join "#"
call TinyDB1 .GetValue
tag "ID"
valueIfTagNotThere ""
set Button1 . Enabled to true
set Button2 . Enabled to true
set Button3 . Enabled to true
set Button4 . Enabled to true

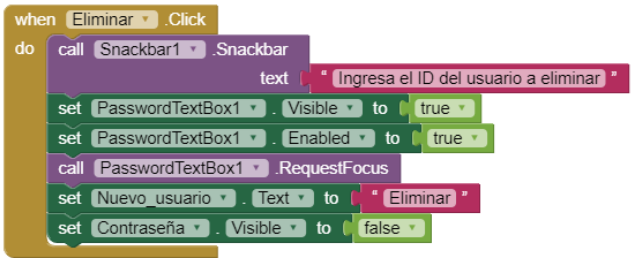

when Button5 .Click
do set ListView1 . Visible to false
set HorizontalArrangement18 . Visible to true
set PasswordTextBox1 . Visible to true
set PasswordTextBox1 . Enabled to true
set Nuevo_usuario . Visible to false
set Button6 . Visible to false
set HorizontalArrangement25 . Visible to false
set PasswordTextBox1 . Hint to "Escribe nuevo nombre"
if Button5 . Text = "Cambiar"
then call BluetoothLE1 .WriteStringValue
service_uuid "0000ffe0-0000-1000-8000-00805f9b34fb"
characteristic_uuid "0000ffe1-0000-1000-8000-00805f9b34fb"
value join "&"
PasswordTextBox1 . Text
"/"
call Snackbar1 .Snackbar
text "Cambio exitoso"
set Button5 . Text to "Nombre BT"
set PasswordTextBox1 . Text to ""
set PasswordTextBox1 . Hint to ""
set Button6 . Visible to true
set HorizontalArrangement18 . Visible to true
set HorizontalArrangement25 . Visible to true
set PasswordTextBox1 . Visible to false
call Bluetooth
else set Button5 . Text to "Cambiar"

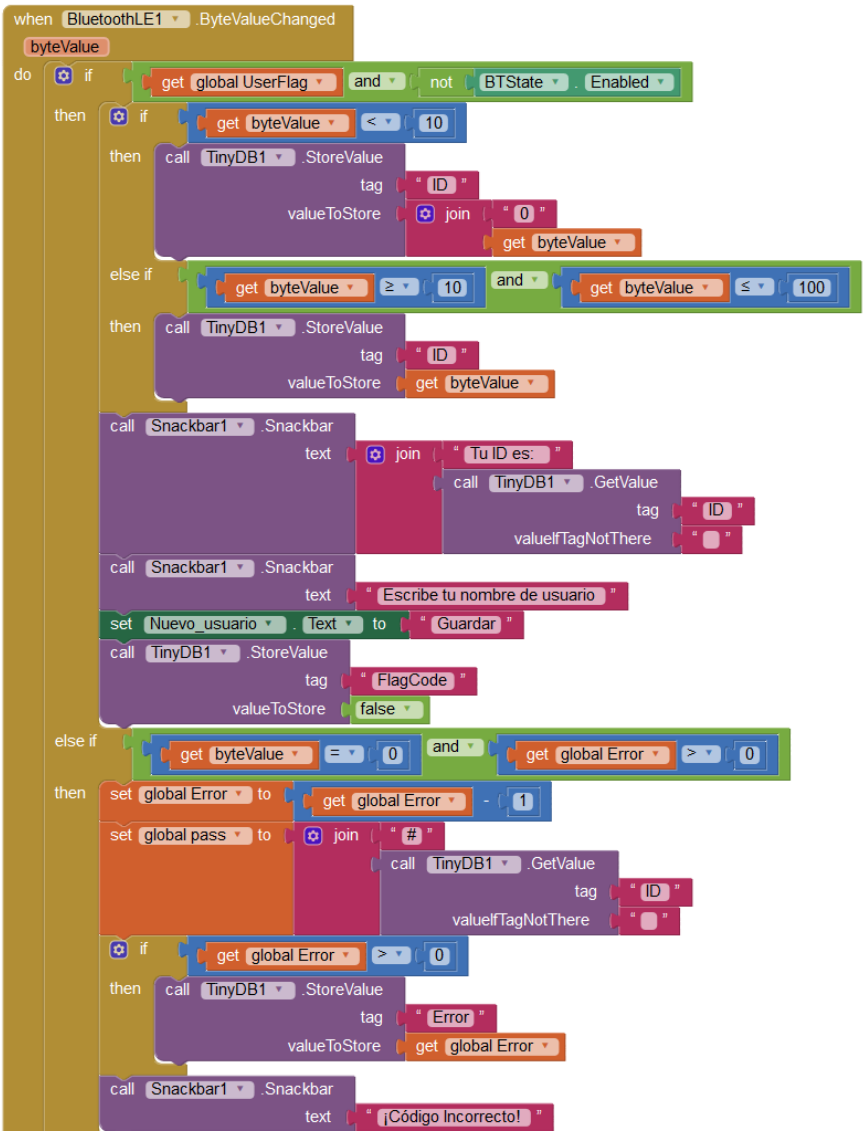

when BluetoothLE1 .Connected
do set BTState . BackgroundColor to
call BluetoothLE1 .ReadByteValue
service_uuid "0000ffe0-0000-1000-8000-00805f9b34fb"
characteristic_uuid "0000ffe1-0000-1000-8000-00805f9b34fb"
call BluetoothLE1 .StopScanning

