INSTITUTO SUPERIOR TECNOLÓGICO

SUDAMERICANO

QUITO - ECUADOR

ESCUELA DE

SISTEMAS DE AUTOMATIZACIÓN

PROYECTO DE TITULACIÓN

TEMA:

Diseño e implementación de un sistema prototipo de casilleros inteligentes con apertura por aplicación móvil, desarrollado para Android, vía Bluetooth, con registro de ingreso (CIV)

AUTORES: GÁLVEZ SÁNCHEZ DIEGO JOSUÉ
DE LA TORRE GAIBOR JUAN CARLOS

TUTOR: MSc. VILLASÍS FABRIZIO

San Francisco de Quito, noviembre del 2020

AUTORÍA

Nosotros, Juan Carlos de la Torre Gaibor, portador de la cédula de ciudadanía No.172632440-1, y Diego Josué Gálvez Sánchez, portador de la cédula de ciudadanía No.172041299-6, declaramos bajo juramento que el trabajo aquí descrito, es de nuestra autoría; que no ha sido previamente presentado para ningún grado, o calificación profesional y que hemos consultado e investigado en base a las referencias bibliográficas que se incluyen en este documento. Esta investigación no contiene plagio alguno y es resultado de un trabajo serio desarrollado en su totalidad por nuestra persona.

Juan Carlos De la Torre Gaibor

Diego Josué Gálvez Sánchez

CERTIFICACIÓN

Una vez que se ha culminado la elaboración del proyecto de titulación, cuyo tema es: “Diseño e implementación de un sistema prototipo de casilleros inteligentes con apertura por aplicación móvil desarrollado para sistema operativo Android, vía Bluetooth y con registro de ingreso (CIV)”, certifico que el mismo se encuentra habilitado para su defensa pública.

_____

MSc. Fabrizio Villasís Chiriboga
Coordinador de la Escuela de
Sistemas de Automatización
Instituto Superior Tecnológico Sudamericano Quito

CERTIFICACIÓN

Por medio del presente certifico que los señores Juan Carlos de la Torre Gaibor, y Diego Josué Gálvez Sánchez han realizado y concluido su trabajo de titulación, cuyo tema es: "Diseño e implementación de un sistema prototipo de casilleros inteligentes con apertura por aplicación móvil desarrollado para sistema operativo Android, vía Bluetooth y con registro de ingreso (CIV)", para obtener el título de Tecnólogos en Sistemas de Automatización, bajo mi tutoría.

MSc. Fabrizio Vicente Villasís Chiriboga
Director del Proyecto de Titulación

AGRADECIMIENTOS

Agradecemos a Dios por permitirnos llegar hasta aquí culminando este gran paso en nuestras vidas, el de ser profesionales de la República del Ecuador. A nuestros padres que, incansablemente, han permanecido a nuestro lado; sin su apoyo nada de esto hubiera sido posible. A nuestros colegas, profesores y amigos, quienes han hecho de la experiencia del aprendizaje un acontecimiento memorable, hasta sus últimos días. Gracias a todos en general.

De igual manera, nuestros agradecimientos al Instituto Superior Tecnológico Sudamericano, por abrirnos sus puertas, y la excelencia académica con la que se han impartido las clases. Nuestro mayor agradecimiento será dejar su nombre en alto, con el diario vivir de nuestra vida laboral.

DEDICATORIA

Este proyecto lleva impregnada la dedicatoria a Dios, en primer lugar, que ha sido nuestro guía en cada paso que hemos dado, para convertirnos en profesionales de calidad y eficacia laboral. Sin su apoyo, su guía y su luz no habríamos podido culminar con la ardua tarea de ser excelentes en nuestras labores estudiantiles.

Dedicamos, además, a nuestros padres y abuelos, quienes se han constituido como pilares fundamentales de nuestras vidas. Su apoyo, su respaldo, su empuje en los peores momentos, son las raíces del fruto que se materializa en este proyecto: trabajo, responsabilidad y excelencia, ante todas las cosas.

RESUMEN

El presente documento trata sobre el diseño y desarrollo de un sistema prototipo de casilleros inteligentes con apertura por dispositivos móviles, cuya aplicación está desarrollada para el sistema operativo Android, con conexión Bluetooth y con registro de ingreso.

Al proyecto se le ha nombrado con las siglas CIV que proviene de Casilleros Inteligentes Visionary, los cuales fueron realizados utilizando software libre y componentes de hardware económicos, además se desarrolló una aplicación de fácil uso y muy intuitiva para el uso por parte de los usuarios.

Se describe el diseño e implementación del prototipo CIV, el cual contiene un microcontrolador Arduino nano el cual funciona como el control general para todos los dispositivos, como el módulo Bluetooth HC05, que permite conectarnos con el dispositivo móvil, el cual contiene la aplicación que funciona como llave, mandando el pulso directo hacia una cerradura electrónica. El sensor de fin de carrera dará un pulso de cierre y de apertura, las cuales se registrarán en la interfaz gráfica que a su vez tiene una conexión a una Base de Datos, la cual, está desarrollada en XAMPP, lo que permite la gestión mediante MySQL de forma local. La interfaz gráfica permite visualizar de manera clara y detallada la información de cierre y de apertura del casillero lo que permite al usuario estar en detalle.

Con este proyecto CIV se brinda la confianza y la seguridad a las pertenencias de los usuarios, ya que es impenetrable debido a su estructura metálica, su cerradura electrónica y su única llave que es el dispositivo móvil personal, evitando los molestos candados de llaves o con combinación los cuales pueden ser vulnerados fácilmente.

ABSTRACT

This document deals with the design and development of a prototype system of smart lockers that can be opened by mobile devices, the application of which is developed for the Android operating system, with a Bluetooth connection and with registration.

The project has been named with the acronym CIV that comes from Visionary Smart Lockers, which were made using free software and inexpensive hardware components, in addition an easy-to-use and very intuitive application was developed for use by users.

The design and implementation of the CIV prototype is described, which contains an Arduino nano microcontroller which functions as the general control for all devices, such as the HC05 Bluetooth module, which allows us to connect with the mobile device, which contains the application that works. as a key, sending the pulse directly to an electronic lock. The limit switch sensor will give a closing and opening pulse, which will be recorded in the graphical interface, which in turn has a connection to a Database, which is developed in XAMPP, which allows management through MySQL locally. The graphical interface allows a clear and detailed view of the locker closing and opening information, allowing the user to be in detail.

With this CIV project, confidence and security are provided to users' belongings, since it is impenetrable due to its metallic structure, its electronic lock and its only key, which is the personal mobile device, avoiding the annoying padlocks of keys or with combination which can be easily breached.

ÍNDICE

ÍNDICE DE FIGURAS

LISTA DE ANEXOS

1. Introducción

En el presente proyecto se plantea el diseño e implementación de un sistema prototipo de casilleros inteligentes con apertura por aplicación móvil, desarrollados para dispositivos móviles que incluyan el sistema operativo Android vía Bluetooth y con registro de ingreso. Al proyecto se lo denomina (CIV) por sus siglas Casilleros Inteligentes Visionary. El prototipo CIV tiene como finalidad dejar de lado el uso de candados y optimizar la seguridad mediante la tecnología móvil actual.

CIV está conformado por tres partes fundamentales, una de ellas es la aplicación para dispositivos móviles como un Smartphone, luego se tiene el casillero con toda la electrónica necesaria y `por último se necesita de un computador, el cual contiene la base de datos para el registro de las entradas a los casilleros.

La aplicación móvil tendrá una interfaz gráfica muy intuitiva que permitirá, de forma sencilla, acceder a los casilleros a los usuarios con la utilización de un PIN (Personal Identification Number) de acceso, el cual puede ser personalizada. El usuario con su dispositivo móvil se vinculará por medio de conexión Bluetooth a su casillero CIV y registrado en la base de datos su acceso. La computadora con la base de datos se encontrará próxima a los casilleros de la Institución y para la cual se desarrollará una interfaz gráfica para utilizar la información de la base de datos, además, contará con un menú de búsqueda en el registro de actividades la cual examina las acciones de cada casillero.

Con el prototipo CIV se busca dar solución al problema de la inseguridad de los casilleros al dejar de lado los candados y las llaves, CIV ofrece mayor seguridad permitiendo que el dispositivo móvil sea usado para el acceso y el registro de actividades de apertura y cierre del casillero.

El contenido del presente documento presenta la siguiente estructura: como primera parte se justifica y se analizan los antecedentes que dan pie a este proyecto, fundamentalmente se analiza la problemática del uso de los candados con llaves en la actualidad en los sectores educativos. Luego se explica el diseño y desarrollo del software,

así como del hardware necesario, tanto para la seguridad física como para el almacenamiento y registro de la información.

Posteriormente, se explica la implementación del sistema prototipo CIV dirigido para los sectores educativos, pero dejando que se pueda implementar en cualquier institución de cualquier tipo.

Por último, se realizan las pruebas de funcionamiento y solución de errores para finalmente presentar las conclusiones y recomendaciones.

2. Justificación

Las instituciones educativas ofrecen casilleros a sus estudiantes, con la finalidad de que puedan guardar sus pertenencias, ropa o utensilios, de acuerdo a sus necesidades. Los casilleros comúnmente son de metal y solo tienen como seguridad un candado de llave común, los cuales presentan problemas como la pérdida o extravío de las llaves. Además, que en ocasiones las llaves son de malos materiales, llegando a doblarse o inclusive a, romperse, también, pueden llegar a incomodar a las personas el tener que transportarlas.

Otra opción que tienen los casilleros es utilizar candados de combinación. Estos, sin embargo, presentan inconvenientes como: una clave simple que se pueda adivinar; que el dueño olvide recombinar su clave dejando por error abierto el casillero; o que personas mal intencionadas busquen en Internet las maneras disponibles que existen para forzar este tipo de candados, quedando vulnerables las pertenencias de los usuarios.

Para solucionar estos diversos problemas, que presentan el uso de candados en los casilleros, se ha desarrollado el presente proyecto-prototipo de Casilleros Inteligentes (CIV). Este ofrece, a los usuarios de las instituciones educativas, la facilidad de utilizar una aplicación para dispositivos móviles con sistema operativo Android, como llave que, mediante Bluetooth, permita abrir el casillero introduciendo un PIN de 4 dígitos, previamente establecido por el usuario. Además, tener un registro de control de acceso a los casilleros.

Este proyecto de Casilleros Inteligentes permitirá, a las instituciones educativas que lo posean, ofrecer a sus usuarios (estudiantes, profesores, autoridades y sistema administrativo) un sistema cómodo, seguro y novedoso para salvaguardar sus pertenencias; reemplazando las anticuadas llaves por los dispositivos móviles como el celular, que ahora son parte infaltable e inalienable de las personas.

3. Antecedentes

Las entidades educativas que ofrecen casilleros a sus estudiantes lo hacen con la finalidad de salvaguardar sus pertenencias. Estos casilleros, normalmente, son de metal y utilizan como medio de seguridad candados con llaves las cuales pueden ser extraviadas u olvidas por los estudiantes, impidiendo que se acceda al contenido del casillero. Como alternativa, existen candados con clave o combinación, pero pueden ser olvidadas por el estudiante o, al cerrar el candado, no se recombine la combinación dejando abierto y vulnerable el contenido del casillero.

Además, con el avance tecnológico se han ido implementando nuevos tipos de candados. En estos momentos existen modelos, los cuales se pueden abrir mediante la huella digital, pero estos pueden ser violentados mediante “trucos” como el uso de cinta adhesiva, plastilina y silicona, robando la huella del propietario para poder ingresar a su casillero.

Otra desventaja es el tamaño del sensor, al tener que leer la huella completa, el candado tiene un mayor tamaño, lo que le hace propenso a ser ultrajado por dueños de lo ajeno.

En la actualidad en el país todavía siguen siendo utilizados los candados tradicionales de llaves y contraseña; son pocos los lugares que han implementado seguridad digital en las zonas de áreas personales como los casilleros, debido a su costo y al pensar que serán difíciles de emplear.

Como alternativa se propone el uso del dispositivo más común en la sociedad moderna que son los celulares o smartphones, los cuales convergen todas las utilidades de la comunicación de la información en un solo lugar y a disposición de las personas, como también permiten tener el manejo o control de distintos dispositivos, como en este caso la apertura del casillero, ofreciendo comodidad, seguridad y rapidez al usuario.

CIV es todo en uno al poder tener la rapidez de la conexión Bluetooth, la seguridad de un celular personal y la comodidad de no cargar llaves; esto garantiza la eficiencia de CIV al momento de utilizar los casilleros.

4. Objetivos

4.1 Objetivo General

Diseñar e implementar un sistema prototipo de casilleros inteligentes con apertura por aplicación móvil desarrollada para Android, con comunicación Bluetooth y vinculado a una base de datos para el registro de ingreso.

4.2 Objetivos Específicos

1. Describir los conceptos, materiales y herramientas necesarios para la elaboración del proyecto, a manera de marco teórico.
2. Diseñar e implementar el proyecto partiendo de las representaciones de comportamiento y estructural.
3. Diseñar el circuito electrónico y eléctrico para el prototipo de Casilleros Inteligentes.
4. Desarrollar la aplicación para dispositivos móviles con sistema operativo Android.
5. Implementar la Base de Datos para el registro de ingresos.
6. Realizar las pruebas de funcionamiento del prototipo de Casilleros Inteligentes.

5. Marco teórico

5.1 Cerradura

Las cerraduras de seguridad son dispositivos que se instalan para mejorar en los negocios de hostelería, para aumentar la seguridad de las habitaciones y de los espacios comunes que no están abiertos al público general y son de acceso limitado al personal del hotel o colectividad.

A grandes rasgos se dividen en cerraduras de seguridad de accionamiento mecánico y las cerraduras electrónicas de seguridad, con un mayor nivel de sofisticación, con control de entradas y de horarios, mejoras en la accesibilidad y llaves virtuales como el móvil. [1]

5.1.1 Tipos de cerraduras [1]

- **Cerraduras multipunto:** Esta variedad de cerraduras es bastante segura. Esta seguridad es posible a través de los diferentes puntos de anclaje de la cerradura. Dependiendo del modelo elegido, el número de anclajes podrá ser mayor o menor.
- **Cerraduras tubulares:** El sistema de apertura es el mismo que el que ofrece el propio picaporte. Una de las principales ventajas de esta variedad de cerraduras es que se pueden cerrar desde dentro con tan solo pulsar un botón.
- **Cerraduras de sobreponer:** Este tipo de cerraduras suelen ser puestas en las puertas exteriores de las viviendas. Se ponen en la parte interior de la puerta, lo que hace que la gran parte de la cerradura quede a la vista.
- **Cerraduras embutidas:** Conocidas bajo el nombre de cerraduras empotradas, son de las más comunes en las puertas de entrada de la gran mayoría de viviendas. Puede decirse que es la cerradura principal y luego a esta se la pueden añadir otras cerraduras complementarias, con el fin de conseguir una mayor seguridad.
- **Cerraduras cilíndricas:** En muchas ocasiones son conocidas bajo el nombre de cerraduras con perfil europeo. Es otro modelo de cerradura muy utilizado en las puertas de entrada a las viviendas. Su sistema en forma de cilindro es bastante sencillo, pero solo se puede accionar si la llave que se introduce es la correcta.
- **Cerrojos:** cuentan con un botón que impide que otras personas puedan abrir el cerrojo mientras el sistema de seguridad esté activo. Además, muchos modelos cuentan también con cadena de seguridad, para aumentar un poco más la seguridad que ofrecen.

- **Cerraduras invisibles:** Estas cerraduras se abren y se cierran a través de un mando. Para aumentar la seguridad, los códigos son actualizados en cada uso.

Figura 1. Tipos de cerraduras.
Fuente: [1].

- **Cerraduras digitales:** Las mismas pueden ser abiertas a través de diferentes sistemas. Uno de los más comunes es a través de la tarjeta. Esta tarjeta se introduce en un lector y abre automáticamente la cerradura. Pero hay otros sistemas que se pueden usar con esta variedad de cerraduras como el sistema de códigos digitales o la huella dactilar.

5.2 Arduino

Arduino es una plataforma de desarrollo basada en una placa electrónica de hardware libre que incorpora un microcontrolador reprogramable y una serie de pines hembra. Estos permiten establecer conexiones entre el microcontrolador y los diferentes sensores y actuadores de una manera muy sencilla (principalmente con cables DuPont). [2]

Figura 2. Logo del Programa Arduino.
Fuente: [2].

Su entorno de programación es multiplataforma. Se puede instalar y ejecutar en sistemas operativos Windows, Mac OS y Linux.

Lenguaje de programación de fácil compresión. Su lenguaje de programación basado en C++ es de fácil compresión. C++ permite una entrada sencilla a los nuevos programadores y a la vez con una capacidad tan grande, que los programadores más avanzados pueden exprimir todo el potencial de su lenguaje y adaptarlo a cualquier situación.

Reusabilidad y versatilidad. Es reutilizable porque una vez terminado el proyecto es muy fácil poder desmontar los componentes externos a la placa y empezar con un nuevo proyecto. De igual manera todos los pines del microcontrolador están accesibles a través de conectores hembra y esto permite sacar partido de todas las bondades del microcontrolador con un riesgo muy bajo de hacer una conexión errónea. [3]

5.2.1 Arduino UNO

La placa Arduino UNO es la mejor placa para iniciar con la programación y la electrónica. Si es tu primera experiencia con la plataforma Arduino, la Arduino UNO es la opción más robusta, más usada y con mayor cantidad de documentación de toda la familia Arduino. [3]

Arduino UNO es una placa basada en el microcontrolador ATmega328P. Tiene 14 pines de entrada/salida digital (de los cuales 6 pueden ser usando con PWM), 6 entradas analógicas, un cristal de 16Mhz, conexión USB, conector Jack de alimentación, terminales para conexión ICSP y un botón de reseteo. Tiene toda la electrónica necesaria para que el microcontrolador opere, simplemente hay que conectarlo a la energía por el puerto USB o con un transformador AC-DC [2]

Figura 3. Placa Microcontrolador Arduino Uno.
Fuente: [3].

5.2.2 Arduino Nano [4]

El Arduino Nano es una pequeña y completa placa basada en el ATmega328 (Arduino Nano 3.0) o el ATmega168 en sus versiones anteriores (Arduino Nano 2.x) que se usa conectándola a una protoboard. Tiene más o menos la misma funcionalidad que el Arduino Duemilanove, pero con una presentación diferente. No posee conector para alimentación externa, y funciona con un cable USB Mini-B.

Características:

- Microcontrolador: Atmel ATmega328 (ATmega168 versiones anteriores)
- Tensión de Operación (nivel lógico): 5 V
- Tensión de Entrada (recomendado): 7-12 V
- Tensión de Entrada (límites): 6-20 V
- Pines E/S Digitales: 14 (de los cuales 6 proveen de salida PWM
- Entradas Analógicas: 8 Corriente máx por cada PIN de E/S: 40 mA
- Memoria Flash: 32 KB (ATmega328) de los cuales 2KB son usados por el bootloader (16 KB – ATmega168)
- SRAM: 2 KB (ATmega328) (1 KB ATmega168)
- EEPROM: 1 KB (ATmega328) (512 bytes – ATmega168)
- Frecuencia de reloj: 16 MHz
- Dimensiones: 18,5mm x 43,2mm
- Energía: El Arduino Nano posee selección automática de la fuente de alimentación y puede ser alimentado a través de:
- Una conexión Mini-B USB.
- Una fuente de alimentación no regulada de 6-20V (pin 30).
- Una fuente de alimentación regulada de 5V (pin 27)

Al alimentar el Arduino a través del Mini USB, el CH340 proporciona una salida de 3.3V en el pin 16 de la placa. Por ende, cuando se conecta a una fuente externa (no USB), los 3.3V no se encuentran disponibles.

