# ESCUELA POLITÉCNICA NACIONAL

## ESCUELA DE FORMACIÓN DE TECNÓLOGOS

### PROGRAMACIÓN DEL SISTEMA DE AUTOMATIZACIÓN DE PRÉSTAMO DE PROYECTORES Y REGISTRO DE CLASES DICTADAS EN LA ESFOT

**TRABAJO PREVIO A LA OBTENCIÓN DEL TÍTULO DE TECNÓLOGO EN ELECTRÓNICA Y TELECOMUNICACIONES**

**JORGE GEOVANNY AGUIRRE AMAY**
jorge.aguirre@epn.edu.ec

**NATHALY DAYÁN HERRERA JIMÉNEZ**
nathaly.herrera@epn.edu.ec

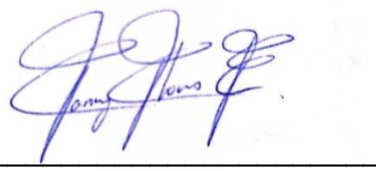
**DIRECTOR: Ing. FANNY PAULINA FLORES ESTÉVEZ MSc.**
fanny.flores@epn.edu.ec

**CODIRECTOR: Ing. MÓNICA DE LOURDES VINUEZA RHOR MSc.**
monica.vinueza@epn.edu.ec

**Quito, Agosto 2020**

# CERTIFICACIÓN

Certificamos que el presente trabajo fue desarrollado por Jorge Geovanny Aguirre Amay y Nathaly Dayán Herrera Jiménez, bajo nuestra supervisión.



Ing. Fanny Flores MSc.
**DIRECTOR DE PROYECTO**

Ing. Mónica Vinueza MSc.
**CODIRECTOR DEL PROYECTO**

# DECLARACIÓN

Nosotros, Jorge Geovanny Aguirre Amay y Nathaly Dayán Herrera Jiménez 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 las referencias bibliográficas que se incluyen en este documento.

Sin perjuicio de los derechos reconocidos en el primer párrafo del artículo 114 del Código Orgánico de la Economía Social de los Conocimientos, Creatividad e Innovación -COESC-, somos titulares de la obra en mención y otorgamos una licencia gratuita, intransferible y no exclusiva de uso con fines académicos a la Escuela Politécnica Nacional. Entregaremos toda la información técnica pertinente. En el caso de que hubiese una explotación comercial de la obra por parte de la EPN, se negociará los porcentajes de los beneficios conforme lo establece la normativa nacional vigente.

Jorge Geovanny Aguirre Amay

Nathaly Dayán Herrera Jiménez

# DEDICATORIA

Dedico este trabajo a mis padres, Marina y Felipe quienes han sido el pilar fundamental durante toda mi vida y que han estado presentes en cada etapa importante, quienes con su amor y paciencia infinita han hecho que sea acreedora de este inmenso logro.

A mi hermana, Amiel, quien siempre ha estado presta a brindarme su ayuda y consejos para lograr este objetivo.

Finalmente dedico esta importante meta culminada a mi novio, Axel, quien ha estado presente en cada circunstancia, buena o mala que he atravesado y que sin él nada de esto hubiera sido posible, porque ha hecho que este camino sea más divertido y llevadero. Te amo

Nathaly H.

# DEDICATORIA

El presente trabajo es principalmente dedicado a mi madre Luz Matilde Amay quien ha cumplido con el papel de padre y madre en mi vida, que con su gran esfuerzo me ayudó y me forjó como la gran persona que soy en la actualidad. A mis hermanos Luis y Magaly quienes han sido una parte fundamental para poder cumplir con este objetivo de vida debido a que siempre me ayudaron económica y afectivamente para que pueda acceder a una buena educación. A mi novia Nicol que ahora se ha convertido en la más grande motivación para llegar al éxito en mi vida, pues ella no solo aportó con un inmenso apoyo para el desarrollo de mi tesis, sino también para mi vida siendo mi inspiración y motivación. Finalmente, a toda mi familia Aguirre que siempre me dieron grandes enseñanzas y motivaciones para poder alcanzar mis anhelos.

Geovanny A.

# AGRADECIMIENTOS

Agradecemos primero a Dios por ser el motor que nos ha guiado en este largo camino de estudios y que sin Él nada de esto sería posible.

A la Escuela de Formación de Tecnólogos por ayudarnos en nuestra formación académica y personal para lograr nuestros objetivos en el área profesional.

A la Ingeniera Fanny Flores que ha sido nuestro apoyo durante la carrera y en el transcurso de la elaboración de este trabajo.

A la empresa Equysum que nos permitió incluirnos en el campo laboral y que nos brindó el apoyo incondicional para el desarrollo de esta meta.

En pocas palabras, agradecemos a todo aquel que nos brindó su mano para culminar con éxito esta etapa.

Nathaly y Geovanny

# ÍNDICE DE CONTENIDOS

# ÍNDICE DE FIGURAS

# ÍNDICE DE TABLAS

# RESUMEN

El presente documento se estructura en seis secciones. En la primera, se desarrolla una Introducción, enfocada al planteamiento del problema, la justificación del proyecto y una revisión teórica de todos los dispositivos necesarios para la ejecución del sistema.