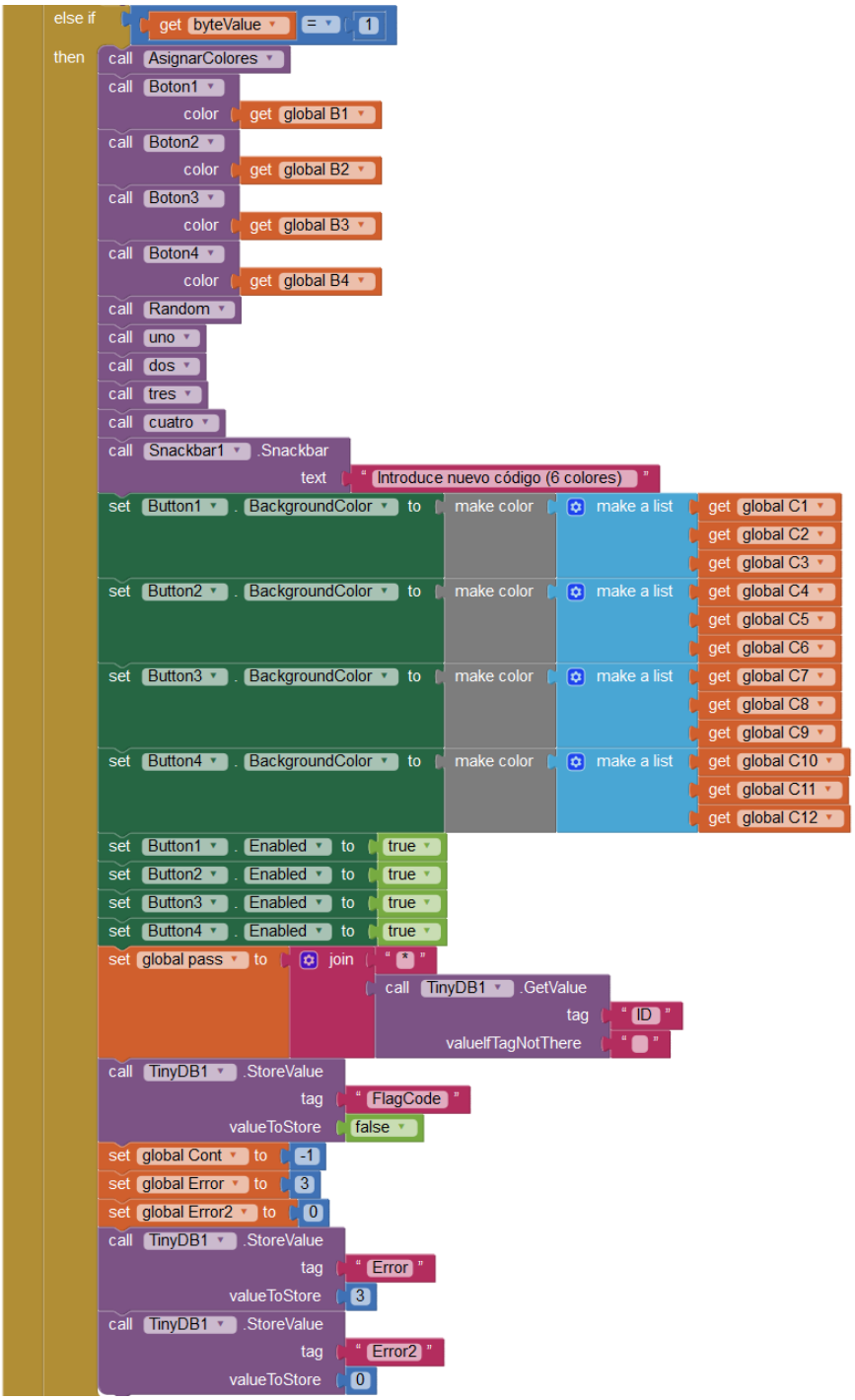

when BluetoothLE1 .DeviceFound
do call BluetoothLE1 .ConnectWithAddress
address call TinyDB1 .GetValue
tag "Mac"
valueIfTagNotThere ""