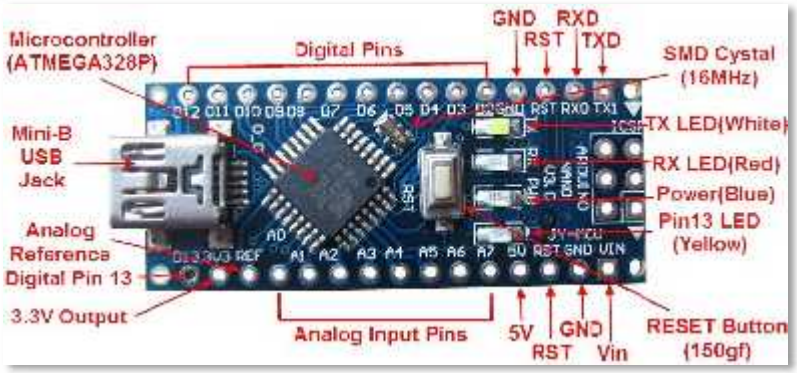


Figura 4. Placa Microcontrolador Arduino Nano.
Fuente: [4].

5.3 LCD [5]

El LCD (Liquid Crystal Display) o pantalla de cristal líquido es un dispositivo empleado para la visualización de contenidos o información de una forma gráfica, mediante caracteres, símbolos o pequeños dibujos dependiendo del modelo. Está gobernado por un microcontrolador el cual dirige todo su funcionamiento.

En este caso vamos a emplear un LCD de 16x2, esto quiere decir que dispone de 2 filas de 16 caracteres cada una. Los píxeles de cada símbolo o carácter, varían en función de cada modelo.

¿Cómo es su conexionado?

En la siguiente imagen de Proteus se puede observar la estructura de sus pines. Lo podemos dividir en los pines de alimentación, pines de control y los pines del bus de datos bidireccional. Por lo general podemos encontrar además en su estructura los pines de Ánodo de led backlight y cátodo de led backlight.

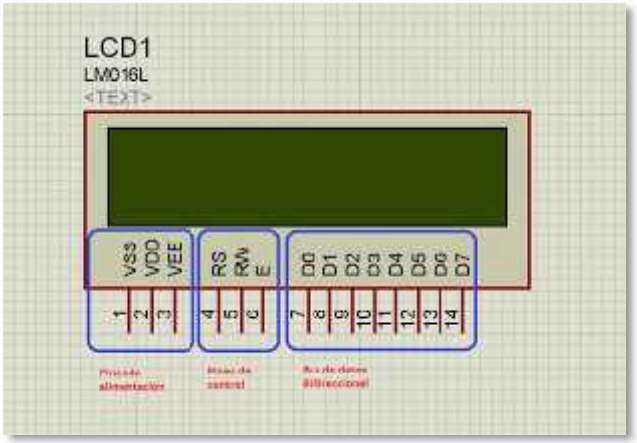


Figura 5. Esquema gráfico de LCD.
Fuente: [5].

Pines de alimentación:

- ***Vss:*** Gnd
- ***Vdd:*** +5voltios
- ***Vee:*** corresponde al pin de contraste, lo regularemos con un potenciómetro de 10K conectado a Vdd.

Pines de control: [5]

RS: Corresponde al pin de selección de registro de control de datos (0) o registro de datos (1). Es decir, el pin RS funciona paralelamente a los pines del Bus de datos. Cuando RS es 0 el dato presente en el Bus pertenece a un registro de control/instrucción. Y cuando RS es 1 el dato presente en el Bus de datos pertenece a un registro de datos o un carácter. ***RW:*** Corresponde al pin de Escritura (0) o de Lectura (1). Permite escribir un dato en la pantalla o leer un dato desde la pantalla.

E: Corresponde al pin Enable o de habilitación. Si E (0) esto quiere decir que el LCD no está activado para recibir datos, pero si E (1) se encuentra activo y podemos escribir o leer desde el LCD.

Pines de Bus de datos: [5]

El Bus de datos bidireccional comprende desde los pines D0 a D7. Para realizar la comunicación con el LCD podemos hacerlo utilizando los 8 bits del Bus de datos (D0 a D7) o empleando los 4 bits más significativos del Bus de datos (D4 a D7). En este caso vamos a explicar la comunicación con el Bus de 4 bits.

¿DDRAM y CGROM? [5]

Son las dos zonas de memoria del LCD.

La memoria DDRAM (Data Display RAM): corresponde a una zona de memoria donde se almacenan los caracteres que se van a representar en pantalla. Es decir, es la memoria donde se almacenan los caracteres a mostrar con su correspondiente posición.

La memoria CGROM es una memoria interna donde se almacena una tabla con los caracteres que podemos visualizar en el LCD. En la imagen podemos ver un ejemplo de la tabla con un contenido de 192 caracteres.

5.3.1 LCD I2C [6]

El controlador de LCD I2C es un dispositivo que permite controlar una pantalla a través del Bus I2C, usando únicamente dos cables. La conexión es sencilla, simplemente alimentamos el módulo desde Arduino mediante GND y 5V y conectamos el pin SDA y SCL de Arduino con los pines correspondientes del controlador LCD I2C.

Figura 6. Módulo I2C para LCD.
Fuente: [6].

Este singular módulo es totalmente compatible para las versiones **1602 LCD** y **2004**.

LCD. Ambos display LCD 16×2 y 20×4 tienen un total de 16 pines en la parte de arriba, el módulo puede ser utilizado para ambas pantallas, solo hay que adaptar algunos pines hembras. Este módulo simplemente se soldará en la parte trasera de la pantalla, para ello se necesita los siguientes materiales:

- Módulo de interfaz serial I2C para display LCD
- Arduino (funciona con cualquier tipo de placa)
- Algunos jumper macho-hembra
- Una pantalla LCD 16×2 o 20×4, cualquier tipo de estas dos pantallas funciona.

El único trabajo por hacer es soldar correctamente el módulo I2C en la parte trasera de la pantalla LCD y con esta simple acción se podrá tener una pantalla que se adapte a las necesidades sin los 16 cables que se requieren para conectar la pantalla LCD, con este módulo se pasa de 16 cables a solo 4.

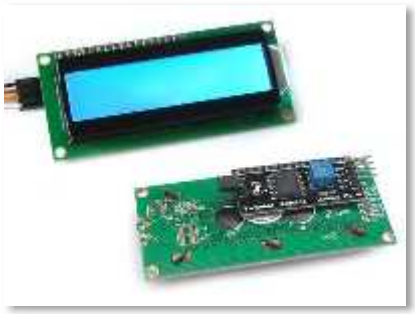


Figura 7. LCD I2C.
Fuente: [6].

5.4 Modulo Bluetooth [7]

El Bluetooth es un estándar de comunicación inalámbrica que permite la transmisión de datos a través de radiofrecuencia en la banda de 2,4 GHz. Existen muchos módulos Bluetooth para usarlos en proyectos de electrónica, pero los más utilizados son los módulos de JY-MCU, ya que son muy económicos y fáciles de encontrar en el mercado. Son módulos pequeños y con un consumo muy bajo que permitirán agregar funcionalidades Bluetooth al Arduino. Estos módulos contienen el chip con una placa de desarrollo con los pins necesarios para la comunicación serie.

Figura 8. Logo de Arduino y Bluetooth.
Fuente: [7].

Existen dos modelos de módulos Bluetooth: el HC-05 que puede ser maestro/esclavo (master/slave), y el HC-06 que solo puede actuar como esclavo (slave). La diferencia entre maestro y esclavo es que en modo esclavo es el dispositivo quien se conecta al módulo, mientras que en modo maestro es el módulo quien se conecta con un dispositivo.

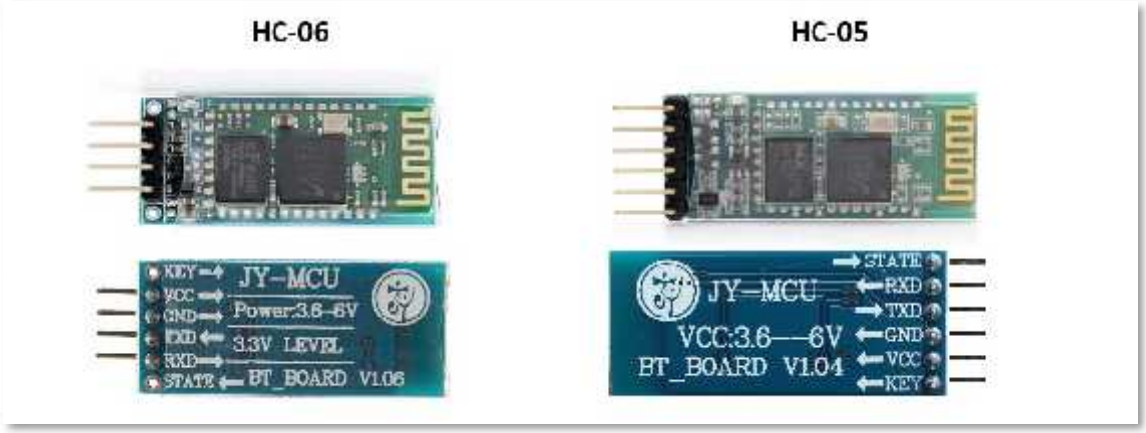


Figura 9. Módulos Bluetooth.
Fuente: [8].

Físicamente, los dos módulos son muy parecidos, solo varían algunas conexiones. Los pines que se encontraran son los siguientes:

- Vcc: Alimentación del módulo entre 3,6V y 6V.
- GND: La masa del módulo.
- TXD: Transmisión de datos.
- RXD: Recepción de datos a un voltaje de 3,3V.
- KEY: Poner a nivel alto para entrar en modo configuración del módulo (solo el modelo HC-05)
- STATE: Para conectar un led de salida para visualizar cuando se comuniquen datos. [9]

5.4.1 Módulo Bluetooth HC-06 [8]

El módulo Bluetooth HC-06 permite conectar los proyectos con Arduino a un smartphone, celular o PC de forma inalámbrica (Bluetooth), con la facilidad de operación de un puerto serial. La transmisión se realiza totalmente en forma transparente al programador, por lo que se conecta en forma directa a los pines seriales del microcontrolador preferido (respetando los niveles de voltaje, ya que el módulo se alimenta con 3.3V). Todos los parámetros del módulo se pueden configurar mediante comandos AT. La placa también incluye un regulador de 3.3V, que permite alimentar el módulo con un voltaje entre 3.6V - 6V. Este módulo es el complemento ideal para los proyectos de robótica, domótica y control remoto con Arduino, PIC, Raspberry PI, ESP8266, ESP32, STM32, etc.

La comunicación Bluetooth se da entre dos tipos de dispositivos: un maestro y un esclavo. Si el objetivo es conectar el proyecto a un smartphone Android podemos utilizar tanto un módulo HC-06 o un HC-05 configurado como esclavo. El módulo Bluetooth HC-06 viene configurado de fábrica para trabajar como esclavo, es decir, preparado para escuchar peticiones de conexión. Por otra parte, si el objetivo es conectar dos proyectos, necesitaremos utilizar un módulo HC-05 configurado como maestro y un HC-06 (esclavo).

Este módulo cumple con las especificaciones del estándar Bluetooth 2.0 que es perfectamente compatible con celulares o Smartphone Android, mas no con los IPhone. Para trabajar con IPhone recomendamos utilizar el Módulo Bluetooth 4.0 BLE HM-10, que también es compatible con los celulares Android modernos.

Figura 10. Módulo HC-06.
Fuente: [8].

5.4.2 Módulo Bluetooth HC-05 [10]

El módulo de Bluetooth HC-05 es el que ofrece una mejor relación de precio y características, ya que es un módulo Maestro-Esclavo, quiere decir que además de recibir conexiones desde una PC o Tablet, también es capaz de generar conexiones hacia otros dispositivos Bluetooth. Esto permite, por ejemplo, conectar dos módulos de Bluetooth y formar una conexión punto a punto para transmitir datos entre dos microcontroladores o dispositivos. En otro artículo posterior veremos cómo configurar dos módulos HC-05 para que se enlacen entre ellos y podamos transmitir información de un punto a otro.

El HC-05 tiene un modo de comandos AT que debe activarse mediante un estado alto en el PIN34 mientras se enciende (o se resetea) el módulo. En las versiones para protoboard este pin viene marcado como “Key”. Una vez que estamos en el modo de comandos AT, podemos configurar el módulo Bluetooth y cambiar parámetros como el nombre del dispositivo, password, modo maestro/esclavo, etc. Para comunicarnos con el módulo y configurarlo, es necesario tener acceso al módulo mediante una interfaz serial. Podemos usar un Arduino con un par de cables (aprovechando el puente USB-Serial del Arduino), un kit para XBee o un simple MAX3232 en el puerto serie de la PC.

Figura 11. Módulo HC-05.
Fuente: [10].

5.5 Android [11]

Android es un sistema operativo para móviles, creado por la empresa Android Inc. Esta empresa fue adquirida por Google en el año 2005. Hasta dos años después, en el 2007, no se presentó en sociedad con la primera versión.

Las características más importantes de Android son:

- Es un software de código abierto. Cualquier persona puede acceder al código fuente y modificarlo según sus requerimientos.
- Está basado en Linux, un sistema operativo de código abierto para ordenadores y portátiles.
- Lenguaje de programación nativo, Java.
- Catálogo de aplicaciones en Google Play.
- Soporte para muchos formatos multimedia.

5.5.1 Android Studio [11]

Android Studio es un entorno de desarrollo integrado (IDE), basado en IntelliJ IDEA de la compañía JetBrains, que proporciona varias mejoras con respecto al plugin ADT (Android Developer Tools) para Eclipse. Android Studio utiliza una licencia de software libre Apache 2.0, está programado en Java y es multiplataforma.

Fue presentado por Google el 16 de mayo del 2013 en el congreso de desarrolladores Google I/O, con el objetivo de crear un entorno dedicado en exclusiva a la programación de aplicaciones para dispositivos Android, proporcionando a Google un mayor control sobre el proceso de producción. Se trata pues de una alternativa real a Eclipse, el IDE recomendado por Google hasta la fecha, pero que presentaba problemas debido a su lentitud en el desarrollo de versiones que solucionaran las carencias actuales (es indispensable recordar que Eclipse es una plataforma de desarrollo, diseñada para ser extendida a través de plugins).

Android Studio se ha mantenido durante todo este tiempo en versión beta, pero desde el 8 de diciembre de 2014, en que se liberó la versión estable de Android Studio 1.0, Google ha pasado a recomendarlo como el IDE para desarrollar aplicaciones para su sistema operativo, dejando el plugin ADT para Eclipse de estar en desarrollo activo. Esta versión se puede descargar desde la web de Android Developer.

Figura 12. Logo Aplicación Android Studio.
Fuente: [11].

5.5.2 App Inventor 2 [12]

App Inventor 2 (AI2) es la versión mejorada de una herramienta de programación creada por el MIT (Instituto Tecnológico de Massachusetts) y que fue adoptada por Google para sus usuarios como solución para crear de una forma sencilla aplicaciones para dispositivos Android.

El proceso de creación consta de 3 pasos:

1. Diseñador. Muestra el display de un móvil y se utiliza para el diseño de las pantallas de la aplicación donde se situarán los distintos componentes: imágenes, botones audios, textos, etc. configurando sus propiedades (aspecto gráfico, comportamiento, etc.).
2. Editor de bloques. Permite programar de una forma visual e intuitiva el flujo de funcionamiento del programa utilizando bloques.
3. Generador de la aplicación. Una vez terminada la aplicación se puede emular desde MIT AI2 Companion o desde el emulador de APP Inventor además de generar el instalador APK obteniéndose un código QR para su descarga desde el móvil o bien el propio archivo APK para descargar y enviar.

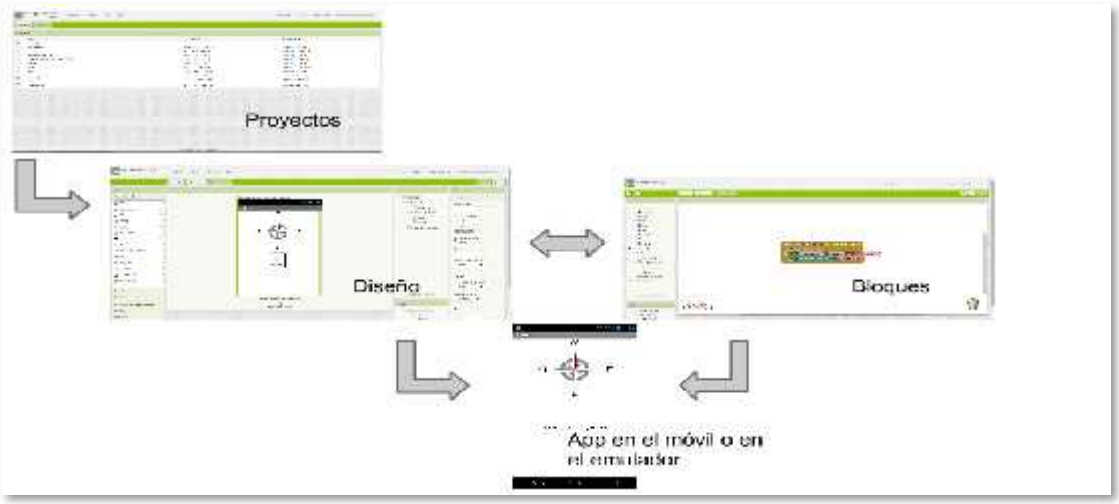


Figura 13. Esquema de funcionamiento MIT App Inventor 2.
Fuente: [12].

AI2 proporciona una herramienta en línea accesible a través de un navegador web si se dispone de una cuenta de usuario en Google. El equipo recomendado es un ordenador PC (Windows, Mac o Linux) – no una tableta - y el navegador recomendado es la última versión de Google Chrome o Mozilla Firefox (Internet Explorer no está soportado). No es necesario tener instalado en el equipo Java ni ningún otro programa.

Figura 14. Logo MIT App Inventor.
Fuente: [12].

5.6 Base de Datos [13]

Desde el punto de vista informático, la Base de Datos es un sistema formado por un conjunto de datos almacenados en discos que permiten el acceso directo a ellos y un conjunto de programas que manipulen ese conjunto de datos.

Cada Base de Datos se compone de una o más tablas que guarda un conjunto de datos. Cada tabla tiene una o más columnas y filas. Las columnas guardan una parte de la

información sobre cada elemento que queramos guardar en la tabla, cada fila de la tabla conforma un registro.

Características

Entre las principales características de los sistemas de Base de Datos podemos mencionar:

- Independencia lógica y física de los datos.
- Redundancia mínima.
- Acceso concurrente por parte de múltiples usuarios.
- Integridad de los datos.
- Consultas complejas optimizadas.
- Seguridad de acceso y auditoría.
- Respaldo y recuperación.
- Acceso a través de lenguajes de programación estándar.

5.6.1 Sistema de Gestión de Base de Datos (SGBD)

Los Sistemas de Gestión de Base de Datos (en inglés Databas Management System) son un tipo de software muy específico, dedicado a servir de interfaz entre la Base de Datos, el usuario y las aplicaciones que la utilizan. Se compone de un lenguaje de definición de datos, de un lenguaje de manipulación de datos y de un lenguaje de consulta. [13]

5.6.2 Tipos de Base de Datos [13]

Entre los diferentes tipos de Base de Datos, se pueden encontrar los siguientes:

- **MySQL:** es una Base de Datos con licencia GPL basada en un servidor. Se caracteriza por su rapidez. No es recomendable usar para grandes volúmenes de datos.
- **PostgreSql y Oracle:** Son sistemas de Base de Datos poderosos. Administra muy bien grandes cantidades de datos, y suelen ser utilizadas en intranets y sistemas de gran calibre.
- **Access:** Es una Base de Datos desarrollada por Microsoft. Esta Base de Datos, debe ser creada bajo el programa Access, el cual crea un archivo .mdb con la estructura ya explicada.
- **Microsoft SQL Server:** es una Base de Datos más potente que Access desarrollada por Microsoft. Se utiliza para manejar grandes volúmenes de informaciones.

Figura 15. Tipos de Base de Datos.
Fuente: [13].

5.7 NetBeans [14]

NetBeans es un programa que sirve como IDE (un entorno de desarrollo integrado) que permite programar en diversos lenguajes.

El desarrollo de software se ha diversificado mucho basándose en la cantidad de lenguajes que existen para la programación. Sin embargo, hay lenguajes que van imponiéndose como estándares, entre ellos están Java, PHP, HTML, C++, C#, Ruby.