En la segunda, se presenta la Metodología utilizada donde se describe el análisis del tema, métodos, técnicas y procedimientos requeridos para llevar a cabo el desarrollo del proyecto.

En la tercera sección, se encuentra la parte de Resultados y Discusión. Aquí, se parte del diseño del proyecto, para luego profundizar en el desarrollo del código del programa y posteriormente en las pruebas de funcionamiento del mismo.

En la cuarta sección, se muestran las Conclusiones y Recomendaciones que constituyen la última parte del contenido de este documento, representando un análisis final del trabajo realizado.

En una quinta sección se ha colocado la Bibliografía, que respalda los derechos de autor y donde se ha realizado una revisión profunda de fuentes que apoyan a la teoría de este proyecto.

Finalmente, en la última parte, se encuentran los anexos como documentos, diagramas y códigos que refuerzan el contenido del documento. Adicional, en esta sección se ha colocado un Manual de Usuario para la utilización correcta del sistema.

**Palabras clave:** automatización, sistema de control, Arduino, sistema biómetrico.

# *ABSTRACT*

*This document is structured in six sections. In the first, an Introduction is developed, focused on the approach of the problem, the justification of the project and a theoretical review of all the devices necessary for the execution of the system.*

*In the second, the Methodology used is presented, where the analysis of the subject, methods, techniques and procedures required to carry out the development of the project are described.*

*In the third section, there is the Results and Discussion part. Here, it starts with the design of the project, and then goes deeper into the development of the program code. The fourth section shows the Conclusions and Recommendations that constitute the last part of the content of this document, representing a final analysis of the work performed.*

*In a fifth section, the Bibliography has been placed, which supports the copyright and where an in-depth review of sources that support the theory of this project has been carried out.*

*Finally, in the last part, there are the annexes as documents, diagrams and codes that reinforce the content of the document. Additionally, in this section a User's Manual has been placed for the correct use of the system.*



***Keywords**: automation, control system, Arduino, biometric system.*

# 1. INTRODUCCIÓN

Hoy por hoy, en la Escuela de Formación de Tecnólogos (ESFOT), para los profesores registrar el sílabo, realizar su control diario y además gestionar el préstamo de un proyector para impartir la clase prevista, resultan tareas estresantes que generan una pérdida considerable de tiempo a lo largo de la jornada. Existen varios proyectores en la ESFOT, pero ninguno de ellos brinda información anticipada, para saber si está en buen estado o que cuenta con todos los cables necesarios y solo se obtiene esta información cuando el profesor ya se encuentra en el aula.

Es por ello que, el presente proyecto solventa los problemas detallados, conjugándolos en un solo sistema de automatización que está basado en las interfaces de los módulos Arduinos UNO y MEGA por su facilidad de implementación y por la disponibilidad de puertos necesarios para proyectos a gran escala.

Adicional a esto, se ha considerado sensores infrarrojos y magnéticos para el envío de señales, luces piloto como indicadores de estado y módulos relés que realicen la conmutación entre los *lockers* que contiene el sistema.

La ventaja principal de este sistema es el permitir que los profesores no pierdan tiempo en registrarse ya que harán uso de la herramienta que rige al mundo, la tecnología. De este modo, podrán reservar el proyector que se encuentre disponible y registrar el tema dictado rápidamente, gracias a un completo sistema manejado por un microcontrolador.

Cabe mencionar que el presente proyecto fue desarrollado por siete estudiantes; de modo que este documento se enfoca en la programación del mismo. El diseño, implementación y desarrollo de la interfaz de usuario se trata a profundidad en los documentos de los otros estudiantes.

## 1.1. Marco Teórico

A continuación, se detalla las partes principales que componen este proyecto:

### Módulos de Arduino

Los módulos de Arduino pertenecen a una empresa de desarrollo de *software* y *hardware* de código abierto y que desarrollan plataformas de prototipos que incluyen una fácil programación. [1] El microcontrolador ATMEL, es la base de la placa de una plataforma Arduino y que varía acorde a sus distintas versiones. Para su programación se hace uso del *Arduino Development Environment* basado en *Processing*, que trabaja en conjunto con el *Arduino Language Programming,* que se encuentra basado *en Wiring.* [2] Por el mismo hecho de que se puede interactuar desde el mismo IDE de Arduino y que puede controlarse desde un ordenador utilizando *Processing Maxmsp*, el Arduino brinda diversas facilidades al momento de su programación para la implementación de diversos proyectos electrónicos. [2]

Por las características anteriormente descritas, para el presente proyecto se ha optado por utilizar la plataforma del Arduino Mega 2560 y del Arduino UNO.

### IDE Arduino

Arduino posee un entorno interactivo con una gran cantidad de librerías que facilitan la manera de realizar un código y probarlo en la plataforma; con esta característica se logra el desarrollo e integración del código en la placa. Existen librerías específicas para cada dispositivo, módulo o sensor que sea compatible con Arduino.