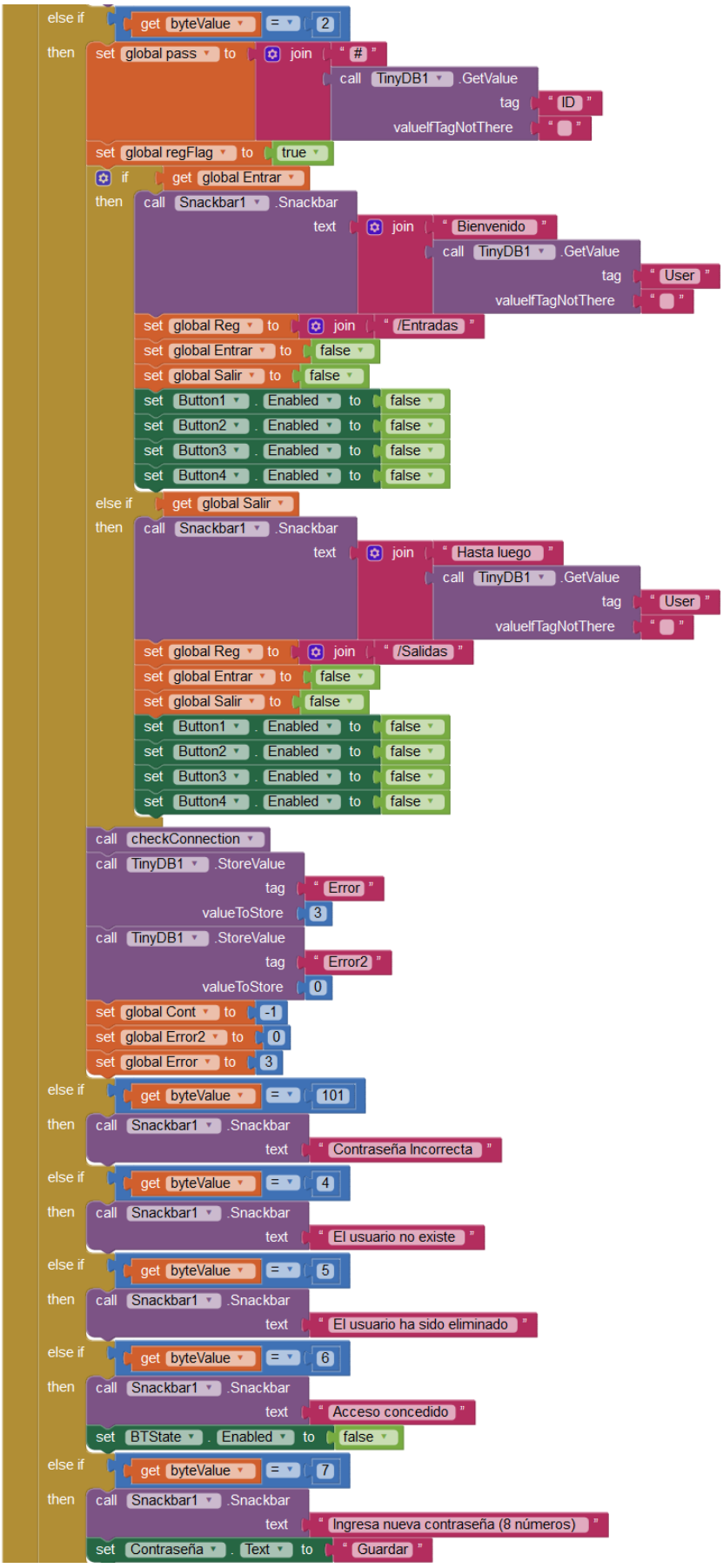

when Screen1 .Initialize
do
set Nuevo_usuario . BackgroundColor to make color make a list 0 230 118
set Contraseña . BackgroundColor to make color make a list 0 230 118
call Arcolris
set Label3 . Visible to false
set Label4 . Visible to false
if call TinyDB1 .GetValue tag " Lock " valueIfTagNotThere false
then call Bloqueado
if not BluetoothClient1 . Enabled
then set global Bluetooth to false
call ActivarBluetooth
call ActivityStarter1 .StartActivity
call BluetoothLE1 .StartScanning
else set global Bluetooth to true
call BluetoothLE1 .StartScanning
if not call TaifunWiFi1 .IsEnabled
then call TaifunWiFi1 .Enable
else if not BluetoothClient1 . Enabled
then call ActivarBluetooth
call ActivityStarter1 .StartActivity
set global Bluetooth to false
call BluetoothLE1 .StartScanning
else set global Bluetooth to true
call BluetoothLE1 .StartScanning
if not call TaifunWiFi1 .IsEnabled
then call TaifunWiFi1 .Enable
if call TinyDB1 .GetValue tag " Mac " valueIfTagNotThere " " = " "
then set Nuevo_usuario . Visible to false
set Entrar . Visible to false
set Salir . Visible to false
set Label5 . Visible to false
set Label6 . Visible to false
set Label7 . Visible to false
call Snackbar1 .Snackbar text " Dispositivo desvinculado "
call Bluetooth
else if call TinyDB1 .GetValue tag " ID " valueIfTagNotThere " " = " "
then call Snackbar1 .Snackbar text " No cuentas con ID "
set global UserFlag to true
call Users
set BTState . Enabled to true
else if call TinyDB1 .GetValue tag " Red " valueIfTagNotThere " " = " "
then call Snackbar1 .Snackbar text " Configura tu clave "
else call Clave
if get global noInternet
then call checkConnection