El problema que se presenta a la mayoría de los programadores es contar con un entorno de desarrollo que sea completo, eficaz, fácil de usar y sea en lo posible gratuito. Todos esos requerimientos se encuentran en NetBeans.

NetBeans es ideal para trabajar con el lenguaje de desarrollo JAVA (y todos sus derivados), así como también ofrece un excelente entorno para programar en PHP. También se puede descargar una vez instalado NetBeans, los complementos para programar en C++. La IDE de NetBeans es perfecta. Tiene un excelente balance entre una interfaz con múltiples opciones y el editor puede autocompletar el código.

Figura 16. Logo Aplicación NetBeans.
Fuente: [14].

6. Desarrollo del Proyecto CIV

En este proyecto la tecnología y las herramientas que están a disposición en el mercado local, dictaminan el diseño y la implementación del mismo. Se empieza el desarrollo del proyecto con el diseño de la aplicación móvil para dispositivos móviles con sistema operativo Android, se continúa con la configuración central del microcontrolador del Arduino nano, dispositivo de conexión Bluetooth, visualizador LCD, sensores de fin de carrera y cerradura electrónica, es decir, todo lo referente a la implementación de cada casillero. Y, por último, se detalla la programación de la interfaz visual del administrador, la cual se comunica con la Base de Datos, para el registro de ingresos por parte de los usuarios.

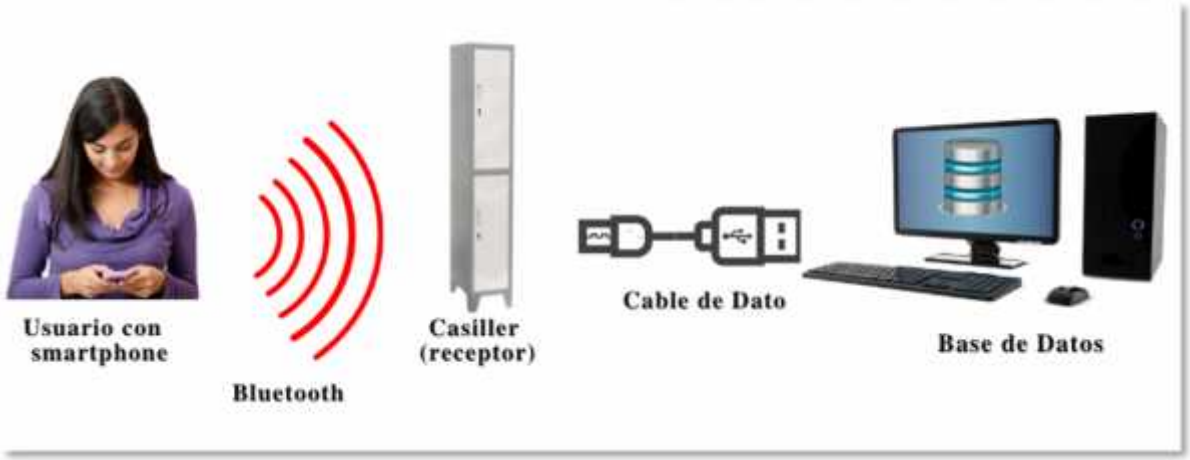


Figura 17. Representación de la idea del funcionamiento del CIV.
Fuente: El autor.

La siguiente Figura 18, se muestra la representación de comportamiento, mostrando las ventajas de CIV a los usuarios.

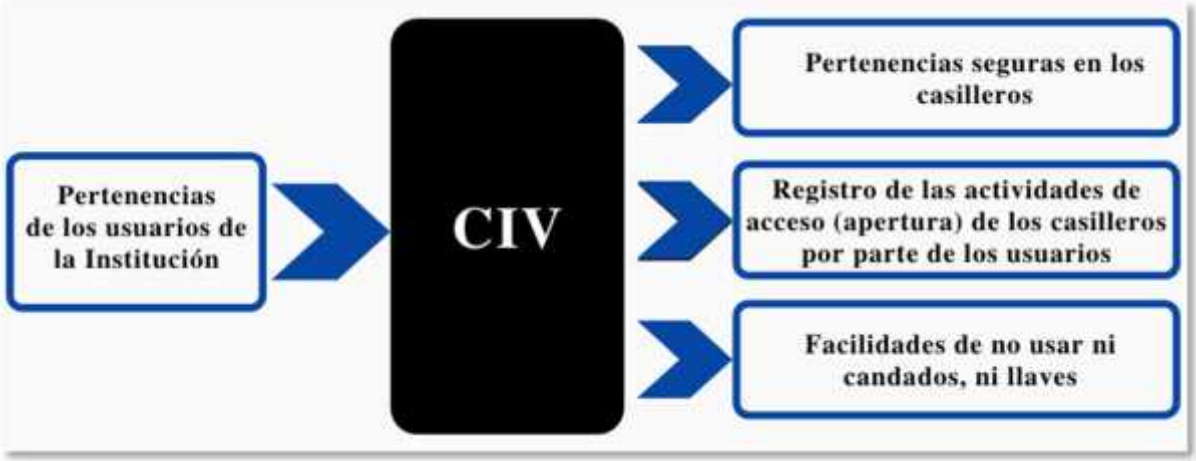


Figura 18. Esquema de comportamiento CIV.
Fuente: El autor.

Como se observa en la Figura 19, la estructura de CIV consta de tres partes fundamentales la aplicación (App), la electrónica/mecánica del casillero y la Base de Datos para el registro de actividad por parte de los usuarios.

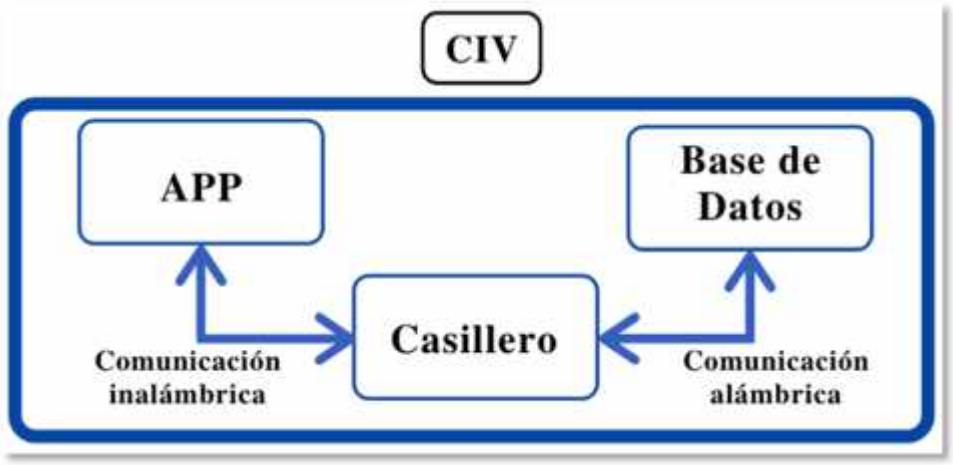


Figura 19. Esquema estructural CIV.
Fuente: El autor.

6.1 Esquema estructural de CIV

En la Figura 20 se muestra el esquema de funcionamiento de CIV el cual inicia desde el usuario con su aplicación móvil enviando los pulsos al microcontrolador Arduino y registrando la información en el programa de administración mediante la interfaz gráfica.

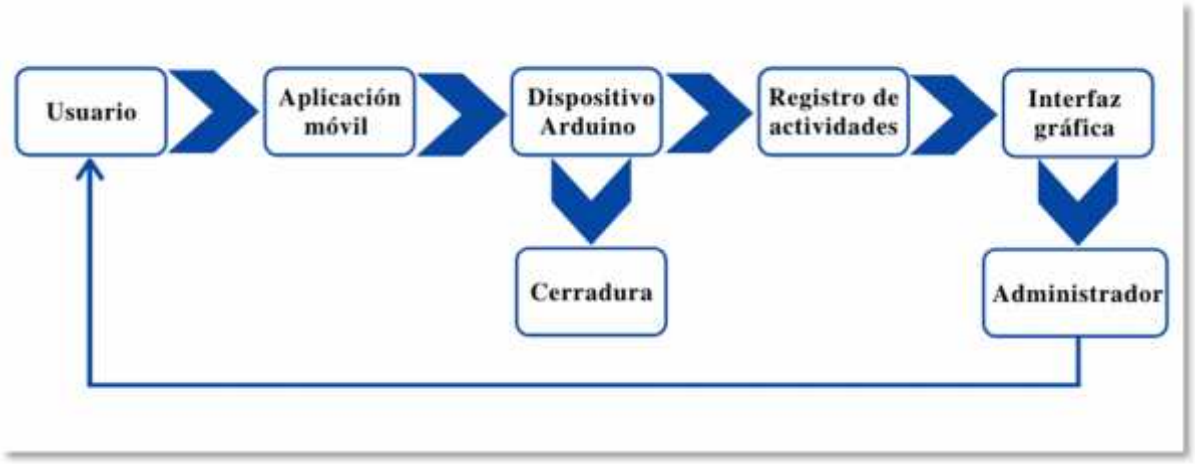


Figura 20. Esquema estructural del CIV.
Fuente: El autor.

En la Figura 21 se muestra la comunicación desde el primer dispositivo hasta la Base de Datos.

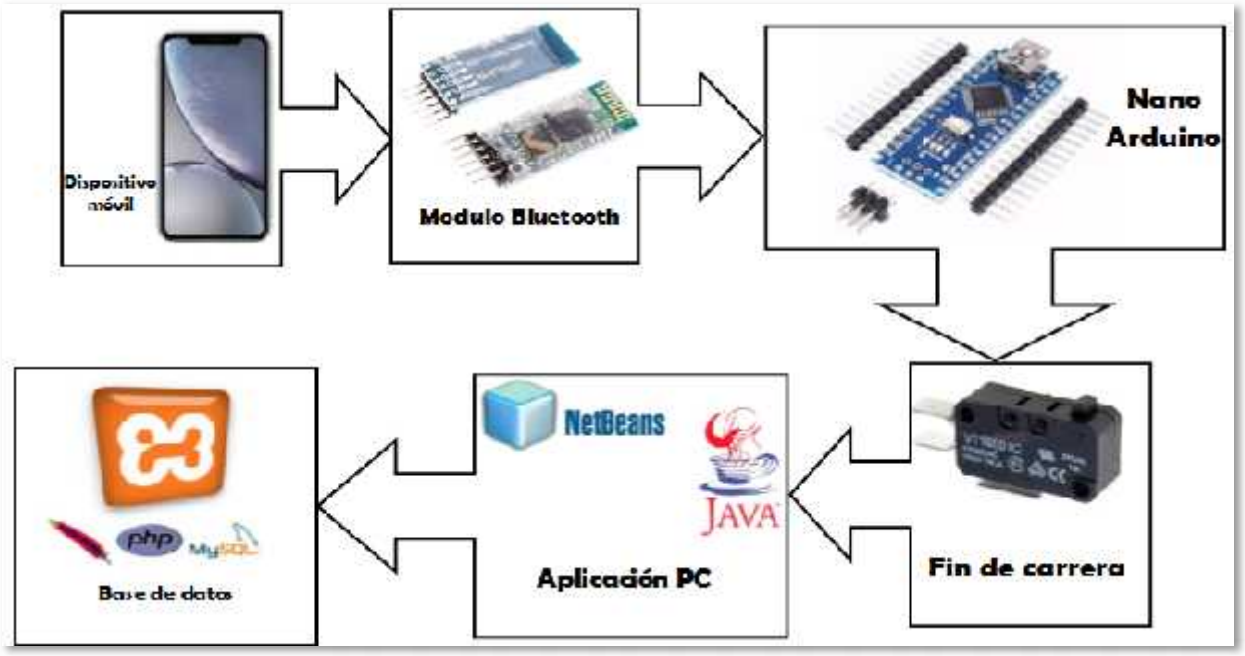


Figura 21 Esquema de envío y recepción de información.
Fuente: El autor.

6.2 Desarrollo de la aplicación para Android

Para que el dispositivo móvil pueda enviar la información del usuario es necesario desarrollar una aplicación móvil, para el sistema operativo Android, para lo cual se utiliza el entorno de desarrollo de software llamado MIT APP Inventor 2.

La siguiente Figura 22, muestra la representación de comportamiento, mostrando la manera en que se utiliza la aplicación CIV.

Figura 22. Esquema de comportamiento aplicación móvil CIV.
Fuente: El autor.

En concordancia, la Figura 23 muestra el esquema estructural que se va a utilizar para el desarrollo de la aplicación para dispositivos móviles con Android en MIT APP Inventor 2.

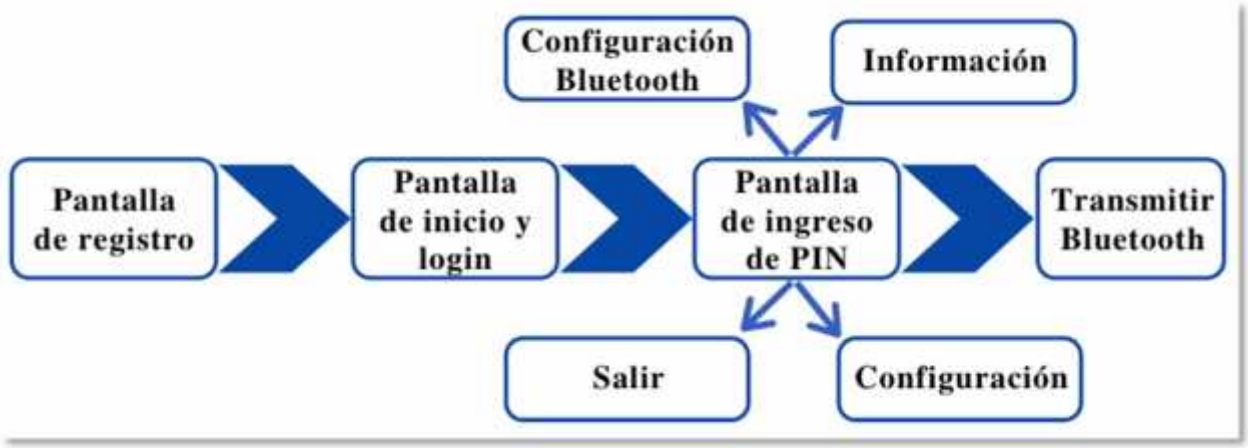


Figura 23. Esquema estructural aplicación móvil CIV.
Fuente: El autor.

6.2.1 Pantalla de registro

Como se mencionó en el diagrama estructural de la aplicación para dispositivos móviles Android la primera pantalla de registro, al momento de abrir la aplicación, verificará que la conexión Bluetooth esté encendida, en caso de no estarlo, mostrará una notificación como se puede apreciar en la Figura 24.

Figura 24. Pantalla de registro notificación de conexión Bluetooth en Aplicación móvil CIV.
Fuente: El autor.

El usuario tendrá que registrar sus datos e introducir un PIN (4 dígitos) de fábrica; es decir un valor que se entrega para configurar por primera vez, después se ingresará un nuevo PIN y se lo confirmará.

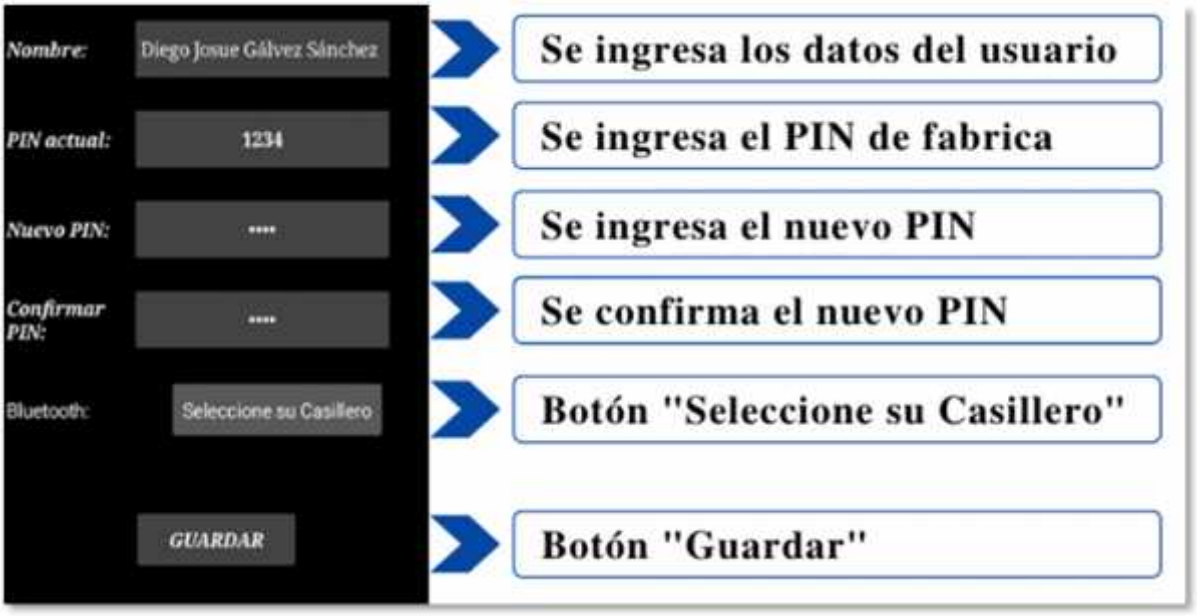


Figura 25. Pantalla de registro Aplicación móvil CIV.
Fuente: El autor.

Al pulsar el botón “Seleccione su Casillero” se desplegará una lista de dispositivos vinculados solo tendrá que seleccionar la conexión Bluetooth del casillero y automáticamente regresará a la pantalla de registro, para terminar, se pulsará el botón “GUARDAR”.

Figura 26. Pantalla dispositivos Bluetooth vinculados a móvil usuario.
Fuente: El autor.

A continuación, se detalla la programación en bloques enlazados que formar la aplicación. Primero se inicializan las variables que van a ayudar a guardar los datos como son: usuario, verificar PIN actual y guardar el nuevo PIN; se vinculan las variables globales a campos de texto, los cuales, hay que llenar con los datos en la pantalla de registro.

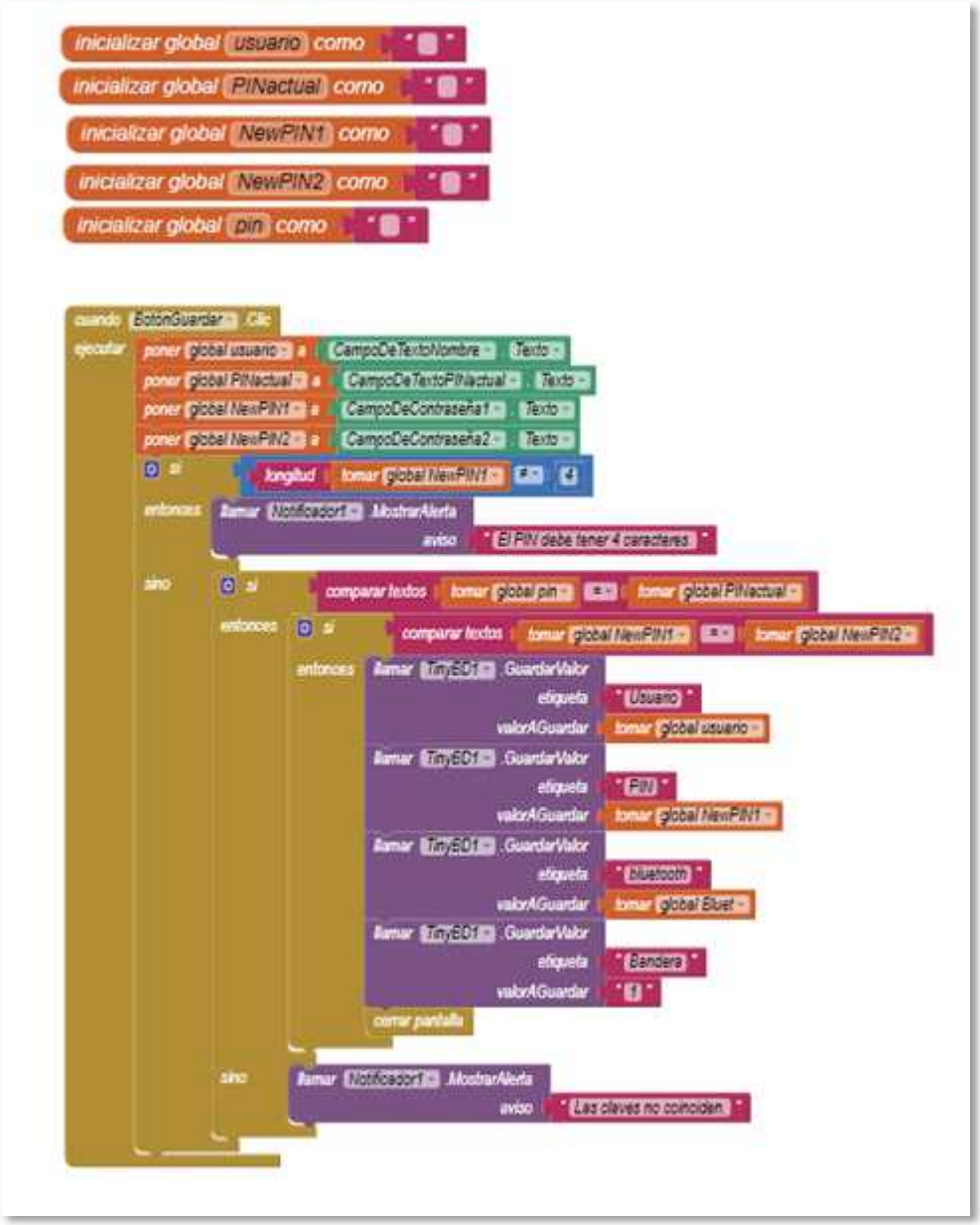


Figura 27. Programación de la pantalla de registro.
Fuente: El autor.