Mediante la conexión USB *(Universal Serial Bus)*, se logra recibir y enviar datos del Arduino sin emplear *software* o *hardware* adicional. [3]

### Arduino Mega 2560

Esta plataforma incluye un microcontrolador Atmega 2560 y dispone de 54 pines de entrada/salida, de los cuales 14 se utilizan para aplicaciones con PWM *(Pulse Width Modulation).* Adicional, posee 16 pines que se emplean como entradas análogas, un oscilador de 16MHz, 4 UART *(Universal Asynchronous Receiver-Transmitter)* que constituyen puertos seriales de carga*,* un *jack* para la alimentación, conector USB, y un

conector ICSP *(In-Circuit Serial Programming),* tal y como se encuentra detallado en la figura 1.1. [4]

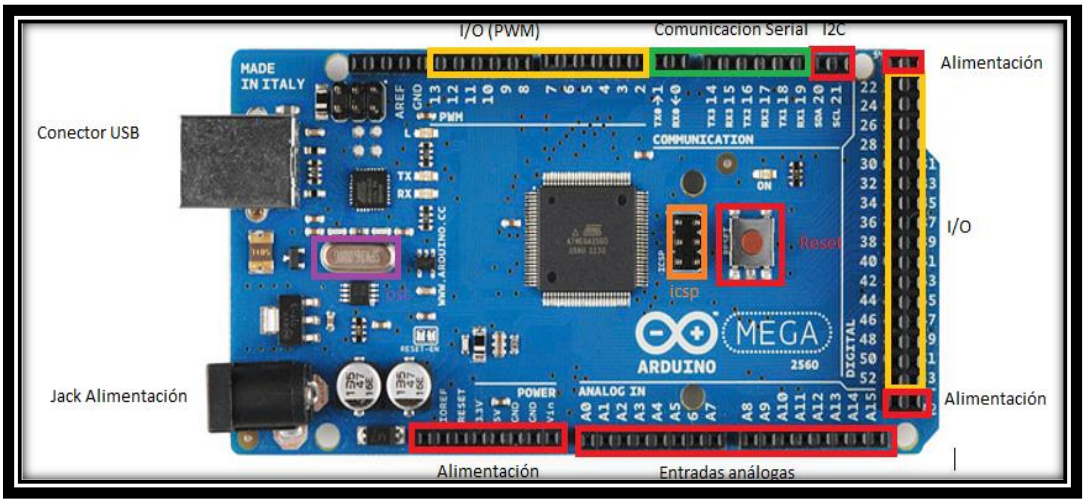



**Figura 1.1: Plataforma Arduino Mega 2560 con sus pines [4]**

Las características de operación del Arduino Mega, se pueden apreciar en la tabla 1.1 que se presenta a continuación.

**Tabla 1.1: Características del módulo Arduino 2560 [4]**

| CARACTERÍSTICAS ARDUINO MEGA 2560 | |
|---|---|
| **Voltaje de funcionamiento** | 5v |
| **Voltaje de entrada (óptimo)** | 7-12v |
| **Voltaje de entrada (máximo)** | 6-20v |
| **Corriente DC por pin (E/S)** | 40mA |
| **Corriente DC por pin (3.3V)** | 50mA |
| **Memoria *Flash*** | 256 KB (8 KB gestor de arranque) |
| **SRAM** | 8 KB |
| **EEPROM** | 4 KB |
| **Oscilador** | 16 MHz |
| **Longitud** | 101.52 mm |
| **Ancho** | 53.3 mm |

## Arduino Uno

Arduino Uno es un módulo basado en un microcontrolador Atmega328. Posee 14 pines de entrada/salida digital (de los cuales 4 pueden ser utilizados para salidas PWM), 6 entradas análogas, un resonador cerámico de 16 MHz, un conector para USB tipo hembra, un *jack* para fuente de poder, un conector ICSP y un botón de reinicio. En la figura 1.2 se presenta la tarjeta de Arduino Uno en la que se observa sus partes principales. [5]


**Figura 1.2: Tarjeta Arduino UNO [6]**

Las características principales de este módulo se amplían en la tabla 1.2, presentada a continuación.

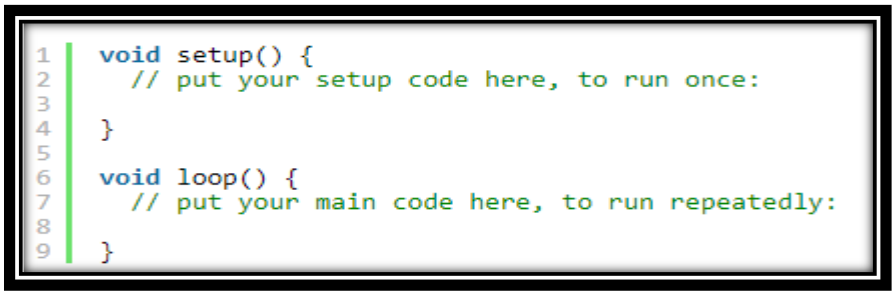
**Tabla 1.2: Características del módulo Arduino UNO [6]**