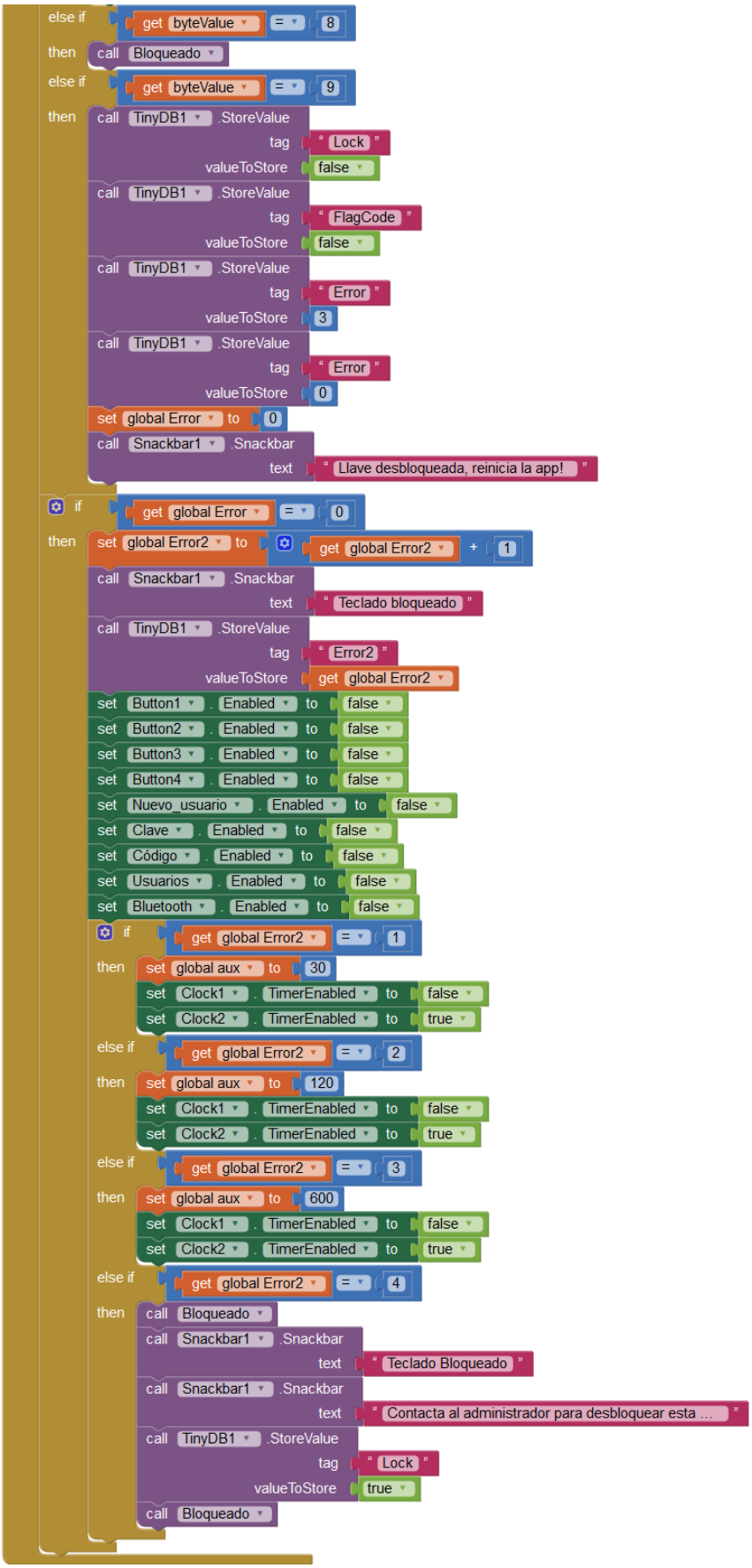

when Button6 .Click
do
set ListView1 . Visible to false
set HorizontalArrangement18 . Visible to true
set PasswordTextBox1 . Visible to true
set PasswordTextBox1 . Enabled to true
set Nuevo_usuario . Visible to false
set Button5 . Visible to false
set HorizontalArrangement25 . Visible to false
set PasswordTextBox1 . Hint to " 6 números "
if Button6 . Text = " Cambiar "
then
call BluetoothLE1 .WriteStringValue
service_uuid " 0000ffe0-0000-1000-8000-00805f9b34fb "
characteristic_uuid " 0000ffe1-0000-1000-8000-00805f9b34fb "
value join " ! "
PasswordTextBox1 . Text
" / "
call Snackbar1 .Snackbar
text " Cambio exitoso "
set Button6 . Text to " Contraseña BT "
set HorizontalArrangement18 . Visible to false
set PasswordTextBox1 . Text to " "
set PasswordTextBox1 . Hint to " "
set Button5 . Visible to true
set HorizontalArrangement25 . Visible to true
set PasswordTextBox1 . Visible to false
call Bluetooth
else
set Button6 . Text to " Cambiar "