Al momento de ingresar los datos si se supera el número máximo del PIN que son 4 caracteres se mostrará una alerta de que el “PIN debe tener 4 caracteres”.

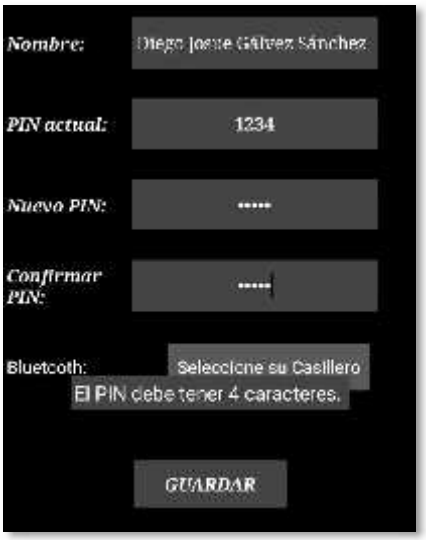


Figura 28. Pantalla de registro alerta PIN.
Fuente: El autor.

Una vez comprobado que este parámetro esté correcto, empieza la verificación del PIN actual, el nuevo PIN y confirmar el nuevo PIN. En caso de no coincidir algún PIN se mostrará la siguiente alerta “Las claves no coinciden”.

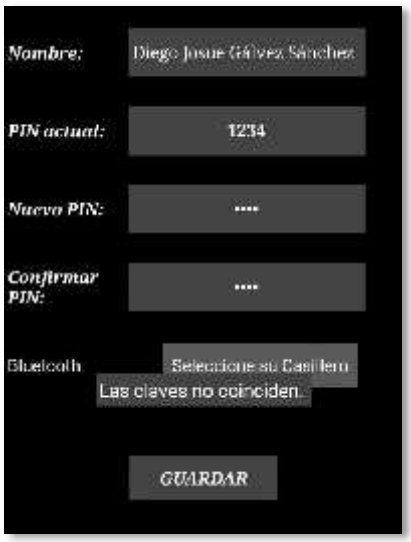


Figura 29. Pantalla de registro alerta comprobación de PIN.
Fuente: El autor.

6.2.2 Pantalla de inicio

En esta pantalla se encuentra el mensaje de bienvenida donde se refleja el nombre del usuario registrado.

Figura 30. Pantalla de inicio.
Fuente: El autor.

Cuando el dispositivo ya se encuentre registrado y se ingrese a la aplicación se mostrará en la pantalla de inicio una notificación, en caso de que el Bluetooth se encuentre apagado, el cual será "¡ATENCIÓN! Bluetooth no está habilitado. ¿Desea encenderlo?".

Si no se activa el Bluetooth, la aplicación CIV se cerrará.

Figura 31. Pantalla de inicio notificación Bluetooth.
Fuente: El autor.

Para lograrlo, en la parte de bloques (programación) se comienza con la configuración de un ***Notificador1***, para encender el Bluetooth en el dispositivo móvil si no lo estuviera. Si se selecciona la opción ***No*** la aplicación se cerrará. Se toma los datos de la variable ***usuario*** para mostrarlo en pantalla.

Luego se toma la información de la pantalla de registro que se guarda en una Base de Datos para darnos acceso a la siguiente pantalla.

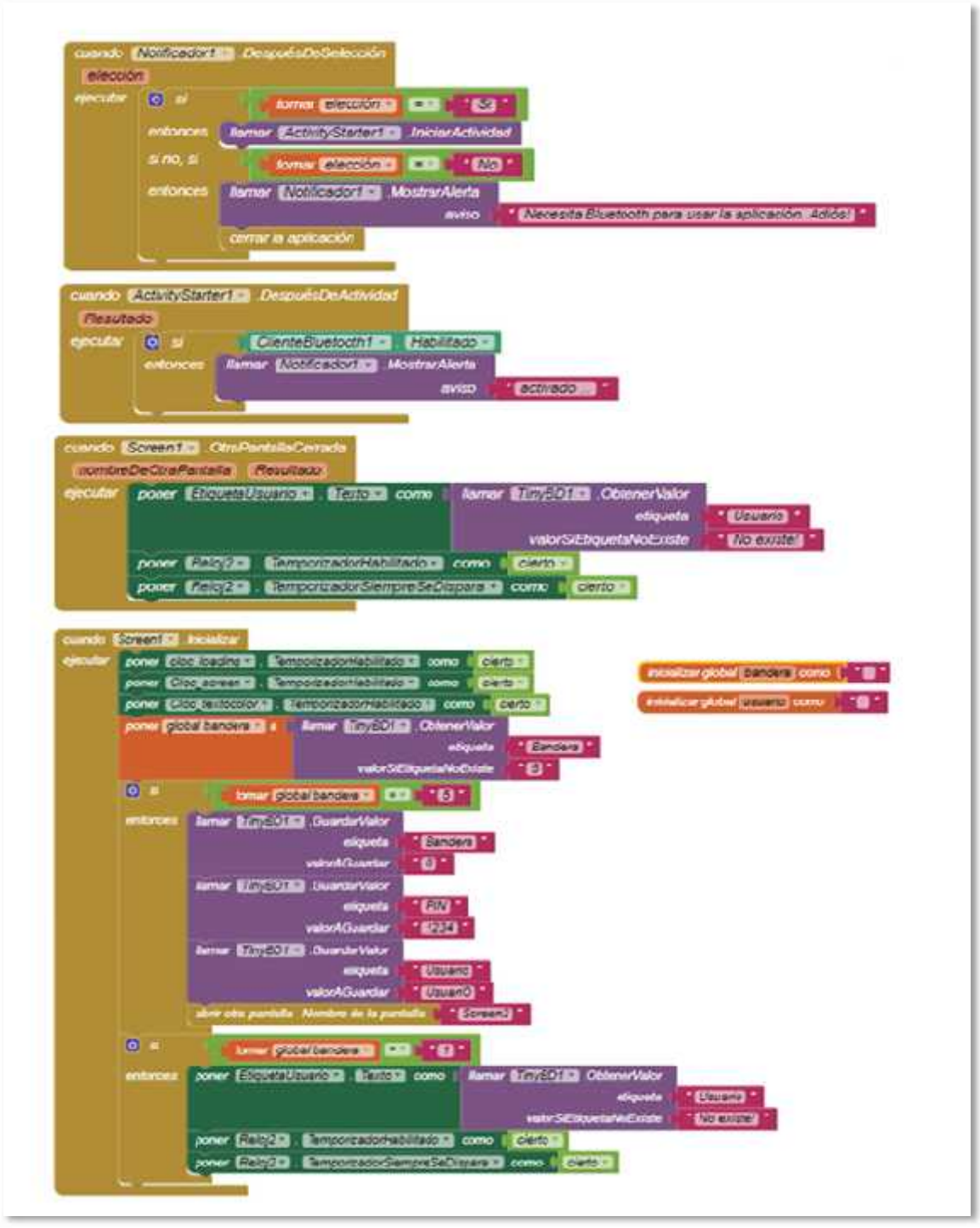


Figura 32. Programación pantalla de inicio.
Fuente: El autor.

En los siguientes bloques se encuentra el color y las imágenes que dan el efecto de rotación y movimiento de “cargando…”.

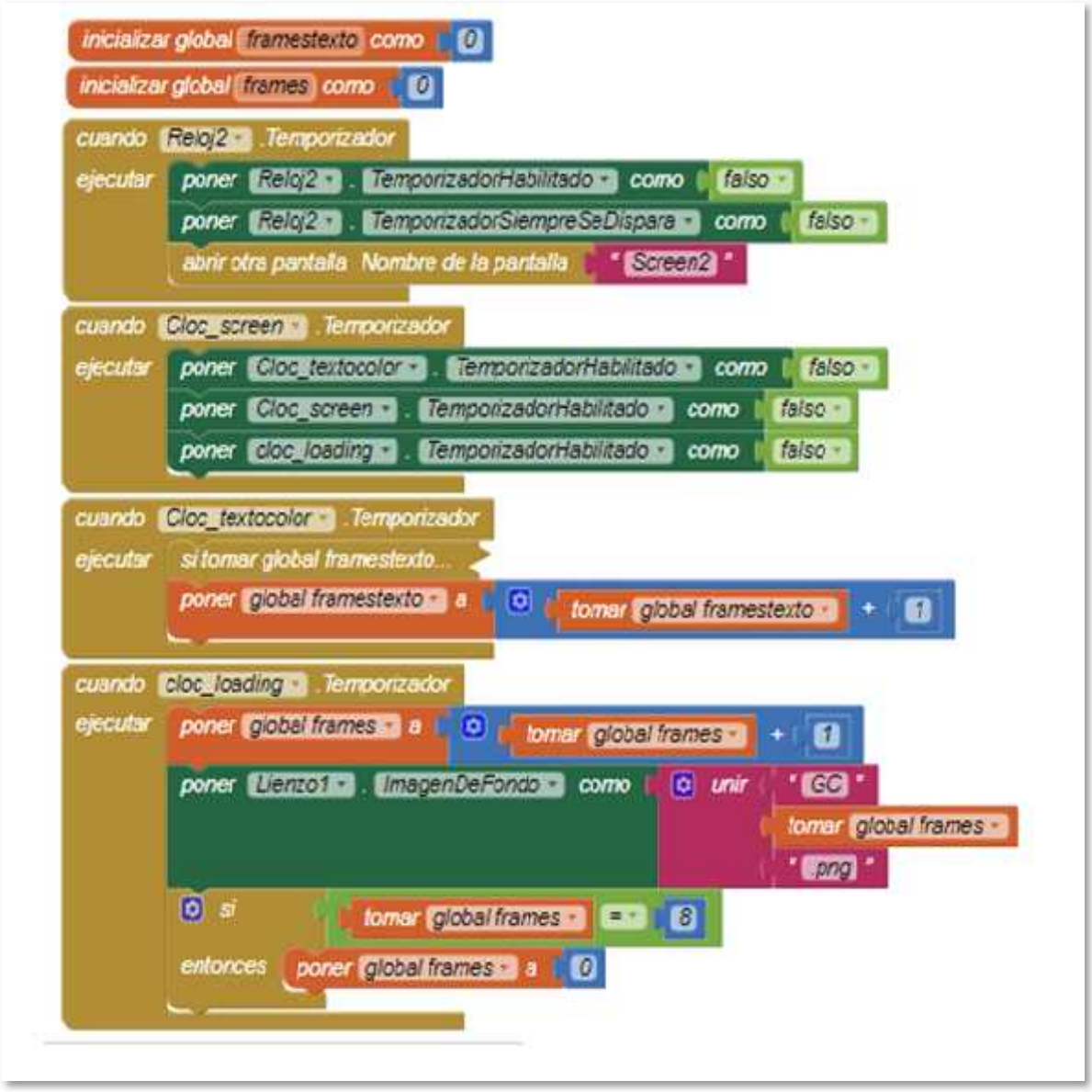


Figura 33. Pantalla de Inicio programación efecto de rotación.
Fuente: El autor.

6.2.3 Pantalla de apertura y opciones de CIV

En esta pantalla, en la parte superior, se encuentra una barra con cinco opciones:

Figura 34. Pantalla de apertura y opciones de CIV.
Fuente: El autor.

- Botón de configuración: El cual permitirá cambiar de nuevo el PIN y el nombre del usuario para la aplicación CIV enviándonos a la pantalla de registro.
- Botón de Bluetooth: Permite verificar que la aplicación móvil esté vinculada con el Bluetooth del casillero.

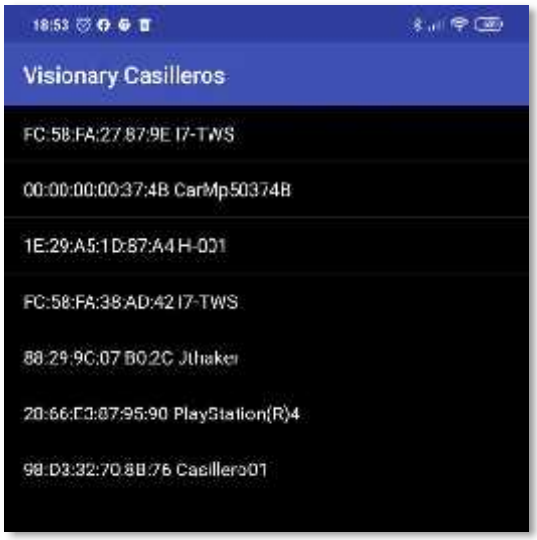


Figura 35. Pantalla de apertura opción Bluetooth.
Fuente: El autor.

- Botón de información: Muestra una notificación en pantalla con la forma de uso de la aplicación.

Figura 36. Pantalla de apertura opción información.
Fuente: El autor.

- Botón de compartir: Permite ir a un enlace el cual dejará compartir la aplicación o descargarla a un nuevo dispositivo móvil

Figura 37. Pantalla de apertura opción compartir.
Fuente: El autor.

- Botón salir: Cierra la aplicación.

A continuación, en la pantalla de apertura de CIV se muestra la información registrada del usuario, en la parte inferior de la pantalla tenemos la sección para ingresar el PIN el cual tiene que ser igual al de la pantalla de registro, si por error no se ingresa el PIN correcto, la aplicación no hará ninguna acción, si se ingresa más caracteres la aplicación mostrará una alerta “El PIN soló es de 4 caracteres”.

Figura 38. Pantalla de apertura alerta PIN.
Fuente: El autor.

En la parte inferior también encontramos el botón “Abrir” el cual enviará un pulso que permitirá la apertura de CIV y cerrará la aplicación automáticamente.

Si el Bluetooth del dispositivo móvil no se encuentra activo la aplicación CIV se cerrará al instante.

En la parte de bloques (programación) se encuentran las variables globales ***pin***, ***piningresado*** las cuales verifican la información en la Base de Datos de la aplicación al momento de ingresar el PIN, si es correcta, enviará el pulso de apertura al casillero, caso contrario, no realizará ninguna acción. Además, hace la verificación que solo se ingrese los cuatro caracteres, sino se cumple, notificará que solo se permiten cuatro caracteres.

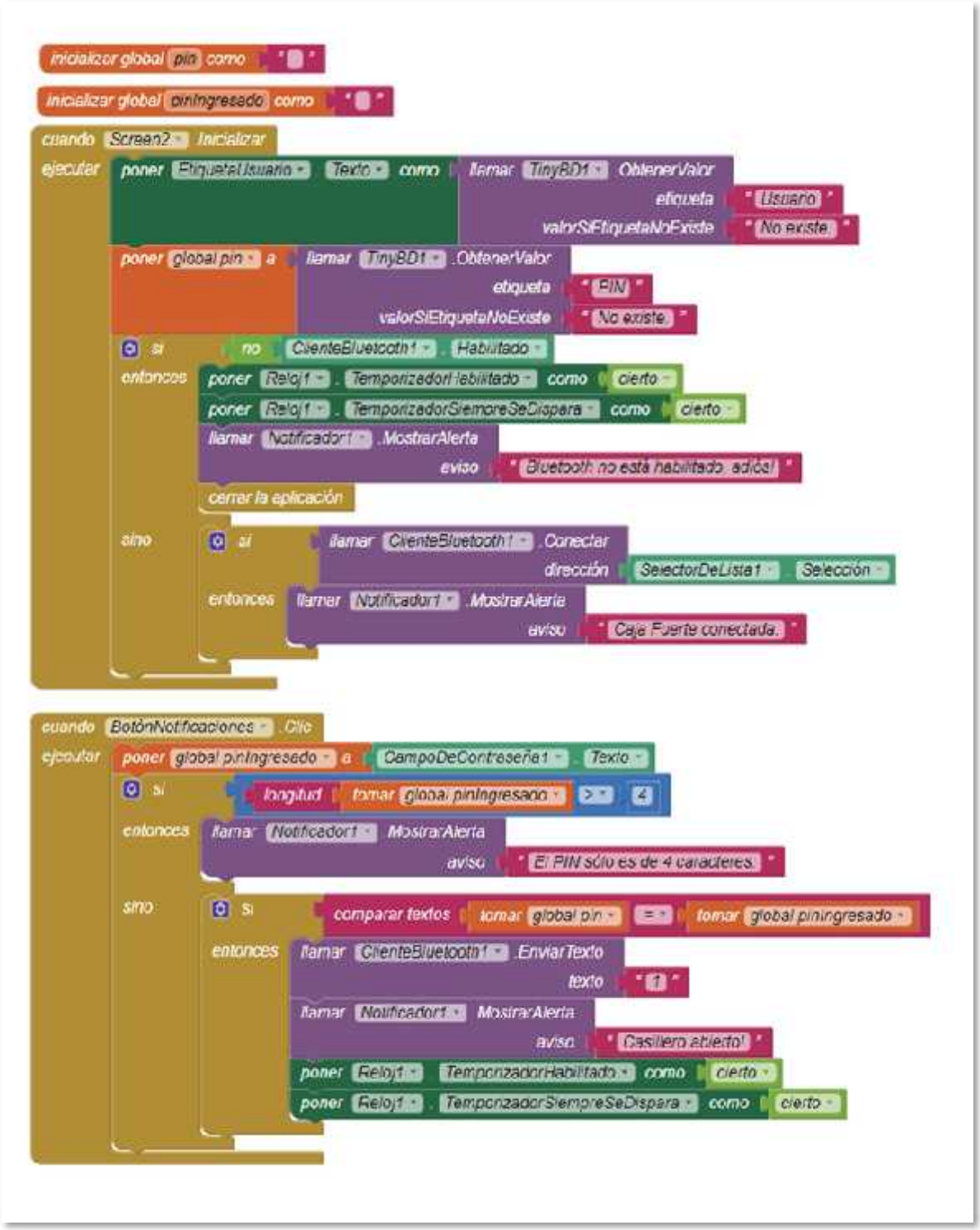


Figura 39. Programación pantalla de apertura.
Fuente: El autor.

En los siguientes bloques enlazados de programación de la app, se encuentran las configuraciones de las opciones de la pantalla de apertura y sus respectivas funciones.