| CARACTERÍSTICAS ARDUINO UNO | |
|---|---|
| **Microcontrolador** | Atmega 328 |
| **Voltaje de funcionamiento** | 5V |
| **Voltaje de entrada (óptimo)** | 7-12V |
| **Voltaje de entrada (máximo)** | 6-20v |
| **Corriente DC por pin (E/S)** | 20mA |
| **Corriente DC por pin (3.3V)** | 50mA |
| **Memoria *flash*** | 32 KB |
| **SRAM** | 2 KB |
| **EEPROM** | 1 KB |
| **Oscilador** | 16 MHz |
| **Longitud** | 68.6 mm |
| **Ancho** | 53.4 mm |

## Estructura *Sketch* en Arduino

Un programa de Arduino se denomina *sketch* o proyecto y tiene la extensión ".ino" la cual es fundamental para que funcione el *sketch*; además el nombre del fichero debe estar en un directorio con el mismo nombre que tiene el *sketch.*

La estructura básica de un *sketch* de Arduino se presenta en la figura 1.3, en la que se observan dos partes obligatorias, las cuales encierran bloques que contienen declaraciones o también conocidas como instrucciones. [7]



```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

**Figura 1.3: Estructura básica del *sketch* de Arduino [7]**

## Funciones importantes en Arduino

- ***Setup()***

Esta función se usa para inicializar variables, establecer valores y para usar bibliotecas. El *setup()* solo se ejecutará una vez, después de cada encendido o reinicio de la placa Arduino. Toda la configuración es recogida en esta función, como se puede ver en la figura 1.4.

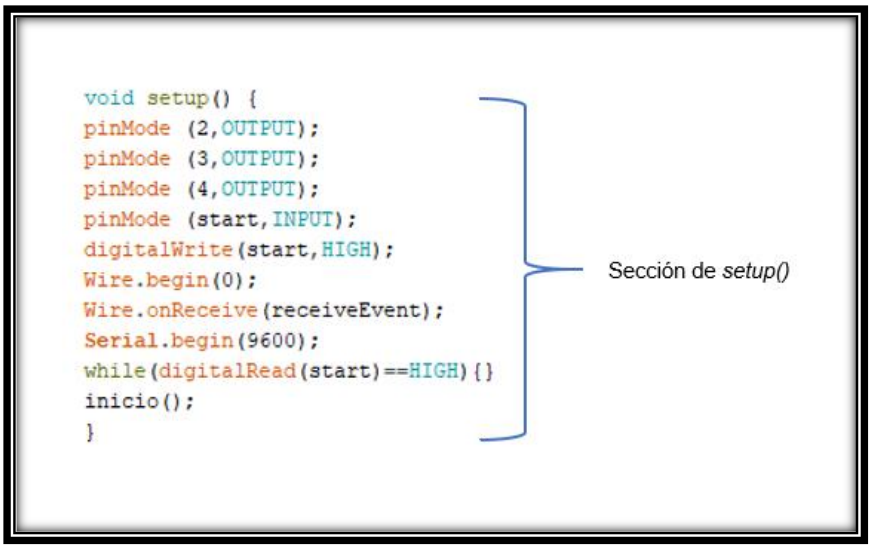



**Figura 1.4: Función *setup***

- ***Loop()***

Como se muestra en la figura 1.5, esta función almacena el programa que se ejecuta de manera cíclica; es decir, que se repite de forma consecutiva, permitiendo que su código cambie y responda.

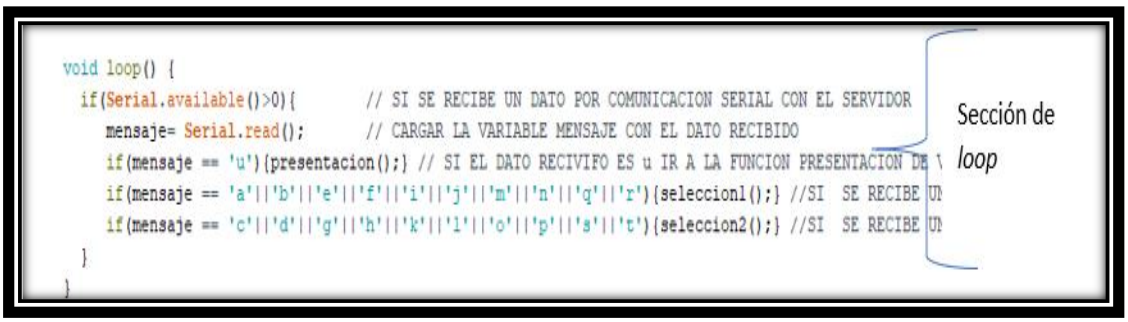



**Figura 1.5: Función *loop***

Adicionalmente, se puede incluir una introducción con los comentarios que describen el programa y la declaración de las variables y llamadas a librerías, como se observa en la figura 1.6.

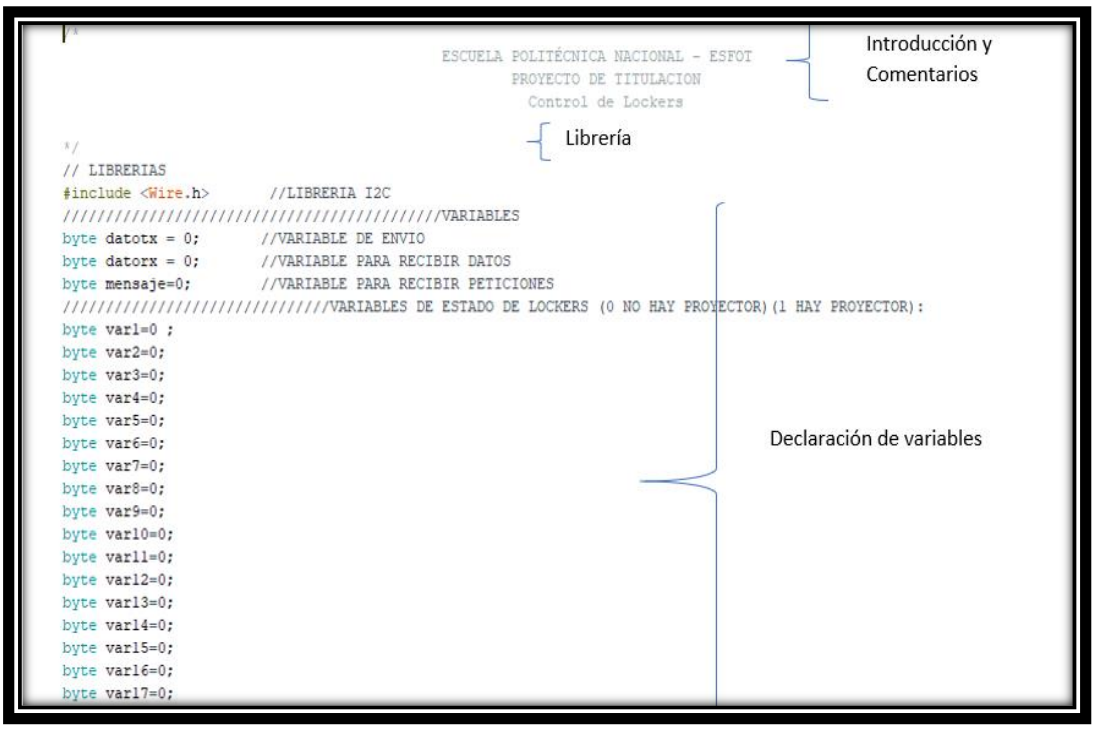



**Figura 1.6: Introducción e Inicialización de variables**

## Componentes importantes en el *sketch* de Arduino

- **Variables**

Como se muestra en la figura 1.7, las variables almacenan datos y ocupan un espacio en memoria. Deben ser ubicadas al inicio del *sketch*, para identificar los elementos que compondrán el código. De esta manera, se codifican y representan datos dentro del *sketch*.

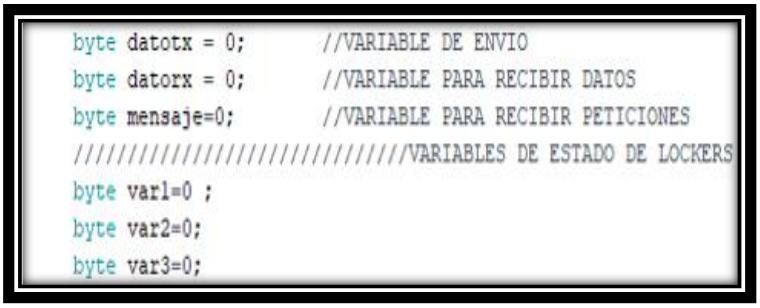



**Figura 1.7: Variables**

En la tabla 1.3 se aprecian los tipos de variables utilizados en el *sketch* del presente proyecto.