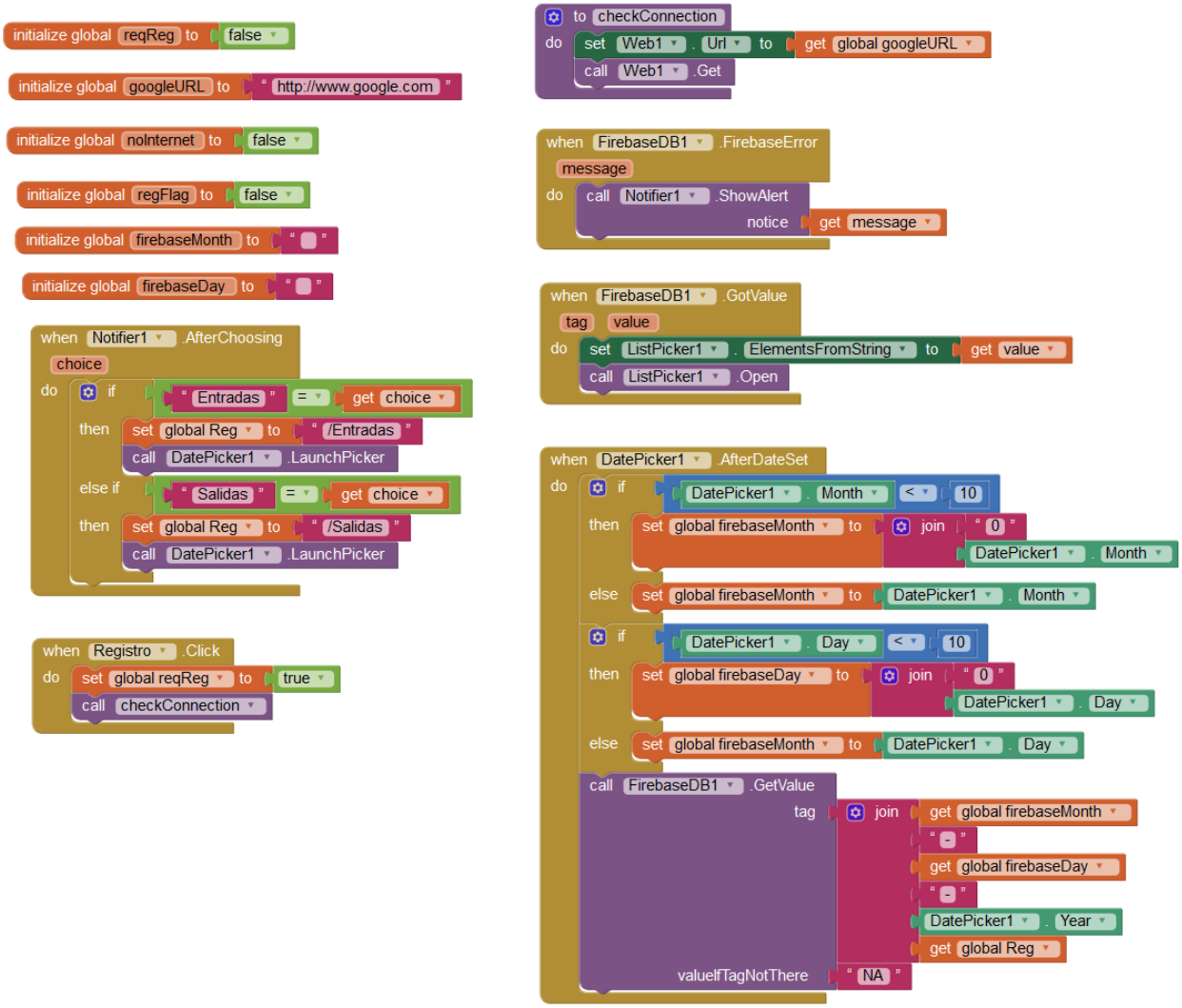

when Eliminar .Click
do
call Snackbar1 .Snackbar
text " Ingresa el ID del usuario a eliminar "
set PasswordTextBox1 . Visible to true
set PasswordTextBox1 . Enabled to true
call PasswordTextBox1 .RequestFocus
set Nuevo_usuario . Text to " Eliminar "
set Contraseña . Visible to false