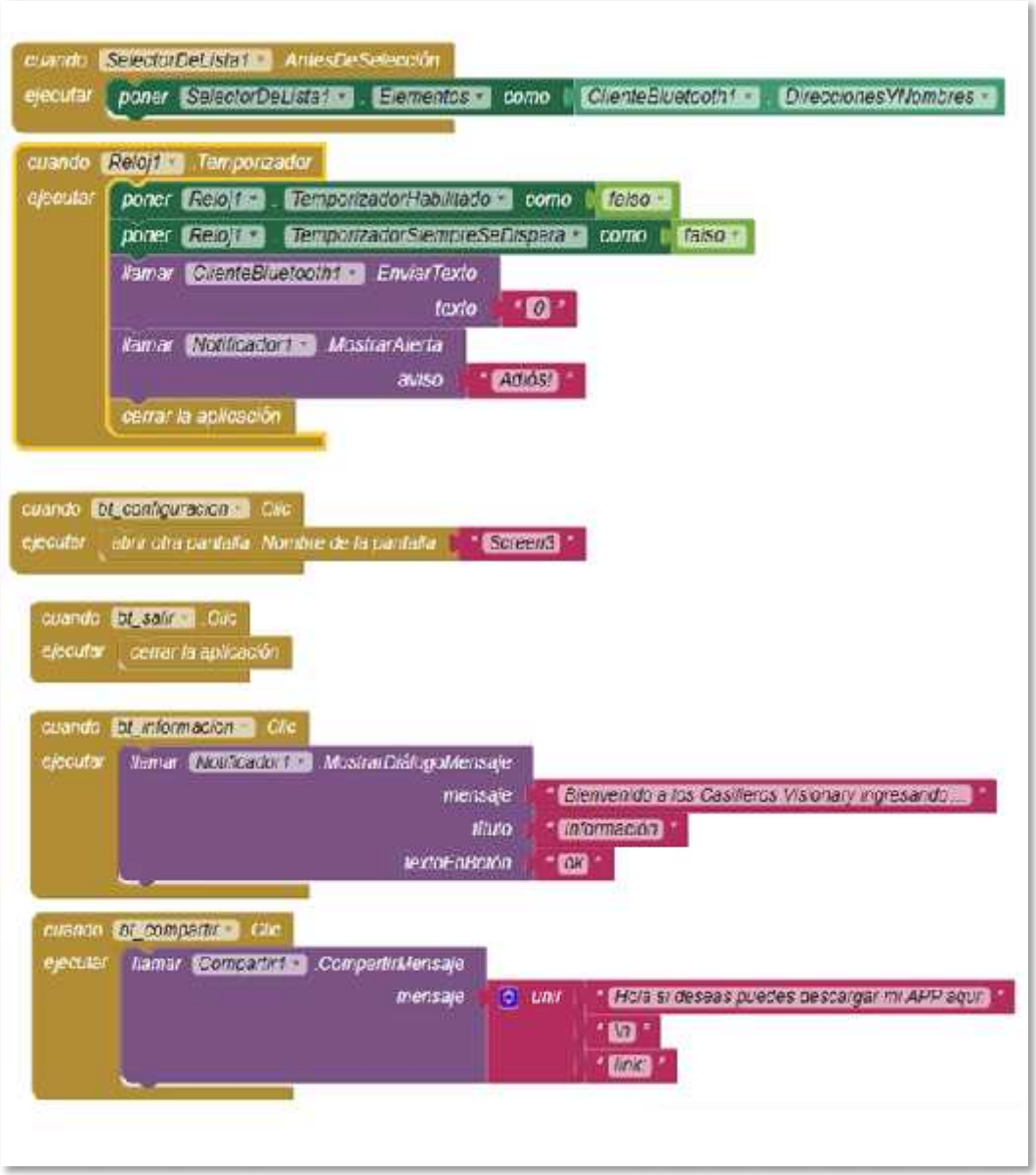


Figura 40. Pantalla de apertura programación de opciones.
Fuente: El autor.

6.3 Programa de Arduino

Para continuar, en la Figura 41, se muestra la representación de comportamiento del casillero en su apartado electrónico, mostrando en un LCD la información del usuario, además posee un receptor Bluetooth, el cual capta la orden enviada del dispositivo móvil y la pasa al microcontrolador, el cual activará la cerradura electrónica y la función del fin de carrera.

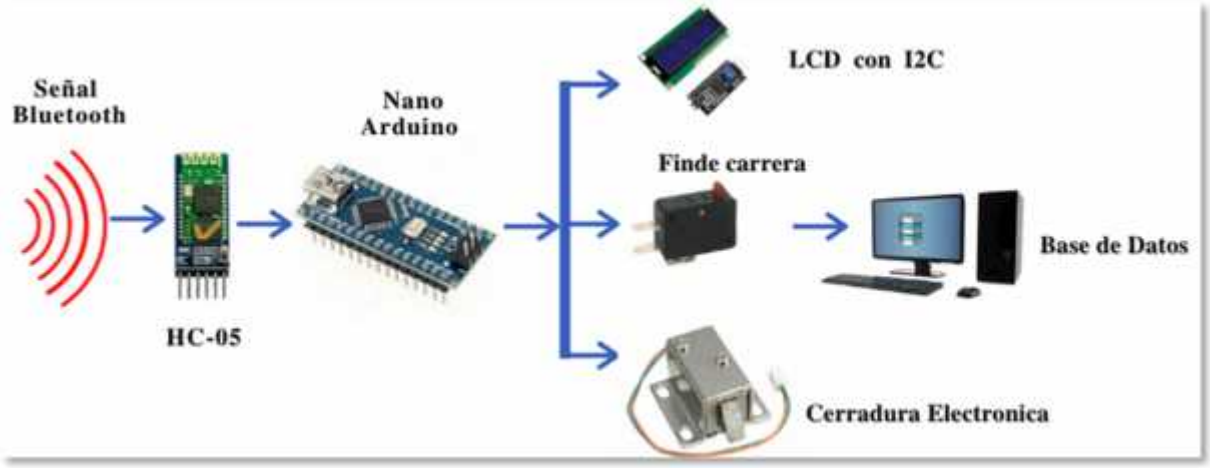


Figura 41. Esquema de comportamiento electrónico de CIV.
Fuente: El autor.

En concordancia, la Figura 42 muestra el esquema estructural que se va a utilizar para programar el microcontrolador Arduino Nano.

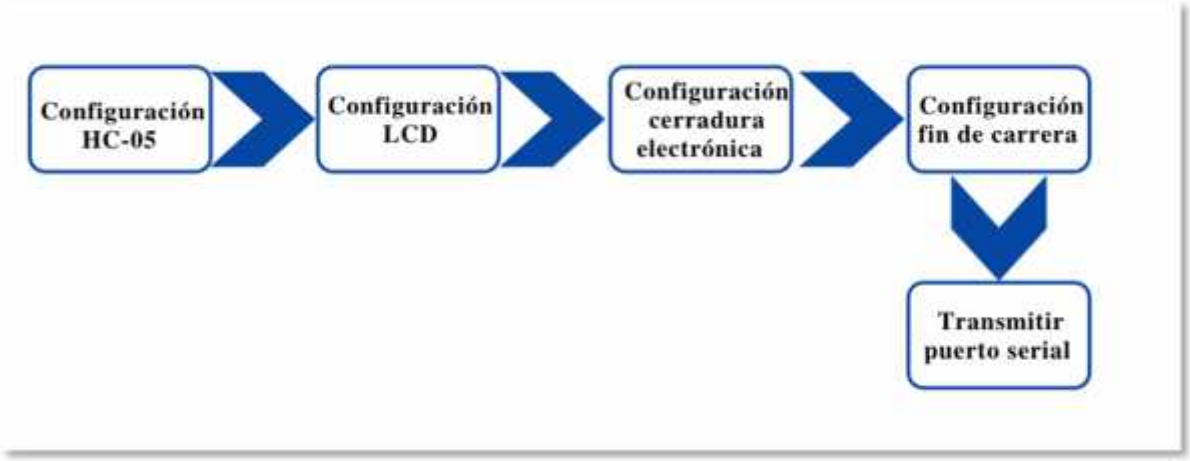


Figura 42. Esquema estructural de programa Arduino.
Fuente: El autor.

Para lograr tener una visión más general del funcionamiento del microcontrolador a continuación, se presenta el diagrama de flujo definido:

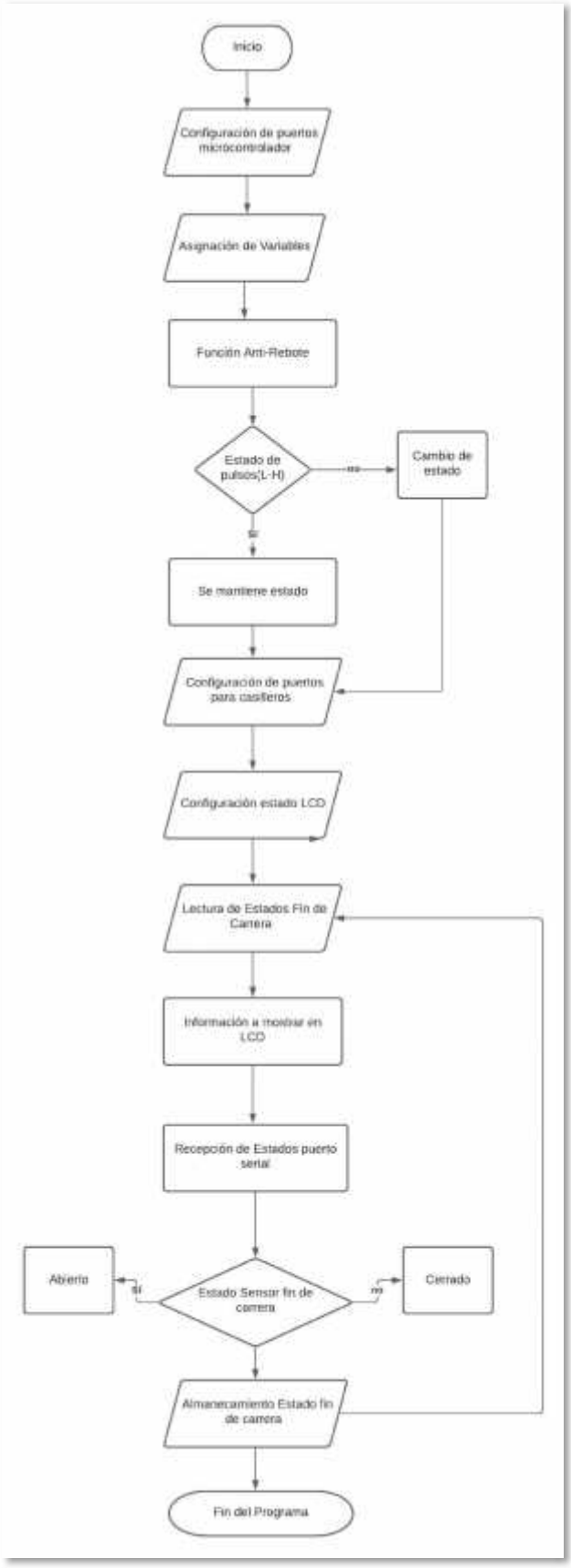


Figura 43. Diagrama de flujo programa Arduino.
Fuente: El autor.

En el programa del microcontrolador Arduino Nano para iniciar se deben añadir las diferentes librerías mediante ***#include***, las primeras en ser declaradas son las necesarias para la conexión con el LCD modelo I2C, se configuran los puertos de comunicación para el funcionamiento del Bluetooth que son los pines 10 y 11 del nano Arduino.

El pin que se va a utilizar para la conexión del fin de carrera es el ***pin 8*** del cual recibirá los datos de apertura y cierre del casillero se añaden dos variables más que son ***estadoBoton*** y ***estadobotonAnterior*** que servirán para la función de ***antirebote***, se nombra una constante de ***tiempoAntirebote*** para evitar el efecto de rebote esto quiere decir, que no se envié doble pulso a la Base de Datos, para evitar un grado de error al cambiar de un estado abierto a cerrado.

```
#include <SoftwareSerial.h>
#include <Wire.h>                  // libreria necesaria para bus I2C
#include <LiquidCrystal_I2C.h>     // libreria necesaria para display I2C

LiquidCrystal_I2C lcd(0x27, 16, 2); // configuremos el LCD en la direccion de bus I2C que es
//    0x27 y el tamaño columnas x filas 16x2
SoftwareSerial BTSerial(10, 11); // RX | TX


const int sensorPin = 8;           // boton conectado al pin 8
const int tiempoAntirebote = 10;

int estadoBoton = 1;
int estadoBotonAnterior = 1;
```

Figura 44. Inicialización variables Programa Arduino.
Fuente: El autor.

Para la función de antirebote se declara una variable ***antirebote*** de forma booleana por lo que se usa estados de falso y verdadero lo que significa si están pulsados o no; un contador que inicie en 0 y los estados anteriores cargados como falso ya que no se encuentran pulsados.

Se usa un ciclo ***do-while*** para iniciar un ciclo repetitivo (también llamados lazos o bucles permiten repetir una operación o secuencia de operaciones en función de ciertas condiciones.), en estado, se carga el pin que se esté usando para que después en una condición (***if)*** se pueda comparar el ***estado*** actual; si se cumple el contador queda en 0 y el ***estadoanterior*** se carga en ***estado***, caso contrario de no cumplirse el ***contador*** se aumentará en 1.

La condición ***while*** dice que si el contador es menor al ***tiempodeantirebote*** retorne el valor de estado una vez que no se cumpla la condición saldrá de la función.

```
/*Funcion antirebote*/
boolean antirebote (int pin) {
  int contador = 0;
  boolean estado = false;          // guarda el estado del boton
  boolean estadoAnterior = false;  // guarda el ultimo estado del boton

  do {
    estado = digitalRead (pin);
    if (estado != estadoAnterior ) { // comparamos el estado actual
      contador = 0;                  // reiniciamos el contador
      estadoAnterior = estado;
    }
    else {
      contador = contador + 1;      // aumentamos el contador en 1
    }
    delay (1);
  }
  while (contador < tiempoAntirebote);
  return estado;
}
```

Figura 45 . Función anti rebote Programa Arduino.
Fuente: El autor.

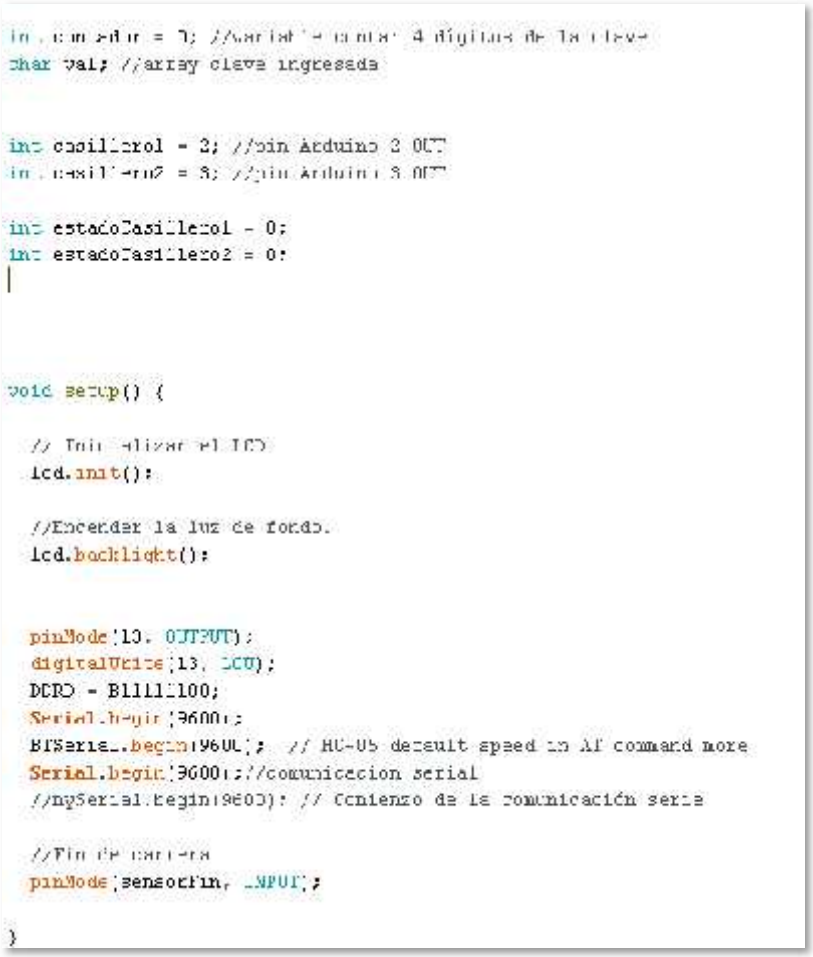
En esta parte del programa se inicia un nuevo ***contador*** el cual servirá para ingresar la clave que será recibida por el Bluetooth, la nueva variable creada será de tipo carácter ***(char)*** declarada como ***val***, la cual almacenará la clave ingresada.

Se configura los pines a ser utilizados como son el 2 y 3 del nano Arduino los cuales enviarán las pulsaciones para abrir el casillero.

En ***void setup ( )*** se añade la primera función en ser ejecutada en el nano Arduino en este espacio se setean (establecer la configuración correcta de un programa o hardware).

Se inicializa las configuraciones del lcd primero habilitando la pantalla y segundo encendiendo la luz de fondo, a continuación, se declara el pin y el estado del pin en el cual se encuentra conectado, que en este caso es el 13 del nano Arduino.

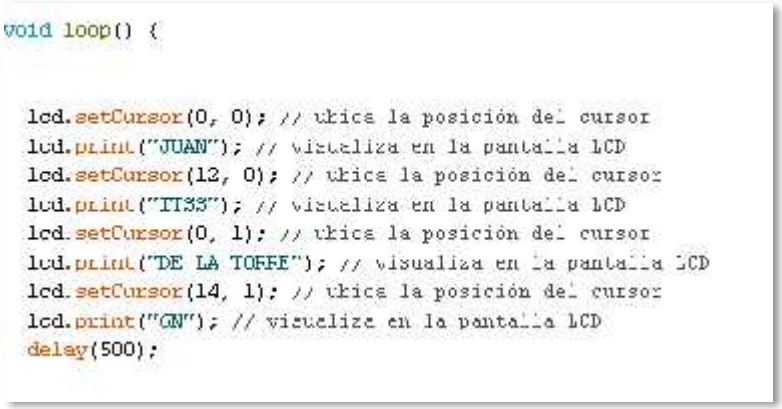
El DDR, determina si el pin es una entrada o una salida, después se inicia la comunicación serial del módulo bluetooth y se declara que ***sensorFin*** se lo vea como un ingreso de datos.



```
int contador = 7; //variable contador 4 digitos de la clave
char val; //array clave ingresada


int casillero1 = 2; //pin Arduino 2 OUT
int casillero2 = 3; //pin Arduino 3 OUT

int estadoCasillero1 = 0;
int estadoCasillero2 = 0;



void setup() {

  // Inicializar el LCD
  lcd.init();

  //Encender la luz de fondo.
  lcd.backlight();


  pinMode(13, OUTPUT);
  digitalWrite(13, LOW);
  DDRD = B11111100;
  Serial.begin(9600);
  BTSerial.begin(9600);  // HC-05 default speed in AT command more
  Serial.begin(9600);//comunicacion serial
  //mySerial.begin(9600); // Comienzo de la comunicación serie

  //Fin de carrera
  pinMode(sensorFin, INPUT);

}
```

Figura 46. Inicialización de LCD Y módulo Bluetooth.
Fuente: El autor.

En la parte del ***voip loop ( )*** se ejecuta un bucle que se repetirá un número infinito de veces, se configura el lcd desde la ubicación del cursor (posición de inicio del mensaje), lo que se quiere mostrar en la pantalla que son los datos personales del usuario.



```
void loop() {


  lcd.setCursor(0, 0); // ubica la posición del cursor
  lcd.print("JUAN"); // visualiza en la pantalla LCD
  lcd.setCursor(12, 0); // ubica la posición del cursor
  lcd.print("IIS3"); // visualiza en la pantalla LCD
  lcd.setCursor(0, 1); // ubica la posición del cursor
  lcd.print("DE LA TORRE"); // visualiza en la pantalla LCD
  lcd.setCursor(14, 1); // ubica la posición del cursor
  lcd.print("GN"); // visualize en la pantalla LCD
  delay(500);
```

Figura 47. Información que mostrará LCD.
Fuente: El autor.