**Tabla 1.3: Tipos de variables [8]**

| TIPO | DESCRIPCIÓN | EJEMPLO |
|---|---|---|
| ***Void*** | Reservado para la declaración de funciones sin valor de retorno. | *void setup()void loop()* |
| ***Byte*** | Un número entero del 0 al 255 codificado en un octeto o *byte* (8 *bits*) | *byte testVariable = 129;* |
| ***Int*** | (Integer=entero). Un número entero entre 32, 767 y -32, 768; codificado en dos octetos (16 *bits*) | *int testVariable = 28927;* |
| ***Char*** | Un caracter ASCII almacenado en 8 *bits* (un *byte*). Esto permite almacenar caracteres como valores numéricos (su código ASCII asociado). | *char testVariable = 'a';*<br><br>*char testvariable = 97;* |

- **Funciones**

Constituyen una porción de código que puede ser usado desde cualquier parte del *sketch*. A la función se le puede llamar directamente, como se observa en la figura 1.8 o pasarle unos parámetros, en función de cómo esté definida.

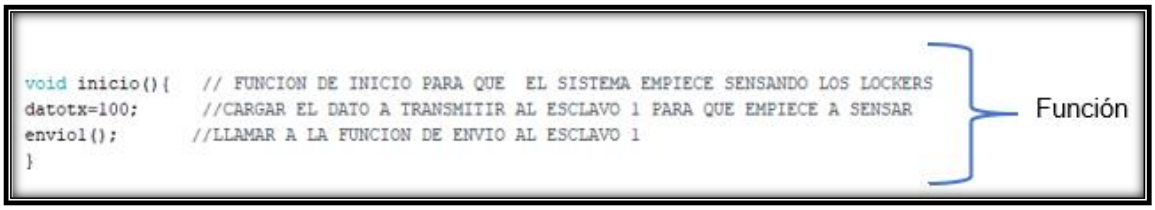



**Figura 1.8: Función de inicio**

Una función puede llamarse múltiples veces, y para hacerlo simplemente se utiliza la siguiente línea de código:

nombreFunción(parámetros);

- **Comentarios**

Son esenciales para documentar el código y llevar un orden en el proyecto que se está desarrollando en el *sketch.*

- **Librerías**

Las librerías son fracciones de código desarrollados por terceros y de libre uso para distintos *sketches*, lo que facilita mucho la programación, haciendo que sea más sencillo el desarrollo y comprensión del código.

La instrucción *"#include"* se emplea para incluir bibliotecas externas en un *sketch*. Esto da al programador el acceso a un gran número de funciones desarrolladas en lenguaje C, escritas especialmente para Arduino. [7]

## Puertos de entrada y salida

- **Función *pinMode (pin, mode)***

Los pines de Arduino funcionan por defecto como entradas, de forma que no necesitan declararse explícitamente como tal. En la figura 1.9, se muestran los 14 pines digitales que son empleados como una entrada o una salida. Cabe recalcar que la corriente

máxima de operación que soporta cada pin es de 40 mA y posee una resistencia de *pull-up* de 20 a 50 *kOhms* que por defecto se encuentra desconectada. [6]

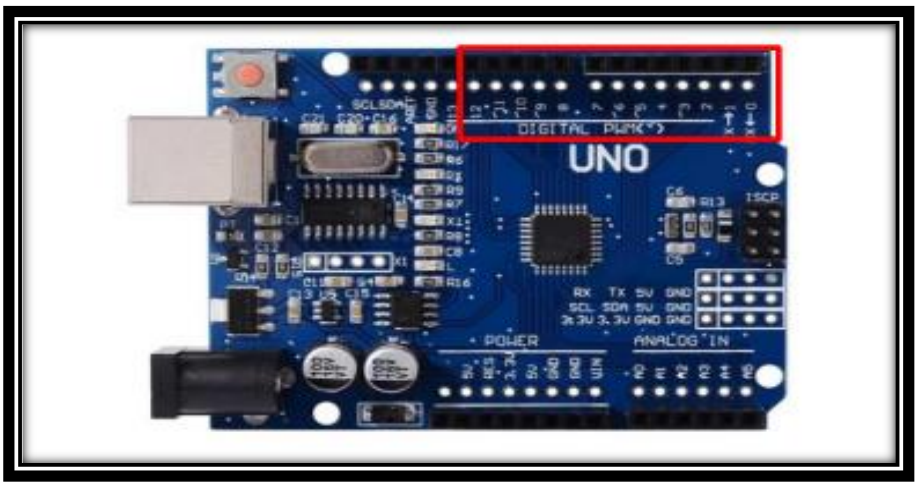



**Figura 1.9: Tarjeta Arduino UNO (Entradas-salidas digitales) [6]**

- **Función *digitalRead (pin)***

Básicamente, lee el valor desde un pin digital específico para luego devolver un valor *High* o *Low*. Puede estar especificado con una variable o una constante. [9]

- **Función *digitalWrite (pin, value)***