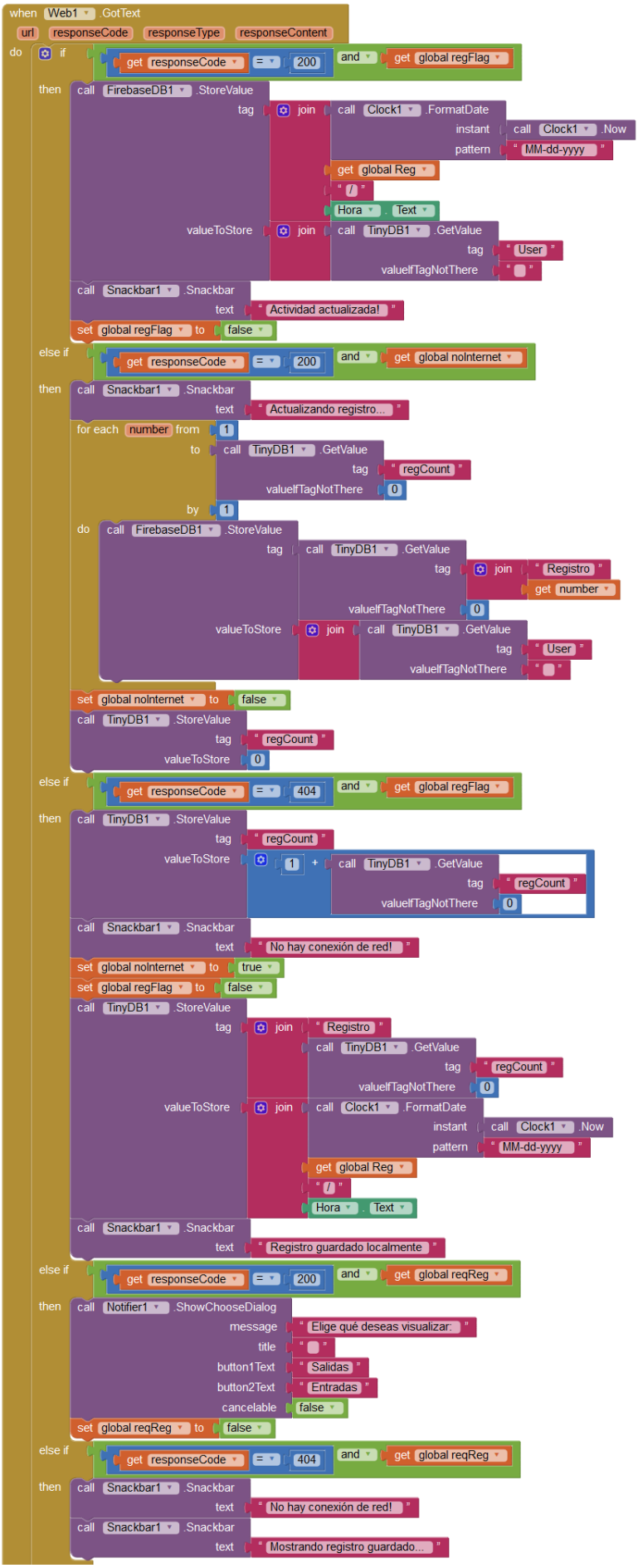

when BluetoothLE1 .ByteValueChanged
byteValue
do if get global UserFlag and not BTState . Enabled
then if get byteValue < 10
then call TinyDB1 .StoreValue
tag "ID"
valueToStore join "0"
get byteValue
else if get byteValue ≥ 10 and get byteValue ≤ 100
then call TinyDB1 .StoreValue
tag "ID"
valueToStore get byteValue
call Snackbar1 .Snackbar
text join "Tu ID es:"
call TinyDB1 .GetValue
tag "ID"
valueIfTagNotThere ""
call Snackbar1 .Snackbar
text "Escribe tu nombre de usuario"
set Nuevo_usuario . Text to "Guardar"
call TinyDB1 .StoreValue
tag "FlagCode"
valueToStore false
else if get byteValue = 0 and get global Error > 0
then set global Error to get global Error - 1
set global pass to join "#"
call TinyDB1 .GetValue
tag "ID"
valueIfTagNotThere ""
if get global Error > 0
then call TinyDB1 .StoreValue
tag "Error"
valueToStore get global Error
call Snackbar1 .Snackbar
text "¡Código Incorrecto!"


else if
get byteValue = 1
then
call AsignarColores
call Boton1
color get global B1
call Boton2
color get global B2
call Boton3
color get global B3
call Boton4
color get global B4
call Random
call uno
call dos
call tres
call cuatro
call Snackbar1 .Snackbar
text "Introduce nuevo código (6 colores)"
set Button1 . BackgroundColor to make color make a list get global C1 get global C2 get global C3
set Button2 . BackgroundColor to make color make a list get global C4 get global C5 get global C6
set Button3 . BackgroundColor to make color make a list get global C7 get global C8 get global C9
set Button4 . BackgroundColor to make color make a list get global C10 get global C11 get global C12
set Button1 . Enabled to true
set Button2 . Enabled to true
set Button3 . Enabled to true
set Button4 . Enabled to true
set global pass to join "*" call TinyDB1 .GetValue tag "ID" valueIfTagNotThere ""
call TinyDB1 .StoreValue tag "FlagCode" valueToStore false
set global Cont to -1
set global Error to 3
set global Error2 to 0
call TinyDB1 .StoreValue tag "Error" valueToStore 3
call TinyDB1 .StoreValue tag "Error2" valueToStore 0