Se continúa con un ciclo repetitivo ***while*** el cual primero revisará si está llegando la información desde el puerto serial, la variable ***val = BTSerial.read ( ),*** indica la información que lee del puerto serial para después iniciar un ***if*** (condición) en el caso que ***val*** reciba 0, casillero mandará un pulso negativo en el caso de que ***val*** reciba 1, casillero recibirá un pulso positivo lo que abrirá el casillero.

Se carga ***estadoboton*** con la información que se reciba de ***sensorFin*** para seguir con algunas condiciones (***if***) la primera verificará si hay un cambio respecto al estado del botón si se cumple la condición se iniciará otra condición la cual realizará la función de antirebote, enviando al puerto serial el valor de “abierto”, para ser guardado en la Base de Datos; si no se cumple esta condición se mandara al puerto serial el valor de “cerrado” que son los estados físicos del casillero.

En caso de que el estado del botón no haya cumplido la condición se volverá a realizar la función de ***antirebote*** para escribir los estados de “abierto” y “cerrado” respectivamente una vez terminado el ciclo repetitivo se guardará el ***estadoboton*** y se repetirá el bucle de *voip loop ( )*.

```
while (BTSerial.available() > 0)
{
  val = BTSerial.read();
}

if ( val == '0' )
{
  digitalWrite(casillero1, LOW);
  delay(1000);

}

if ( val == '1' )
{
  digitalWrite(casillero1, HIGH);
  delay(1000);

}

estadoBoton = digitalRead (sensorFin);            //leemos el estado del boton
if (estadoBoton  != estadoBotonAnterior) {     //si hay cambio con respecto al estado
  if (antirebote (sensorFin)) {                   //checamos  si esta preionado
    Serial.write("abierto");
    Serial.println("");
    delay(300);
  }
  else {
    Serial.write("cerrado");
    Serial.println("");
    delay(300);
  }
}
else {
  if (antirebote (sensorFin)) {                   //checamos  si esta preionado
    Serial.write("abierto");
    Serial.println("");
    delay(300);
  }
  else {
    Serial.write("cerrado");
    Serial.println("");
    delay(300);
  }
}
estadoBotonAnterior = estadoBoton;      // guardamos el estado del boton
}
```

Figura 48. Información recibida del Microcontrolador.
Fuente: El autor.

6.4 Programa de Registro de Actividades - Base de Datos

La Figura 49 se muestra la representación de comportamiento del programa *administrador de información*, que es el programa que registra las actividades de acceso por parte de los usuarios a los casilleros en una Base de Datos MySQL.

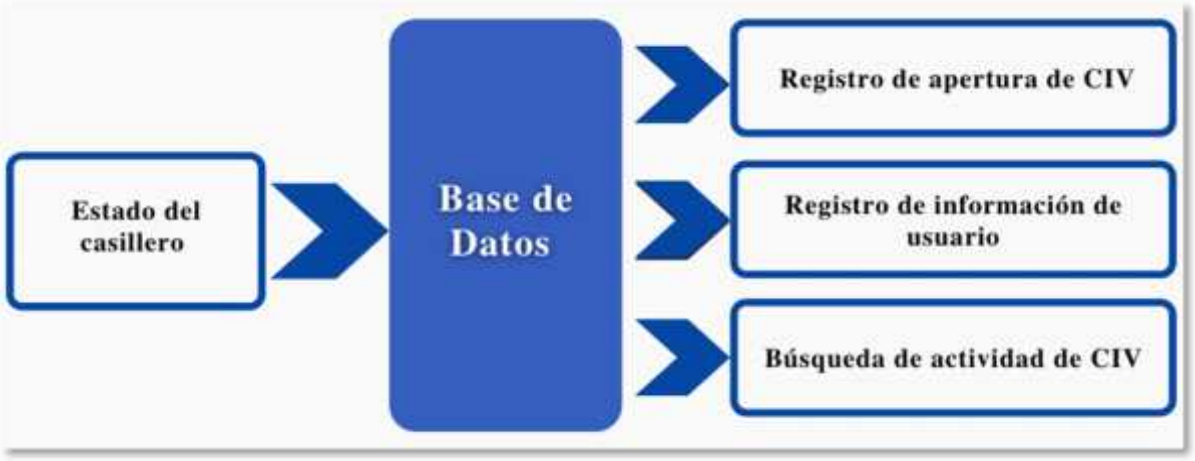


Figura 49. Esquema de comportamiento programa de registro de la Base de Datos.
Fuente: El autor.

El programa se lo desarrolla en el entorno de desarrollo integrado libre NetBeans, su esquema estructural se muestra en la siguiente Figura 50.

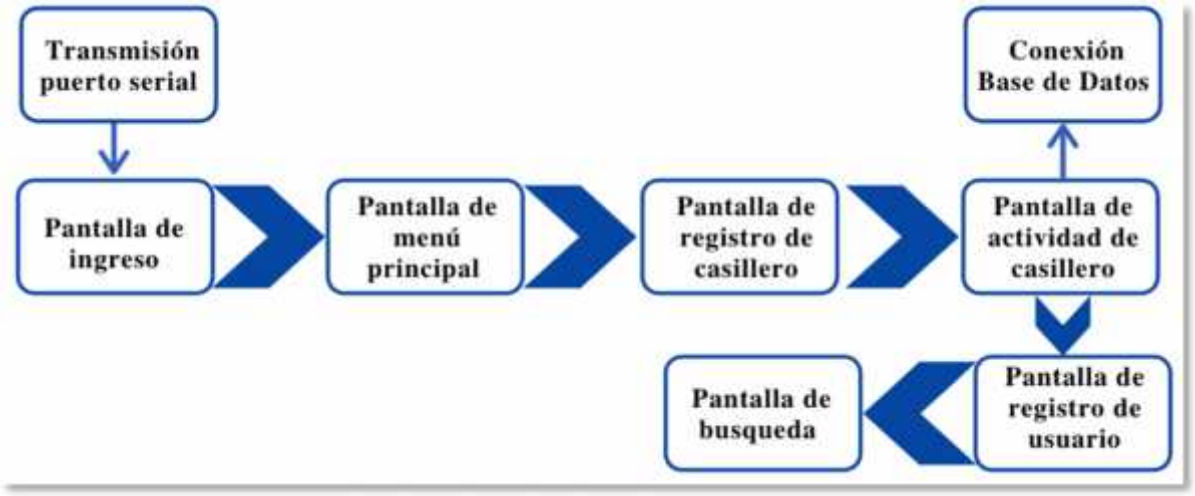


Figura 50. Esquema estructural programa de registro de la Base de Datos.
Fuente: El autor.

Para iniciar se necesita cargar todas las librerías que se va a utilizar tanto para la conexión con el microcontrolador Arduino como con la Base de Datos MySQL.

Figura 51. Estructura Programa NetBeans.
Fuente: El autor.

6.4.1 Conexión con Arduino

Usando una plantilla de Netbeans, se inicia Arduino con las configuraciones principales, en esta parte se declarará el puerto serial en el que está conectado el microcontrolador y los tiempos para sincronizarse como se muestra en la Figura 52.

```
public class arduino {
    //Variables de conexion
    private OutputStream output = null;
    SerialPort serialPort;
    private final String porta="COM3";
    private static final int timeOut = 2000; //2 segundos
    private static final int dataRate = 9600;
```

Figura 52. Conexión puerto Arduino.
Fuente: El autor.

En esta plantilla el programa verificará si hay conexión con el puerto que antes se configuró, en caso de no recibir respuesta mostrará un error “NO SE PUEDE CONECTAR CON EL PUERTO” en la consola de programación.

Figura 53. Verificación de conexión con Arduino.
Fuente: El autor.

Después el puerto serial entregará información y sincronizará los tiempos que configuró en la plantilla anterior.

Para terminar, se programarán mensajes de posibles errores en el envío de datos y conexión.

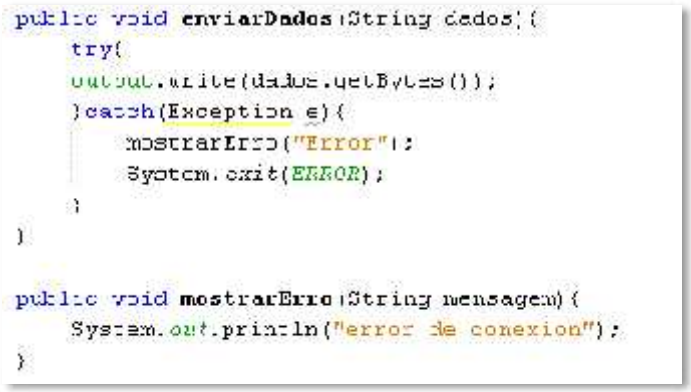


Figura 54. Mensaje de error de conexión con Arduino.
Fuente: El autor.

6.4.2 Pantalla del Sistema Casilleros Inteligentes Visionary (CIV)

La pantalla de inicio del sistema pedirá el ingreso de un usuario y la contraseña de administrador las mismas que serán entregadas al momento de instalar el programa, para poder continuar a la pantalla del menú principal.

Figura 55. Pantalla de ingreso Interfaz Gráfica.
Fuente: El autor.

La programación para esta pantalla se muestra a continuación, iniciando con una plantilla la cual se la nombrará **pantalla sistema** en la que se configurará el tamaño tanto como su altura y su ancho además de la posición en pantalla.

```
public pantallasistema() {
    super("pantallasistema");

  Dimension pantalla = Toolkit.getDefaultToolkit().getScreenSize();
  int height = pantalla.height;
  int width = pantalla.width;
  setSize(width/2, height/2);

  setLocationRelativeTo(null);
  setVisible(true);
    initComponents();
}

/**
 * This method is called from within the constructor to initialize the form.
 * WARNING: DO NOT modify this code. The content of this method is always
 * regenerated by the Form Editor.
 */
@SuppressWarnings("unchecked")
Generated Code
```

Figura 56. Dimensiones de pantalla de ingreso.
Fuente: El autor.

A continuación, se muestra en la Figura 55 la programación que tendrá el botón **entrar**, se declara como ***string*** (cadena de caracteres) tanto el usuario y la contraseña que son requeridos para continuar.

El ***string pass*** se comparará con la contraseña que se ingresa mediante una condición, si es igual, enviará a la pantalla del **menú principal**.

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    String usuario = "usuario";
    String contrasenia = "1234";

    String pass=new String(password.getPassword());

    if (txtusuario.getText().equals(usuario) && pass.equals(contrasenia)){
        Registro panta=new Registro();
        panta.setVisible(true);
        dispose();
    }
    else
    {
        JOptionPane.showMessageDialog(this,"Usuario o contraseña incorrectos");
    }
```

Figura 57. Programación de botón Entrar.
Fuente: El autor.

Caso contrario saldrá un mensaje “Usuario o contraseña incorrectos”.

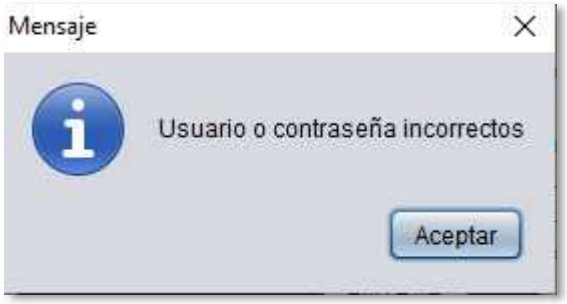


Figura 58. Pantalla de inicio mensaje de datos erróneos.
Fuente: El autor.

6.4.3 Pantalla de Menú Principal

En esta pantalla se programa lo que hace cada botón es la navegación entre las diferentes opciones como son los Registros de usuario y Registro de casilleros, la actividad en tiempo real del casillero y la búsqueda en caso que se desee saber sobre la actividad del casillero.

Figura 59. Pantalla Menú Principal Programa Netbeans.
Fuente: El autor.

A continuación, en la Figura 60, se muestra la programación que tendrá cada uno de los botones y hacía que pantallas se dirigirá.

```
private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
    Pantalla4 panta=new Pantalla4();
        panta.setVisible(true);
}

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
        Window panta=new Window();
        panta.setVisible(true);
}

private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {
        System.exit(0);
}

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        Propiedad panta=new Propiedad();
        panta.setVisible(true);
        dispose();
}

private void jButton5ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}
```

Figura 60. Programación de botones de navegación Menú Principal.
Fuente: El autor.

6.4.4 Registro de casillero

En esta parte del programa el propietario del sistema o encargado añadirá, en este caso, al estudiante con su información general y el número de casillero que le corresponda.

Figura 61. Pantalla Registro de Casillero Programa NetBeans.
Fuente: El autor.

En el programa se inicia configurando las dimensiones de pantalla para ser visibles correctamente.

```
public class Propiedad extends javax.swing.JFrame {

    /**
     * Creates new form Propiedad
     */
    public Propiedad() {
        super("Propiedad");

        Dimension pantalla = Toolkit.getDefaultToolkit().getScreenSize();
        int height = pantalla.height;
        int width = pantalla.width;
        setSize(width/2, height/2);

        setLocationRelativeTo(null);
        setVisible(true);
        initComponents();
    }
```

Figura 62. Dimensiones de pantalla de Registro de Casillero.
Fuente: El autor.

En esta plantilla se configura las acciones que realizará el botón *Guardar*. Primero se iniciarán las variables que se va a necesitar en formato *string* (cadena de caracteres), para continuar se guardarán los datos ingresados en los espacios en blanco que fueron llenados por el administrador, como se muestra en la Figura 63.

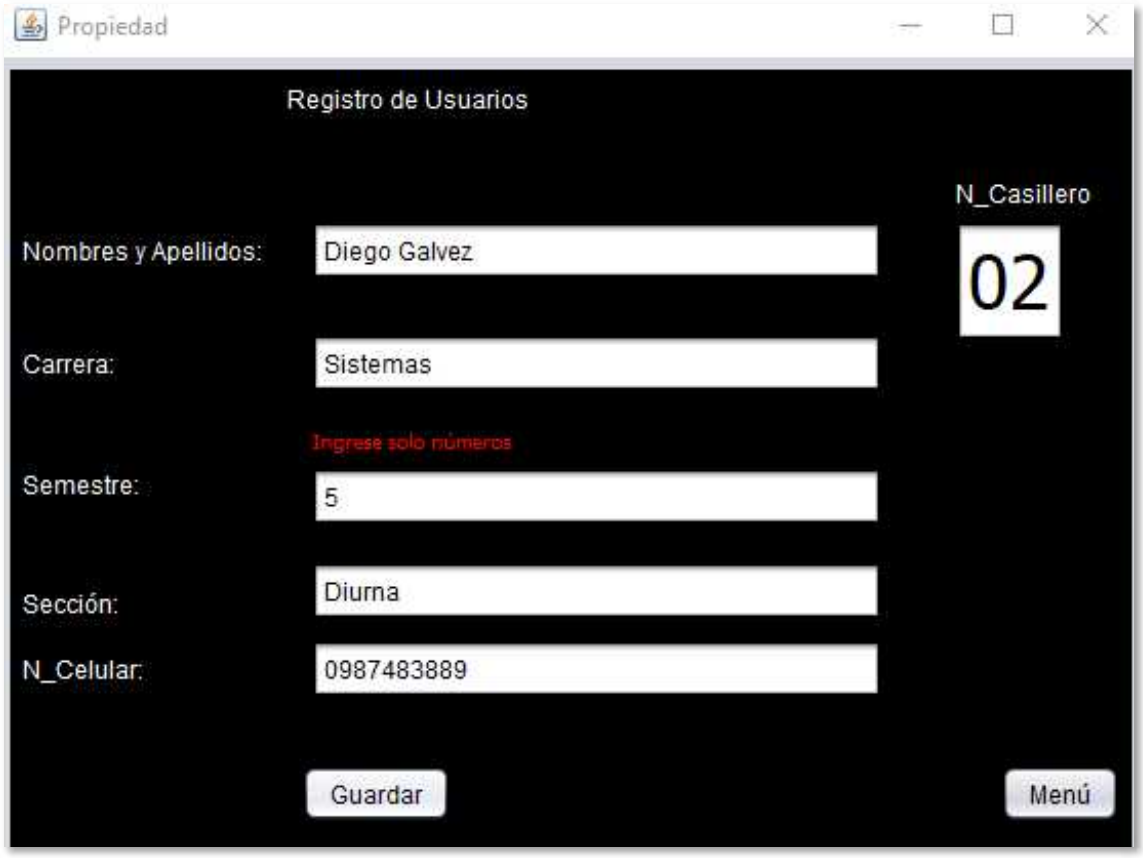


Figura 63. Pantalla registro de casillero información de Usuario.
Fuente: El autor.

Se cargarán estos datos en las variables anteriormente declaradas; con *SQL=* se crea un comando de comunicación con la Base de Datos que gracias a las librerías que se han instalado en un principio se podrán conectar como se muestra en la Figura 64.

Figura 64. Programación del botón Guardar en la Pantalla de registro de casillero.
Fuente: El autor.

A continuación, se llenará la tabla *datos_usuarios* en la Base de Datos con la información que se ha ingresado manualmente.

Cod	Nombres	Carrera	Semestre	Seccion	N_Celular
1	Juan De la Torre	Sistemas	6	diurna	0987183889
2	Diego Galvez	Sistemas	5	Diurna	0987483889

Figura 65. Tabla *datos_usuarios* Base de Datos.
Fuente: El autor.

Una vez se pulse el botón *Guardar* si todo está bien lanzará un mensaje" Registro Guardado con éxito".

Figura 66. Pantalla de registro de casillero mensaje de Guardado.
Fuente: El autor.

La programación del botón *Menú* regresará a la pantalla anterior la de menú principal.

```
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
        Registro panta=new Registro();
        panta.setVisible(true);
        dispose();// TODO add your handling code here:
```

Figura 67. Programación del botón Menú en la Pantalla de registro de casillero.
Fuente: El autor.

6.4.5 Actividad de casillero

La pantalla de registro de actividad de casillero indica cuando se esté usando, el momento en que se abre o se cierra con su hora respectiva.

Figura 68. Pantalla de Actividad de Casillero ventana de estado.
Fuente: El autor.

Para iniciar con la programación se declara cada una de las variables que se va a usar tanto como las de conexión y los diferentes datos que se va a tomar del Arduino y del sistema del computador como es el calendario.

```
public class Window extends javax.swing.JFrame {

    Conectar cc=new Conectar();
    Connection reg = cc.getConnection();
    int Slot=1;
    String Output = "";
    String Fin_correo= "";
    int Lectura=-1;
    String date="";
    int ga=0;
    int gs=0;
    boolean cambio;
    String cerrado="";
    DefaultTableModel Modelo;
    Calendar Calendario = Calendar.getInstance();
    //Se crea una instancia de la libreria PanamaHitek_Arduino
    private PanamaHitek_Arduino ino = new PanamaHitek_Arduino();
```

Figura 69. Programación de Pantalla Actividad de Casilleros.
Fuente: El autor.

En esta plantilla se puede recibir los datos que son enviados del puerto serial; los datos son recibidos del programa de Arduino; se realiza un *if* (condición) en la cual, si se cumple que el dato es igual a “abierto”, la variable ***cambio*** se cargue con un valor de ***false*** y la variable ***ga*** que es casillero abierto sea igual a 0, caso contrario la variable **ga** se aumentará en 1 y su estado cambiará a ***true*** dando otra condición, en el caso de que sea 1 se guardará.