En el pin digital especificado, introduce un nivel alto (*HIGH*) o bajo (*LOW*). Al igual que en la función *digitalRead*, el pin puede ser especificado con una variable o una constante. [9]

- **Función *analogRead (pin)***

Esta función solo funciona en los pines analógicos. El valor resultante es un entero de 0 a 1023. Los pines analógicos, a diferencia de los digitales no necesitan declararse previamente como entrada/salida. [9] En la figura 1.10, se aprecian las 6 entradas analógicas que posee el Arduino, cada una de ellas ofrece 10 bits de resolución y trabaja con un voltaje de 5V. [6]

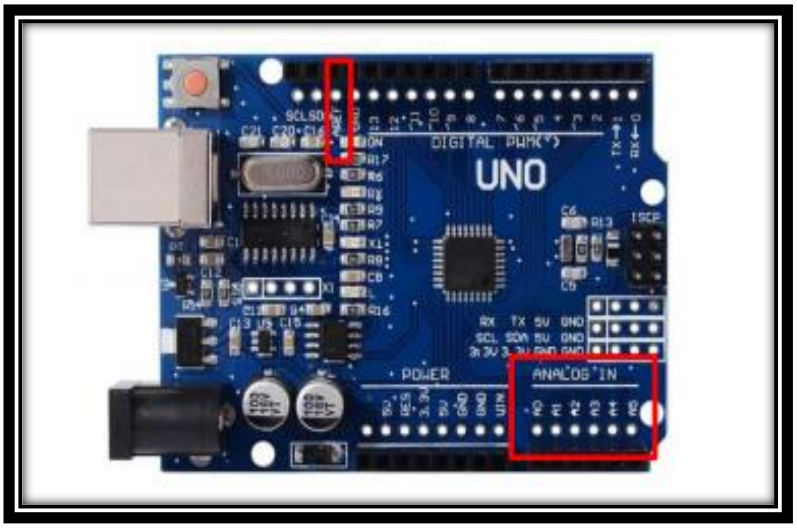



**Figura 1.10: Tarjeta Arduino UNO (Entradas analógicas) [6]**

- **Función *analogWrite (pin, value)***

Usa la modulación por ancho de pulso en un pin de salida marcado como PWM. [9]

## Protocolo de Comunicación I2C *(Inter-Integrated circuits)*

Es un tipo de bus que se utiliza para conectar circuitos integrados. Debido a que posee múltiples maestros, pueden ser conectados varios chips a un mismo bus, con la ventaja de que todos ellos pueden comportarse como maestros o esclavos, permitiendo la transferencia de datos entre ellos. [10] El bus I2C es considerado como una estándar, cuya principal función es la intercomunicación entre microcontroladores o cualquier dispositivo como pueden ser las memorias. Además, permite el intercambio de información a una velocidad de 100 kbps y en ciertos casos el oscilador puede llegar a trabajar hasta los 3.4 Mhz. [10]

Es importante recalcar que la comunicación se da de manera serial y sincrónica por medio de dos señales, una de ellas para la marcación del tiempo y la otra para el intercambio de datos. Siempre será el maestro el encargado del reloj. [10] Al ser un protocolo de dos hilos de control, en la tabla 1.4 se presenta la descripción de las señales que lo componen.

**Tabla 1.4: Descripción de las señales de I2C [10]**

| SEÑALES DEL BUS I2C | |
|---|---|
| SCL *(System Clock)* | Encargado de los pulsos de reloj para la sincronización del sistema. |
| SDA *(System Data)* | Encargada de la transferencia de datos entre los microcontroladores. |
| GND *(Ground)* | Es el común entre los dispositivos conectados al bus. |

Cada dispositivo que se encuentra conectado al bus I2C tiene una dirección de 7 bits. Lo cual resulta $2^{27} = 128$ dispositivos que se pueden conectar a la línea de bus. [11]

La idea es que todos los componentes se conecten en paralelo a las dos líneas del bus, SDA y SCL. Solo puede haber un maestro a la vez; en el caso de la figura 1.11, es el Arduino UNO y los demás se configuran como esclavos.