else if
get byteValue = 2
then
set global pass to join " # "
call TinyDB1 .GetValue
tag " ID "
valueIfTagNotThere " "
set global regFlag to true
if get global Entrar
then
call Snackbar1 .Snackbar
text join " Bienvenido "
call TinyDB1 .GetValue
tag " User "
valueIfTagNotThere " "
set global Reg to join " /Entradas "
set global Entrar to false
set global Salir to false
set Button1 . Enabled to false
set Button2 . Enabled to false
set Button3 . Enabled to false
set Button4 . Enabled to false
else if get global Salir
then
call Snackbar1 .Snackbar
text join " Hasta luego "
call TinyDB1 .GetValue
tag " User "
valueIfTagNotThere " "
set global Reg to join " /Salidas "
set global Entrar to false
set global Salir to false
set Button1 . Enabled to false
set Button2 . Enabled to false
set Button3 . Enabled to false
set Button4 . Enabled to false
call checkConnection
call TinyDB1 .StoreValue
tag " Error "
valueToStore 3
call TinyDB1 .StoreValue
tag " Error2 "
valueToStore 0
set global Cont to -1
set global Error2 to 0
set global Error to 3
else if get byteValue = 101
then
call Snackbar1 .Snackbar
text " Contraseña Incorrecta "
else if get byteValue = 4
then
call Snackbar1 .Snackbar
text " El usuario no existe "
else if get byteValue = 5
then
call Snackbar1 .Snackbar
text " El usuario ha sido eliminado "
else if get byteValue = 6
then
call Snackbar1 .Snackbar
text " Acceso concedido "
set BTState . Enabled to false
else if get byteValue = 7
then
call Snackbar1 .Snackbar
text " Ingresa nueva contraseña (8 números) "
set Contraseña . Text to " Guardar "


else if get byteValue = 8
then call Bloqueado
else if get byteValue = 9
then call TinyDB1 .StoreValue tag "Lock" valueToStore false
call TinyDB1 .StoreValue tag "FlagCode" valueToStore false
call TinyDB1 .StoreValue tag "Error" valueToStore 3
call TinyDB1 .StoreValue tag "Error" valueToStore 0
set global Error to 0
call Snackbar1 .Snackbar text "Llave desbloqueada, reinicia la app!"
if get global Error = 0
then set global Error2 to get global Error2 + 1
call Snackbar1 .Snackbar text "Teclado bloqueado"
call TinyDB1 .StoreValue tag "Error2" valueToStore get global Error2
set Button1 . Enabled to false
set Button2 . Enabled to false
set Button3 . Enabled to false
set Button4 . Enabled to false
set Nuevo_usuario . Enabled to false
set Clave . Enabled to false
set Código . Enabled to false
set Usuarios . Enabled to false
set Bluetooth . Enabled to false
if get global Error2 = 1
then set global aux to 30
set Clock1 . TimerEnabled to false
set Clock2 . TimerEnabled to true
else if get global Error2 = 2
then set global aux to 120
set Clock1 . TimerEnabled to false
set Clock2 . TimerEnabled to true
else if get global Error2 = 3
then set global aux to 600
set Clock1 . TimerEnabled to false
set Clock2 . TimerEnabled to true
else if get global Error2 = 4
then call Bloqueado
call Snackbar1 .Snackbar text "Teclado Bloqueado"
call Snackbar1 .Snackbar text "Contacta al administrador para desbloquear esta ..."
call TinyDB1 .StoreValue tag "Lock" valueToStore true
call Bloqueado


initialize global reqReg to false
initialize global googleURL to " http://www.google.com "
initialize global noInternet to false
initialize global regFlag to false
initialize global firebaseMonth to " "
initialize global firebaseDay to " "
when Notifier1 .AfterChoosing
choice
do if " Entradas " = get choice
then set global Reg to " /Entradas "
call DatePicker1 .LaunchPicker
else if " Salidas " = get choice
then set global Reg to " /Salidas "
call DatePicker1 .LaunchPicker
when Registro .Click
do set global reqReg to true
call checkConnection
to checkConnection
do set Web1 . Url to get global googleURL
call Web1 .Get
when FirebaseDB1 .FirebaseError
message
do call Notifier1 .ShowAlert
notice get message
when FirebaseDB1 .GotValue
tag value
do set ListPicker1 . ElementsFromString to get value
call ListPicker1 .Open
when DatePicker1 .AfterDateSet
do if DatePicker1 . Month < 10
then set global firebaseMonth to join " 0 "
DatePicker1 . Month
else set global firebaseMonth to DatePicker1 . Month
if DatePicker1 . Day < 10
then set global firebaseDay to join " 0 "
DatePicker1 . Day
else set global firebaseMonth to DatePicker1 . Day
call FirebaseDB1 .GetValue
tag join get global firebaseMonth
" - "
get global firebaseDay
" - "
DatePicker1 . Year
get global Reg
valueIfTagNotThere " NA "