```
private SerialPortEventListener listener = new SerialPortEventListener() {
    @Override
    public void serialEvent(SerialPortEvent spe) {
        try {
            /**
             * En esta instrucción se evalua constantemente el puerto serie
             * hasta encontrar un mensaje disponible para ser impreso
             *
             * Se considera que un mensaje ha sido recibido en su totalidad
             * cuando se reciben los bytes 13 y 10, los cuales son enviados
             * por Arduino cuando se utiliza Serial.println().
             *
             */

            if (in..isMessageAvailable()) {
                //System.out.println(in.printMessage()+" prueba");
                dato=in.printMessage();
                //Se imprime el mensaje recibido en la consola
                //jLabelOutput.setText("Resultado: " + in.printMessage()); //Para borrar

                if(dato.equals("abierto")){
                //llamar metodo guardar a la BD.
                cambio=false;
                ga=0;
               }
                else{
                    ga++;
                    cambio=true;
                    if (ga==1)
                        guardar();

                }
```

Figura 70. Programación de Recepción de Información Pantalla de Actividad de Casilleros.
Fuente: El autor.

Se inicia otro condicional en la cual se compara el estado de comunicación con el Arduino, si se encuentra ***gc=0*** se guarde con un estado **true** y se compara de nuevo si es igual a 1 entonces se guarda.

Se utilizarán condicionales para los ***slots*** que significan la cantidad de casilleros y puertos que se usan se dará la lectura del Arduino las veces que se ha leído el dato.

```
                if(dato.equals("cerrado")){
                //llamar metodo guardar a la BD.
                cambio=false;
                gc=0;
               }
                else{
                    gc++;
                    cambio=true;
                    if (gc==1)
                       guardar();

                }


                if (Slot==1){

                    if (Lecturas>=1){
                        Slot=2;
                        Lecturas++;
                        //Fin_carrera="5"+dato;
                    }

                }
                    else if (Slot==2){

                        Slot=1;
                        Lecturas++;
                        Fin_carrera=dato;
                        TableUpdate();
                    }

            }
```

Figura 71. Programa lectura de dato Arduino.
Fuente: El autor.

Con la variable ***Fin carrera*** se cargará el valor del dato enviado por el microcontrolador Arduino y se actualizará la tabla ***datos_casilleros*** en la Base de Datos.

Cod	Nombres	Fecha	Hora	Dato
1	Casillero	2019/07/08	10:30:57	
1	Casillero	2019/07/08	10:31:04	cerrado
1	Casillero	2019/07/08	10:31:11	abierto
1	Casillero	2019/07/08	10:31:24	cerrado
1	Casillero	2019/07/08	10:32:39	
1	Casillero	2019/07/08	10:32:42	cerrado
1	Casillero	2019/07/08	10:32:51	abierto
1	Casillero	2019/07/08	10:32:59	cerrado
1	Casillero	2019/07/08	10:33:07	abierto
1	Casillero	2020/10/01	17:58:38	cerrado
1	Casillero	2020/10/01	17:58:51	abierto
1	Casillero	2020/10/01	17:59:00	cerrado
1	Casillero	2020/10/01	18:00:05	cerrado
1	Casillero	2020/10/01	18:00:15	abierto
1	Casillero	2020/10/01	18:00:16	cerrado
1	Casillero	2020/10/01	18:00:19	abierto
1	Casillero	2020/10/01	18:00:20	cerrado
1	Casillero	2020/10/02	09:42:17	cerrado
1	Casillero	2020/10/02	11:13:15	cerrado

Figura 72. Tabla *datos_casillero* en la Base de Datos.
Fuente: El autor.

En esta parte del programa se toma mediante la variable ***calendario*** la información de tiempo del sistema tanto como la hora, los minutos y los segundos.

Se guarda en una variable de salida para posteriormente ser usado.

```
public void TableUpdate(){
    |

    String hora = Calendario.get(Calendar.HOUR_OF_DAY)+"";
    String minuto = Calendario.get(Calendar.MINUTE)+"";
    String segundos = Calendario.get(Calendar.SECOND)+"";
    if (Integer.parseInt(hora)<10){
    hora = "0" + hora;
    }
    if (Integer.parseInt(minuto)<10){
    minuto = "0" + minuto;
    }
    if (Integer.parseInt(segundos)<10){
    segundos = "0" + segundos;
    }
    Output=hora+":"+minuto+":"+segundos;
    Calendar fin=Calendar.getInstance();
    Modelo.addRow(new Object[]{Output,Fin_carrera});

}
```

Figura 73 Programación Pantalla de Actividad de Casilleros Tiempo del Sistema.
Fuente: El autor.

En la plantilla ***Window*** se carga las dimensiones de la pantalla para poder ser visualizada correctamente, adicional se declara el puerto y la función que realizará el Arduino.

```
public Window() {
    super("Window");

  Dimension pantalla = Toolkit.getDefaultToolkit().getScreenSize();
  int height = pantalla.height;
  int width = pantalla.width;
  setSize(width/2, height/2);

  setLocationRelativeTo(null);
  setVisible(true);
    initComponents();
    Modelo = (DefaultTableModel) jTable1.getModel();
    /*
     * Se inicia la comunicación con el Puerto Serie utilizando la función RXTX,
     * la cual es permite tanto enviar como recibir datos en el Puerto Serie
     */
    try {
        ino.arduinoRXTX("COM5", 9600, listener);
    } catch (ArduinoException ex) {
        Logger.getLogger(Window.class.getName()).log(Level.SEVERE, null, ex);
}

 }
```

Figura 74. Dimensiones de la Pantalla Actividad de Casilleros.
Fuente: El autor.

En la plantilla de guardar primero, se inicia las variables que se van a necesitar para cargar en la Base de Datos. Se inicia cargando los datos de calendario tanto como el año, el mes, el día; a continuación, se carga la información del tiempo de la misma variable hora, minuto y segundo y se guarda en una variable de salida ***Output***.

```
public void guardar (){


   String Fecha="";
   int Cod=0;
   //int N_activaciones=1;

   String Nombres="";
   String Dato="";


   String sql;
   String dia = Calendario.get(Calendar.DATE)+"";
   String mes = Calendario.get(Calendar.MONTH)+"";
   String año = Calendario.get(Calendar.YEAR)+"";
   if (Integer.parseInt(dia)<10){
    dia = "0" + dia;
    }
    if (Integer.parseInt(mes)<10){
    mes = "0" + mes;
    }

    Fecha= año+"/"+ mes+"/"+dia;

    String hora = Calendario.get(Calendar.HOUR_OF_DAY)+"";
    String minuto = Calendario.get(Calendar.MINUTE)+"";
    String segundos = Calendario.get(Calendar.SECOND)+"";
    if (Integer.parseInt(hora)<10){
    hora = "0" + hora;
    }
    if (Integer.parseInt(minuto)<10){
    minuto = "0" + minuto;
    }
    if (Integer.parseInt(segundos)<10){
    segundos = "0" + segundos;
    }
    Output=hora+":"+minuto+":"+segundos;
```

Figura 75. Programación función guardar en Pantalla de actividad de casillero.
Fuente: El autor.

Con la función ***switch*** (es una manera de tomar una decisión a partir de un valor dado, con varios resultados posibles) se carga la variable ***case*** la cual se usa para reconocer el casillero.

```
switch (Slot){
    case 1:
        Cod = 1;
        Nombres="Casillero ";
       //N_activaciones++;
        Dato=Fin_carrera;
        break;
    case 2:
        Cod = 1;
        Nombres="Casillero ";
        //N_activaciones++;
        Dato=Fin_carrera;
        break;
```

Figura 76. Programación reconocimiento de casillero.
Fuente: El autor.

SQL permitirá ir cargando en la tabla **datos_casilleros** la información que se ha recibido del Arduino y de java para ser actualizado cada uno de los registrados.

```
    }
    sql="INSERT INTO datos_casilleros (Cod, Nombres, Fecha, Hora, Dato) values (?,?,?,?,?)";
        try {
            PreparedStatement pst=reg.prepareStatement(sql);
            pst.setInt(1,Cod);
            pst.setString(2,Nombres);
            //pst.setInt(3,N_activaciones);
            pst.setString(3,Fecha);
            pst.setString(4,Output);
            pst.setString(5,dato);

            int n = pst.executeUpdate();
            /*if (n>0){
                JOptionPane.showMessageDialog(null, "Registro Guardado con exito");
                N_activaciones=0;
            }*/

        } catch (SQLException ex) {
            Logger.getLogger(Pantalla4.class.getName()).log(Level.SEVERE, null, ex);
        }

}
```

Figura 77. Programa envió de información a tabla *datos_casillero* en la Base de Datos.
Fuente: El autor.

Para terminar, se configura el botón búsqueda el cual enviará a la pantalla de búsqueda.

```
private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
    Pantalla4 panta=new Pantalla4();
        panta.setVisible(true);
        //dispose();
}
```

Figura 78. Programación botón Búsqueda.
Fuente: El autor.

6.4.6 Registro de usuario

Esta pantalla muestra la información de los usuarios ingresados.

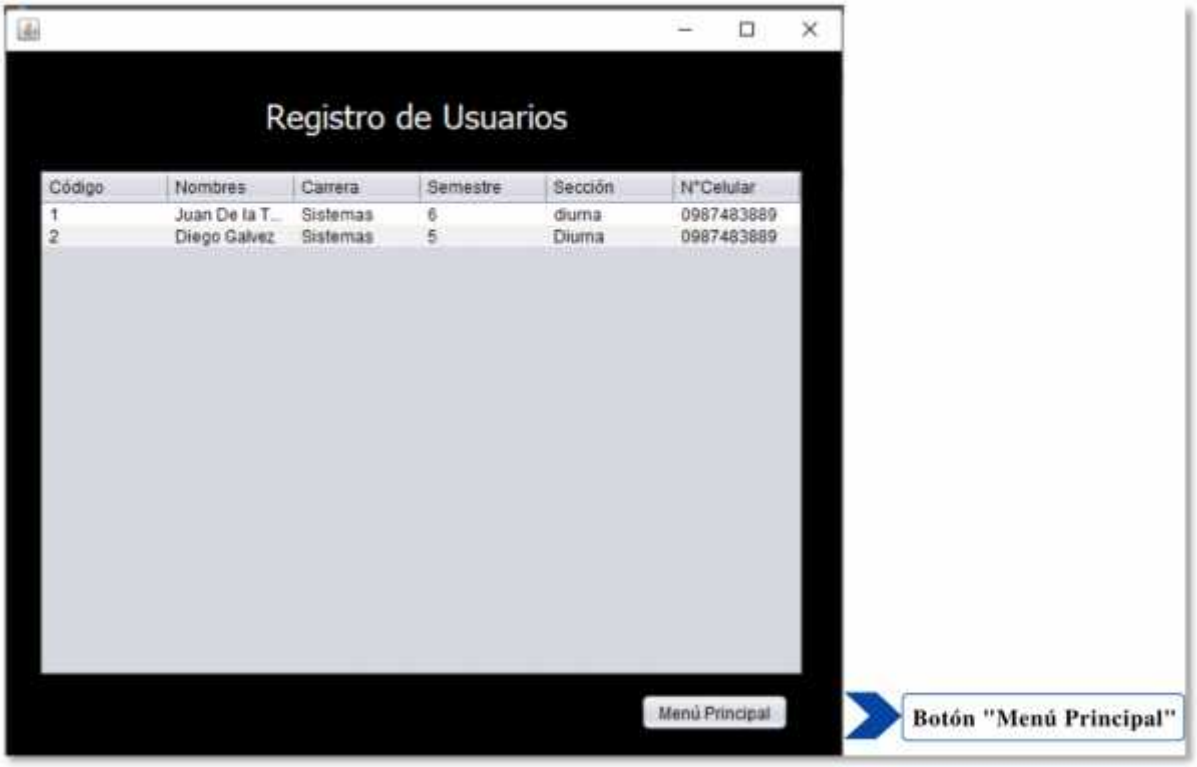


Figura 79. Pantalla de Registro de Usuarios.
Fuente: El autor.

A continuación, mediante una función ***mostrar Tabla*** se procede a crear una tabla que será mostrada en la ventana la cual tendrá los datos del usuario que fueron tomados desde la tabla ***datos_usuarios*** en la Base de Datos.

```
    public Registro() {
        initComponents();
        mostrarTabla();
    }
    public void mostrarTabla(){
        Conectar cc=new Conectar();
        Connection reg = cc.getConnection();
        DefaultTableModel mod = new DefaultTableModel();

        int codigo=0;
        mod.addColumn("Código");
        mod.addColumn("Nombres");
        //mod.addColumn("activaciones");
        mod.addColumn("Carrera");
        mod.addColumn("Semestre");
        mod.addColumn("Sección");
        mod.addColumn("N°Celular");
        tableuser.setModel(mod);

        //String sql="SELECT * FROM datos_sensores WHERE Cod="+codigo;

        String sql="SELECT Cod,Nombres,Carrera,Semestre,Seccion,N_Celular FROM datos_usuarios" ;

        String dat[]=new String[6];

        Statement st;
     try {
         st = reg.createStatement();
         ResultSet rs=st.executeQuery(sql);
         while (rs.next()){
             dat[0]=rs.getString(1);
             dat[1]=rs.getString(2);
             dat[2]=rs.getString(3);
             dat[3]=rs.getString(4);
             dat[4]=rs.getString(5);
             dat[5]=rs.getString(6);
             mod.addRow(dat);
         }
     tableuser.setModel(mod);
 } catch (SQLException ex) {
     Logger.getLogger(Pantalla4.class.getName()).log(Level.SEVERE, null, ex);
 }
}
```

Figura 80. Programación de pantalla de Registro de Usuarios.
Fuente: El autor.

Una vez obtenidos los datos se creará una tabla para mostrar la información de la Base de Datos.

Código	Nombres	Carrera	Semestre	Sección	N°Celular
1	Juan De la T...	Sistemas	6	diurna	0987483889
2	Diego Galvez	Sistemas	5	Diurna	0987483889

Figura 81. Tabla información de usuarios.
Fuente: El autor.

Para finalizar se programa el botón que enviará a la pantalla del menú principal.

```
    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
            Registro pantalla=new Registro();
            panta.setVisible(true);
            dispose();
// TODO add your handling code here:
    }
```

Figura 82. Programación botón Menú Principal.
Fuente: El autor.

6.4.7 Pantalla de búsqueda

En esta pantalla se busca la actividad del casillero, los momentos en que fue abierto y cerrado.

Figura 83. Pantalla de Búsqueda de Actividad de Casilleros.
Fuente: El autor.

Para iniciar la programación se debe usar la función del calendario para tomar los datos de tiempo del sistema, en la plantilla se configurará las dimensiones que tendrá la pantalla.

```
public class Pantalla4 extends javax.swing.JFrame {
 Calendar Calendario = Calendar.getInstance();
    /**
     * Creates new form Pantalla4
     */
    Conectar con;
    public Pantalla4() {
        super("Pantalla4");

      Dimension pantalla = Toolkit.getDefaultToolkit().getScreenSize();
      int height = pantalla.height;
      int width = pantalla.width;
      setSize(width/2, height/2);

      setLocationRelativeTo(null);
      setVisible(true);
        initComponents();

    }
```

Figura 84. Programación registro de tiempo y tamaño de Pantalla.
Fuente: El autor.

A continuación, se procede a crear una tabla la cual mostrará en la pantalla de Búsqueda los estados e información del casillero que fueron tomados desde la tabla ***datos_casillero*** en la Base de Datos.

```
    public void mostrarTabla(){
        Conectar cc=new Conectar();
        Connection reg = cc.getConnection();
        DefaultTableModel mod = new DefaultTableModel();

        int codigo=0;
        String fecha="";
        fecha=campo_fecha.getText();
        System.out.println(fecha);
        //int codigo=Integer.parseInt(campo_codigo.getText());
        String cod1=campo_codigo.getText() ;
        codigo=Integer.parseInt(cod1);
        mod.addColumn("codigo");
        mod.addColumn("nombres");
        //mod.addColumn("Activaciones");
        mod.addColumn("fecha");
        mod.addColumn("hora");
        mod.addColumn("estado");
        tabladatos.setModel(mod);
```

Figura 85. Programación creación de tabla en pantalla de Búsqueda.
Fuente: El autor.

Al momento de realizar la conexión se declara la variable que se va a utilizar y lo que se necesita tomar de la Base de Datos mediante SQL se realiza una consulta como se muestra a continuación:

String sql="SELECT c. Cod, u. Nombres, c. Fecha, c. Hora, c. Dato FROM datos_casilleros c, datos_usuarios u WHERE c. Fecha=fecha AND c. Cod="+codigo+" AND c. Cod=u.Cod";

La cual seleccionará la información que se quiere mostrar de la Base de Datos y el lugar en el que se encuentran registradas.

```
    String dat[]=new String[5];

    Statement st;
try {
    st = reg.createStatement();
    ResultSet rs=st.executeQuery(sql);
    while (rs.next()){
        dat[0]=rs.getString(1);
        dat[1]=rs.getString(2);
        dat[2]=rs.getString(3);
        dat[3]=rs.getString(4);
        dat[4]=rs.getString(5);
        //dat[5]=rs.getString(6);
        mod.addRow(dat);
    }
    tabladatos.setModel(mod);
} catch (SQLException ex) {
    Logger.getLogger(Pantalla4.class.getName()).log(Level.SEVERE, null, ex);
}
}
```

Figura 86. Programación vinculó a Base de Datos.
Fuente: El autor.

Una vez llenado los campos de texto con lo que se quiere buscar en el programa se pulsa el botón **Consultar y** se obtienen los datos para ser visualizados en la Pantalla de Búsqueda.

Figura 87. Pantalla de Búsqueda consulta casillero.
Fuente: El autor.

Para finalizar se programa el botón menú principal el cual direccionará a la pantalla del mismo nombre.