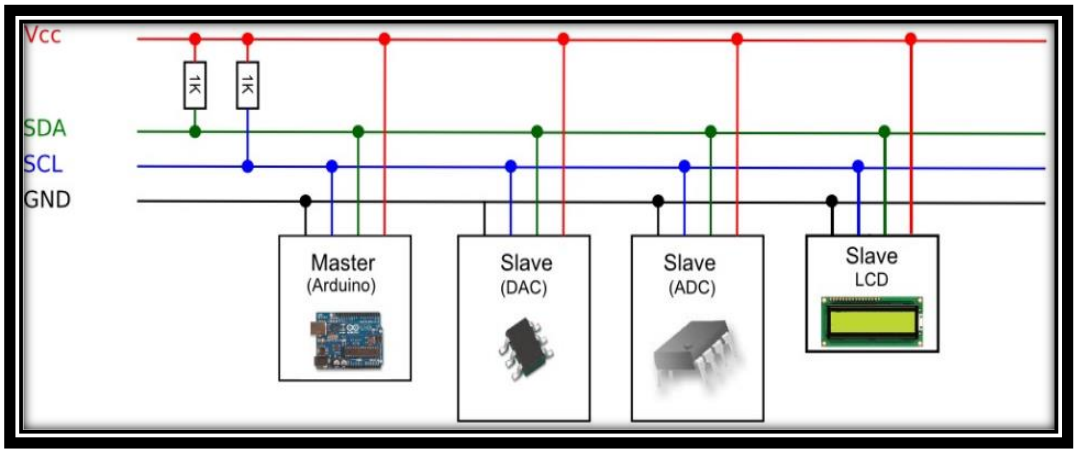



**Figura 1.11: Dispositivos Maestro-Esclavos [11]**

- **Características**
  - ✓ Utiliza dos resistencias de *pull-up* conectadas a SDA (Datos) y SCL (Reloj). Al ser el bus un activo bajo, la señal activa es un 0 y no un 1; es decir, que es inverso.

- ✓ El módulo Arduino trabaja con el protocolo I2C de fábrica con una librería estándar, utilizando dos de los pines analógicos para las funciones SDA y SCL.
- ✓ En el Arduino UNO, los pines I2C están en los pines analógicos etiquetados como A4 (SDA) y A5 (SCL) y en el Arduino Mega, son el 20 (SDA) y el 21(SCL).
- ✓ En Arduino, la librería I2C se llama *Wire* y gestiona toda la comunicación.

- **Descripción de la comunicación I2C**

Para que se realice el proceso de la comunicación del bus I2C, primero el dispositivo maestro debe comunicarse con un esclavo, lo que ocasiona una serie de inicio en el bus. Esta secuencia constituye una de las dos secuencias especiales que posee el bus I2C; y la otra es la secuencia conocida como de parada. Estas dos secuencias pertenecen a los dos únicos casos en los que SDA cambie cuando la línea SCL está en valor alto (*high)*.

Cuando se están transmitiendo datos, la línea SDA debe permanecer sin ningún cambio, lo cual involucra que la línea SCL esté en alto. Cabe recalcar que los datos se transfieren en secuencias de 8 bits. Estos bits se colocan en la línea SDA, comenzando por el bit más significativo. Cuando se ha puesto un bit en SDA, se lleva la línea SCL a alto, gracias al resistor de polarización. Por cada 8 bits transferidos, el *chip* que recibe el dato, envía de regreso un bit de reconocimiento, lo que equivale a 9 pulsos de reloj por cada 8 bits de dato. Si el dispositivo que está recibiendo la información envía un bit de reconocimiento en bajo, muestra que se ha recibido el dato y que puede receptar otro *byte*. Caso contrario, si devuelve un alto, significa que no puede recibir más datos y el maestro está obligado a concluir la transferencia y envía una secuencia de parada. [10]

## Comunicación Serial

La comunicación serial es de gran relevancia para los protocolos utilizados en la actualidad debido a que muchos dispositivos de comunicación inalámbrica la usan para hablar con el módulo Arduino como los módulos *bluetooth* y los módulos *Xbee*. Además, la comunicación serial es la que comunica el Arduino con el ordenador.

Todos los módulos Arduino tienen por lo menos un puerto serie disponible en los pines digitales 0 (RX) y 1 (TX) compartido con el USB. Por lo tanto, no es posible usar estos pines como entradas/salidas digitales. [6]