when Web1 .GotText
url responseCode responseType responseContent
do if get responseCode = 200 and get global regFlag
then call FirebaseDB1 .StoreValue
tag join call Clock1 .FormatDate
instant call Clock1 .Now
pattern "MM-dd-yyyy"
get global Reg
"/"
Hora . Text
valueToStore join call TinyDB1 .GetValue
tag "User"
valueIfTagNotThere ""
call Snackbar1 .Snackbar
text "Actividad actualizada!"
set global regFlag to false
else if get responseCode = 200 and get global noInternet
then call Snackbar1 .Snackbar
text "Actualizando registro..."
for each number from 1
to call TinyDB1 .GetValue
tag "regCount"
valueIfTagNotThere 0
by 1
do call FirebaseDB1 .StoreValue
tag call TinyDB1 .GetValue
tag join "Registro"
get number
valueIfTagNotThere 0
valueToStore join call TinyDB1 .GetValue
tag "User"
valueIfTagNotThere ""
set global noInternet to false
call TinyDB1 .StoreValue
tag "regCount"
valueToStore 0
else if get responseCode = 404 and get global regFlag
then call TinyDB1 .StoreValue
tag "regCount"
valueToStore 1 + call TinyDB1 .GetValue
tag "regCount"
valueIfTagNotThere 0
call Snackbar1 .Snackbar
text "No hay conexión de red!"
set global noInternet to true
set global regFlag to false
call TinyDB1 .StoreValue
tag join "Registro"
call TinyDB1 .GetValue
tag "regCount"
valueIfTagNotThere 0
valueToStore join call Clock1 .FormatDate
instant call Clock1 .Now
pattern "MM-dd-yyyy"
get global Reg
"/"
Hora . Text
call Snackbar1 .Snackbar
text "Registro guardado localmente"
else if get responseCode = 200 and get global reqReg
then call Notifier1 .ShowChooseDialog
message "Elige qué deseas visualizar:"
title ""
button1Text "Salidas"
button2Text "Entradas"
cancelable false
set global reqReg to false
else if get responseCode = 404 and get global reqReg
then call Snackbar1 .Snackbar
text "No hay conexión de red!"
call Snackbar1 .Snackbar
text "Mostrando registro guardado..."

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE GRÁFICAS

BIBLIOGRAFÍA

[1] C. Franklin and J. Layton, "howstuffworks," [Online]. Available: https://electronics.howstuffworks.com/bluetooth.htm. [Accessed 12 Junio 2017].

[2] Jimb0, «sparkfun,» [En línea]. Available: https://learn.sparkfun.com/tutorials/serial-communication/serial-intro. [Último acceso: 4 Diciembre 2017].

[3] «Android Open Source Project,» [En línea]. Available: https://source.android.com/setup/. [Último acceso: 8 Junio 2017].

[4] «open handset alliance,» [En línea]. Available: https://www.openhandsetalliance.com/android_overview.html. [Último acceso: 8 Junio 2017].

[5] «Mc Graw Hill Education,» [En línea]. Available: https://www.mheducation.es/bcv/guide/capitulo/8448147227.pdf. [Último acceso: 12 Junio 2017].

[6] M. I. o. Technology, «MIT App Inventor,» [En línea]. Available: http://appinventor.mit.edu/explore/about-us.html. [Último acceso: 4 Enero 2018].

[7] T. H. Depot, «The Home Depot,» [En línea]. Available: http://www.homedepot.com.mx/ferreteria/cerraduras. [Último acceso: 6 Junio 2017].

[8] Samsung, «Samsung,» [En línea]. Available: https://www.samsungdigitallife.com. [Último acceso: 6 Junio 2017].

[9] Kickstarter, «Kickstarter,» [En línea]. Available: https://www.kickstarter.com/projects/candyhouse/sesame-your-key-reinvented?ref=nav_search&result=project&term=sesame. [Último acceso: 6 Junio 2017].

[10] A. Corporation, «Microchip,» 11 2016. [En línea]. Available: http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-42735-8-bit-AVR-Microcontroller-ATmega328-328P_Datasheet.pdf. [Último acceso: 25 Junio 2018].

[11] J.-M. Chung, «Coursera,» [En línea]. Available: https://www.coursera.org/lecture/wireless-communication-technologies/bluetooth-xbvzP. [Último acceso: 8 Junio 2017].

[12] T. Instruments, «Texas Instruments,» Junio 2013. [En línea]. Available: http://www.ti.com/lit/ds/symlink/cc2541.pdf. [Último acceso: 14 Junio 2017].

[13] T. Instruments, «Texas Instruments,» Enero 2016. [En línea]. Available: http://www.ti.com/lit/ds/symlink/l293.pdf. [Último acceso: 8 Mayo 2018].

[14] M. Inc., «Microchip Technology,» 2017. [En línea]. Available: http://ww1.microchip.com/downloads/en/AppNotes/00002447A.pdf. [Último acceso: 22 Junio 2018].

[15] A. Corporation, «Microchip Technology,» Abril 2015. [En línea]. Available: https://www.microchip.com/webdoc/AVRLibcReferenceManual/index.html. [Último acceso: 14 Marzo 2018].

[16] R. Irani, «Romin Irani's Blog,» 9 Septiembre 2016. [En línea]. Available: https://rominirani.com/tutorial-mit-app-inventor-firebase-4be95051c325. [Último acceso: 22 12 2017].

[17] D. Jahshan, P. Hutchinson, F. Tappero, C. Jarron y M. Van den Berg, «KiCad EDA,» 24 Agosto 2017. [En línea]. Available: http://docs.kicad-pcb.org/stable/es/getting_started_in_kicad.pdf. [Último acceso: 12 Mayo 2018].