```
private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
    Registro panta=new Registro();
    panta.setVisible(true);
    dispose();
}
```

Figura 88. Programación botón Menú Principal.
Fuente: El autor.

6.5 Base de Datos

Para la Base de Datos se usa phpMyadmin (es una herramienta escrita en lenguaje de programación PHP con la intención de manejar la administración de MySQL Base de Datos a través de páginas web) para administrarla la cual se puede utilizar mediante la instalación de XAMPP (es un paquete de software libre, que consiste principalmente en el sistema de gestión de bases de datos MySQL).

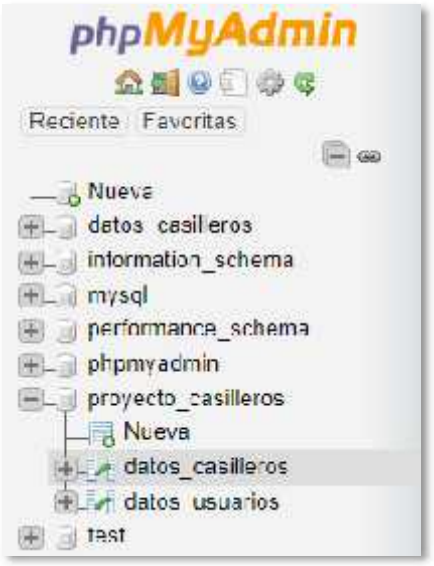


Figura 89. Interfaz phpMyAdmin.
Fuente: El autor.

Se realiza la creación de la Base de Datos, en este caso: ***proyecto_casilleros.***

Una vez creada se añaden dos tablas declarando los nombres y el tipo de dato que van a almacenar como se muestra en la Figura 90.

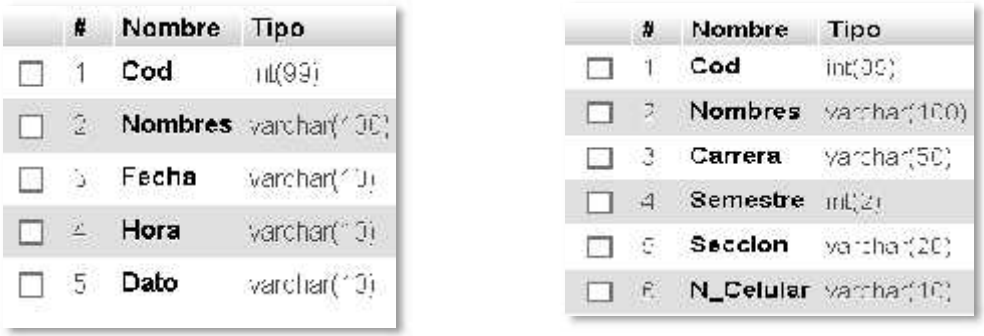


Figura 90. Estructura de Tablas de Base de Datos.
Fuente: El autor.

De esta manera toda la información que se añade del programa en NetBeans (es un programa que sirve como IDE (un **entorno de desarrollo integrado**)) será cargado y almacenado en estas tablas.

En la Figura 91 se muestra la información almacenada que fue tomada de la Pantalla de Registro de actividad del casillero.

Cod	Nombres	Fecha	Hora	Dato
1	Casillero	2019/07/08	10:30:57	
1	Casillero	2019/07/08	10:31:04	cerrado
1	Casillero	2019/07/08	10:31:11	abierto
1	Casillero	2019/07/08	10:31:24	cerrado
1	Casillero	2019/07/08	10:32:39	
1	Casillero	2019/07/08	10:32:42	cerrado
1	Casillero	2019/07/08	10:32:51	abierto
1	Casillero	2019/07/08	10:32:59	cerrado
1	Casillero	2019/07/08	10:33:07	abierto

Figura 91. Almacenamiento de información en Base de Datos tabla datos_casillero.
Fuente: El autor.

En la Figura 92 se muestra la información almacenada que fue tomada de la Pantalla de Registro de casillero.

Cod	Nombres	Carrera	Semestre	Seccion	N_Celular
1	Juan De la Torre	Sistemas	6	diurna	0987483889
2	Diego Galvez	Sistemas	5	Diurna	0987483889

Figura 92. Almacenamiento de información en Base de Datos tabla datos_usuario.
Fuente: El autor.

6.6 Casilleros

6.6.1 Características físicas contempladas para el prototipo CIV

En la Figura 93 se muestran las dimensiones con las cuales se construyó el casillero la medida que se utiliza son los centímetros, el material utilizado fue cartón, pero se puede fabricar con otros materiales como láminas de acero, etc.

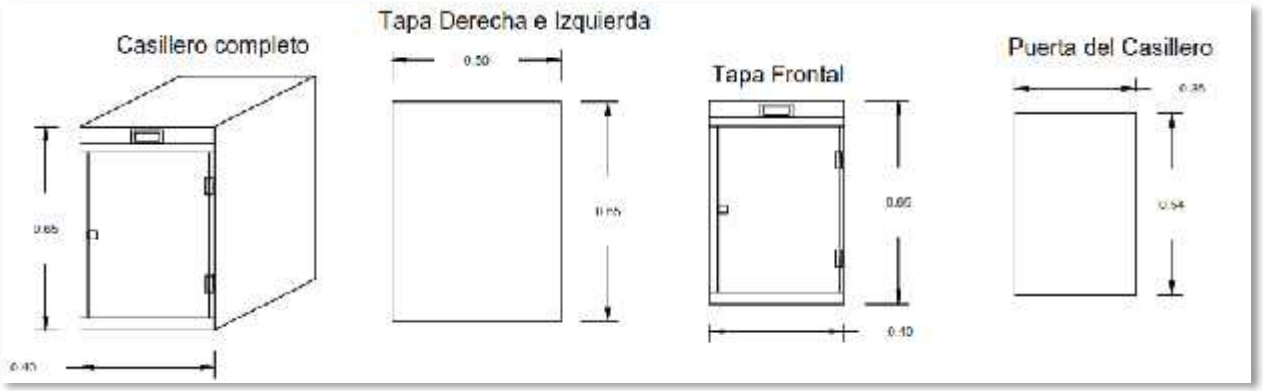


Figura 93. Dimensiones del casillero.
Fuente: El autor.

En la Figura 94 se muestran los componentes electrónicos y eléctricos instalados en el prototipo CIV.

Figura 94. Vista frontal prototipo CIV.
Fuente: El autor.

En la Figura 95 se muestran la parte superior interna con los componentes electrónicos y eléctricos instalados en el prototipo CIV.

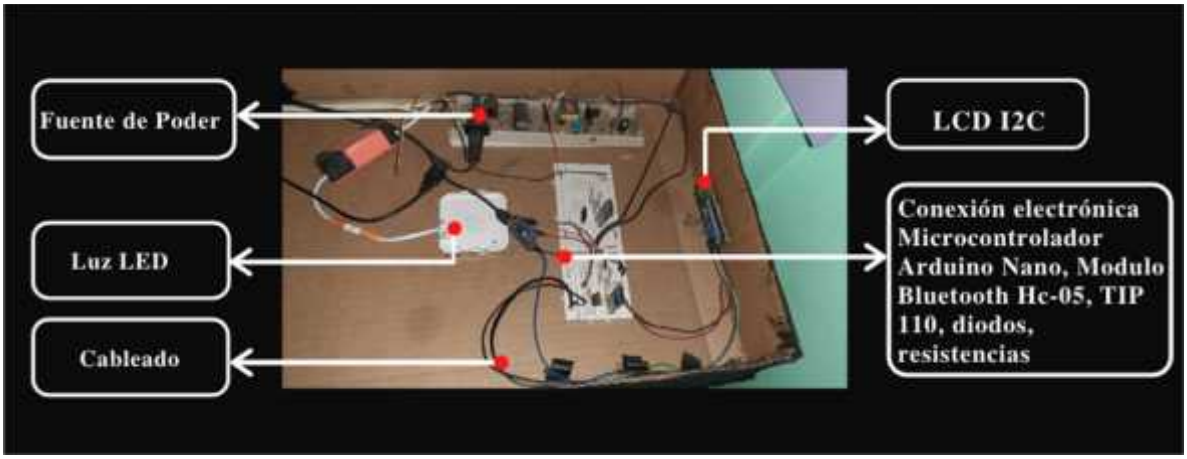


Figura 95. Vista superior interna de CIV.
Fuente: El autor.

6.6.2 Características eléctricas y electrónicas de CIV

En el prototipo CIV para la parte electrónica se posee un techo falso que se encuentra tras el LCD en la parte superior del casillero en la que se ubican todos los componentes como el módulo Bluetooth HC-05, el microcontrolador Nano Arduino, el microcomponente TIP 110 y las respectivas resistencias, de esta misma parte sale un cable de conexión USB hacia la computadora que posee el programa de registro de apertura.

La alimentación para todos estos componentes es una fuente de poder que se encuentra en la parte lateral izquierda del techo falso, esta fuente brinda 5 voltios y 12 voltios lo que es suficiente para la alimentación de todos los componentes.

La cerradura electrónica está ubicada en la parte centro izquierda del casillero, esta también es alimentada por la fuente de poder.

En el casillero también se encuentran dos fines de carrera en la parte inferior derecha la función de uno de ellos es dar el pulso que recibirá el microcontrolador para ser visualizado en el programa de registro de apertura y almacenarse en la Base de Datos, el siguiente fin de carrera se encarga del control de la iluminación mediante un foco LED alimentado independientemente con 110 voltios que se encenderá al abrir el casillero.

Potencia eléctrica del Prototipo	
6	Vatios

Figura 96. Potencia eléctrica del prototipo.
Fuente: El autor.

6.7 Pruebas del Prototipo

6.7.1 Pruebas realizadas en conjunto

Durante un periodo de 2 semanas, una vez armado el prototipo de CIV, se inició el proceso de pruebas el cual constaba del correcto funcionamiento de la aplicación con el módulo Bluetooth, la comunicación con el microcontrolador nano Arduino y el correcto envío de información a la Base de Datos.

6.7.2 Pruebas realizadas del App

Como primera prueba en la aplicación móvil se nos presentó una falla en la conexión entre la aplicación móvil y el módulo Bluetooth HC-05 no se desplegaba la lista de dispositivos para poder vincular CIV con la aplicación al momento de emparejar la MAC (siglas en inglés de Media Access Control Se la conoce también como dirección física) con el dispositivo Android.

Figura 97. Fallo de conexión Bluetooth.
Fuente: El autor.

6.7.3 Pruebas realizadas con la Base de Datos

La segunda prueba verificando el funcionamiento entre el microcontrolador y la Base de Datos se encontró una falla en el envío de información no se podía almacenar en las tablas dentro del servidor local.

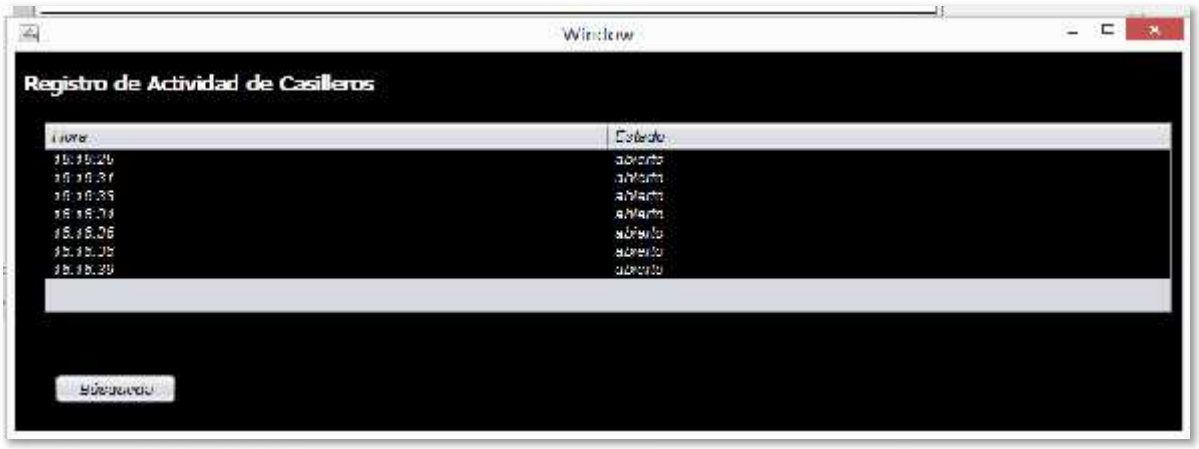


Figura 98. Registro fallido de recepción de datos CIV.
Fuente: El autor.

Al no tener una manera de verificar la información de apertura se tomó la decisión de usar un sensor de fin de carrera el cual envía un pulso serial de esta manera se pudo almacenar el estado para ser visualizado en el programa de registro y posteriormente ser almacenado en la Base de Datos.

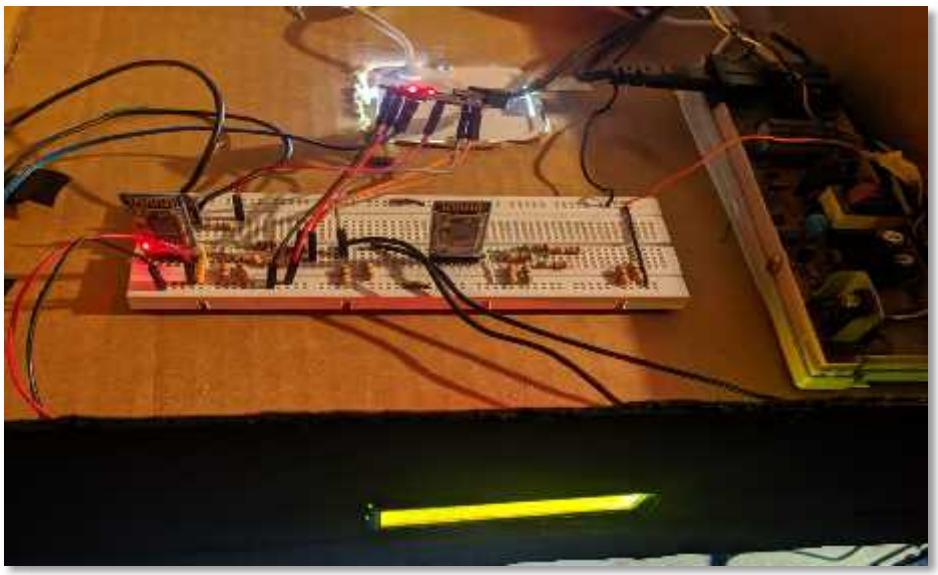


Figura 99. Error de lectura sensor fin de carrera.
Fuente: El autor.

Como prueba final, teniendo a CIV en pleno uso, se produjo un fallo por el recalentamiento de la cerradura electrónica (solenoide) la cual se averió por un cortocircuito del TIP 110 que es un componente que se encarga de la apertura de la cerradura.

Figura 100. Recalentamiento de Cerradura Electrónica.
Fuente: El autor.

A continuación, se presenta un cuadro resumen con las pruebas realizadas del proyecto. Se realizaron en total un total de 10 pruebas de funcionamiento en cada uno de los puntos que detalla el cuadro y se indican los porcentajes de éxito en el funcionamiento.

Prueba	exitosas	no exitosas	Porcentaje
Ejecución de la Base de Datos y ambiente grafico.	10	0	100%
APP dispositivo movil apertura y registro de usuario	10	0	100%
Conexión de Bluetooth de la APP dispositivo movil a casillero	10	0	100%
Envió de información del casillero a la Base de Datos	10	0	100%
Apertura de la puerta del casillero	10	1	90%
Prueba general de todo el prototipo	10	0	100%

Figura 101. Porcentaje de éxitos de las pruebas de CIV.
Fuente: El autor.

6.8 Costos

6.8.1 Costo del Prototipo

En la Figura 102 se muestra el costo que se invirtió para desarrollar el prototipo CIV.

Costo del prototipo	
Material	Costo
Cableado	20,00 $
Componentes	80,00 $
Pintura	5,00 $
Materiales varios	25,00 $
Total	130,00 $

Figura 102. Costo del prototipo.
Fuente: El autor.

6.8.2 Proyección de costos de un casillero real

En la Figura 103 se muestra el costo estimado de los casilleros CIV como producto final.

Costos para casilleros real	
Elementos	Costos
Aplicación	10,00 $
Base de Datos	5,00 $
PC	150,00 $
Interfaz Gráfica	40,00 $
Estructura casillero	450,00 $
Total	655,00 $

Figura 103. Costo casilleros real.
Fuente: El autor.

El costo de $ 450,00 representa al modelo que contiene el armazón de 4 casilleros, el precio puede variar según la cantidad de los casilleros y el montaje de la estructura metálica. Se bajo el costo de la aplicación y el desarrollo de la Base de Datos con miras a que CIV tenga un precio atractivo.

7. Conclusiones y Recomendaciones

7.1 Conclusiones

1. Se describió los conceptos, materiales y herramientas necesarios para la elaboración del proyecto a manera de marco teórico.
2. Se diseñó e implementó el proyecto partiendo de las representaciones de comportamiento y estructural de cada una de las partes que conforman CIV. Definiendo los requerimientos del Prototipo de Casilleros Inteligentes los cuales fueron crear un dispositivo que envíe un PIN mediante Bluetooth hacia el microcontrolador para permitir la apertura de los casilleros y el registro de la actividad en una Base de Datos ubicada en algún lugar de las instalaciones de las instituciones.
3. Se diseñó el circuito electrónico y eléctrico necesarios para el prototipo.
4. Se desarrolló la aplicación para dispositivos móviles con sistema operativo Android el cual cumplirá la función de llave para controlar la actividad de apertura de los casilleros.
5. Se implementó una Base de Datos MySQL dentro del servidor local para contener el registro de apertura, la información de cada usuario y casillero para ser usados en el programa de registro de actividades de CIV.
6. Se diseñó la interfaz gráfica de control de la Base de Datos para recibir los datos enviados por la Aplicación Móvil, el microcontrolador, el sensor fin de carrera y permitir visualizarlos en el programa de registro de actividades.
7. Se realizaron las pruebas de funcionamiento del prototipo de Casilleros Inteligentes probando él envió de información la visualización y el almacenamiento en la Base de Datos.

7.2. Recomendaciones

1. Se recomienda dar un mantenimiento preventivo al circuito electrónico como también a la instalación eléctrica lo cual incluye la revisión del cableado, conexión, estado del cable, estado del foco LED al interior del casillero y el correcto funcionamiento de los microcontroladores, esto se aconseja se lo realice al menos semestralmente.
2. Estos casilleros pueden ser implementados en cualquier lugar amplio y en sitios de trabajo, escuelas, colegios, universidades e institutos. Estos se harían cargo de los gastos correspondientes.
3. Se recomienda que la Base de Datos sea respaldada cada tres meses en un dispositivo seguro como un disco duro externo, en la nube o un NAS.
4. Verificar que la interfaz gráfica no presente bugs ni cuelgues al momento de ser ejecutada y estar siempre pendientes de las actualizaciones de JAVA.
5. Los circuitos eléctricos y electrónicos deben ser debidamente empotrados en el armazón de los casilleros y los cables correctamente peinados para evitar accidentes eléctricos y también ayudará con el mantenimiento de CIV.
6. Por ser un prototipo, el circuito electrónico se lo armó en un protoboard, pero para un producto final se deberá diseñar y crear el PCB correspondiente.
7. Con respecto a la fuente de poder se recomienda que sea una fuente de computadora de 300W con las modificaciones de conectores necesarias para adaptarlo al CIV, esta tiene la suficiente potencia para alimentar al proyecto.
8. Se concluye para las pruebas del casillero hacerlas en intervalos de funcionamiento similares a el uso que tendrían los usuarios, para evitar el recalentamiento de los componentes electrónicos.

Referencias

[1] Jesus, «todoparatuhotel,» 08 Marzo 2020. [En línea]. Available: https://www.todoparatuhotel.com/es/blog/cerraduras-de-seguridad-en-hosteleria/.

[2] Arduino, «Aduino,» [En línea]. Available: https://arduino.cl/producto/arduino-uno/. [Último acceso: 08 Marzo 2020].

[3] Arduino, «Arduino,» [En línea]. Available: http://arduino.cl/que-es-arduino/. [Último acceso: 08 Marzo 2020].

[4] Electronilab, «Electronilab,» [En línea]. Available: https://electronilab.co/tienda/arduino-nano-v3-atmega328-5v-cable-usb/. [Último acceso: 08 Marzo 2020].

[5] Willyfox, «todoelectrodo.blogspot,» [En línea]. Available: http://todoelectrodo.blogspot.com/2013/02/lcd-16x2.html. [Último acceso: 08 Marzo 2020].

[6] l. Llamas, «Luisllamas,» [En línea]. Available: https://www.luisllamas.es/arduino-lcd-i2c/. [Último acceso: 08 Marzo 2020].

[7] Eneka, «Eneka,» [En línea]. Available: http://www.eneka.com.uy/robotica/modulos-comunicacion/m%C3%B3dulo-interfaz-serail-i2c-detail.html. [Último acceso: 08 Marzo 2020].

[8] Naylampmechatronics, «Naylampmechatronics,» [En línea]. Available: https://naylampmechatronics.com/inalambrico/24-modulo-bluetooth-hc06.html. [Último acceso: 08 Marzo 2020].

[9] Diymakers, «Diymakers,» [En línea]. Available: http://diymakers.es/arduino-bluetooth/. [Último acceso: 08 Marzo 2020].

[10] Geekfactory, «Geekfactory,» [En línea]. Available: . thttps://www.geekfactory.mx/tutoriales/bluetooth-hc-05-y-hc-06-tutorial-de-configuracion/. [Último acceso: 08 Marzo 2020].

[11] Academia Android, «Academiaandroid,» [En línea]. Available: https://academiaandroid.com/android-studio-v1-caracteristicas-comparativa-eclipse/. [Último acceso: 08 Marzo 2020].

[12] P:U.S.E, «repositorio.puce,» [En línea]. Available: http://repositorio.puce.edu.ec/bitstream/handle/22000/13113/Anexo%203%20MIT

%20App%20Inventor%202.pdf?sequence=4&isAllowed=y. [Último acceso: 08 Mrzo 2020].

[13] D. P. Valdés, «maestrosdelweb,» [En línea]. Available: http://www.maestrosdelweb.com/que-son-las-bases-de-datos/. [Último acceso: 08 Marzo 2020].

[14] wikipedia, «wikipedia,» [En línea]. Available: https://es.wikipedia.org/wiki/NetBeans. [Último acceso: 08 Marzo 2020].

[15] C. B2B, «cerrajerosb2b,» [En línea]. Available: https://cerrajerosb2b.com/tipos-de-cerraduras/ . [Último acceso: 08 Marzo 2020].

[16] Euskera, «educacion.navarra,» [En línea]. Available: https://codigo21.educacion.navarra.es/autoaprendizaje/primeros-pasos-con-app-inventor-2/. [Último acceso: 08 Marzo 2020].

ANEXOS

ANEXO 1: ARDUINO NANO DATASHEET

ANEXO 2: HC05 DATASHEET

ANEXO 3: TIP 1010 DATASHEET

ANEXO 4: I2C DATASHEET