El Arduino Mega dispone de tres puertos adicionales Serial1 en los pines 19 (RX) y 18 (TX), Serial2 en los pines 17 (RX) y 16 (TX), Serial3 en los pines 15 (RX) y 14 (TX). Estos pines no están conectados al interfaz USB del Arduino.

Las funciones más importantes que se debe conocer para manejar el puerto serie son presentadas en la tabla 1.5

**Tabla 1.5: Funciones de la comunicación serial [6]**

| COMUNICACIÓN SERIE | |
|---|---|
| ***begin()*** | Establece la velocidad de la UART en "baudios" para la transmisión serie, también es posible configurar el número de bits de datos, la paridad y los bits de parada; por defecto es 8 bits de datos, sin paridad y un bit de parada. |
| ***read()*** | Lee el primer *byte* entrante del *buffer* serie. |
| ***write()*** | Escribe datos en binario sobre el puerto serie. El dato es enviado como un *byte* o serie de *bytes*. |
| ***print()*** | Imprime datos al puerto serie como texto ASCII, también permite imprimir en otros formatos. |
| ***available()*** | Da el número de *bytes* (caracteres) disponibles para leer en el puerto serie, son datos que han llegado y se almacenan en el *buffer* serie que tiene un tamaño de 64 *bytes.* |
| ***end()*** | Deshabilita la comunicación serie, permitiendo a los pines RX y TX ser usados como pines digitales. |

## Luces Piloto y Módulos Relé

En la Figura 1.12 se observa las luces AD16-22D/S en color Verde y Rojo, utilizadas para indicar el estado de los casilleros.


**Figura 1.12: Luces AD16-22D/S en color verde y rojo [12]**

Este tipo de luces fueron elegidas debido a sus características técnicas que se presentan en la tabla 1.6, lo cual garantiza mucho más tiempo de vida útil que otras luces.

**Tabla 1.6: Especificaciones técnicas de las luces piloto [12]**

| CARACTERÍSTICAS TÉCNICAS DE LUCES PILOTO | |
|---|---|
| **Nombre del producto** | LED Indicador Luz Piloto |
| **Modelo** | AD16-22D/S |
| **Voltaje/Corriente** | AC 110 V/20 mA |
| **Número de pines** | 2 |
| **Material** | Plástico y componentes eléctricos |
| **Color de la luz** | Verde y rojo |
| **Agujero de montaje** | 22 mm/0.87” |
| **Tamaño** | 29x50mm/1.1”x2” |
| **Peso** | 35gr |

## Módulo *Relé*

El módulo elegido para el presente proyecto está comprendido por 16 relés como indica la figura 1.13, que está alimentado por un voltaje de 5 voltios y cuya principal función es activar cualquier tipo de dispositivos ya que se pueden conectar a un pin digital del Arduino, manejando un nivel bajo. [12]

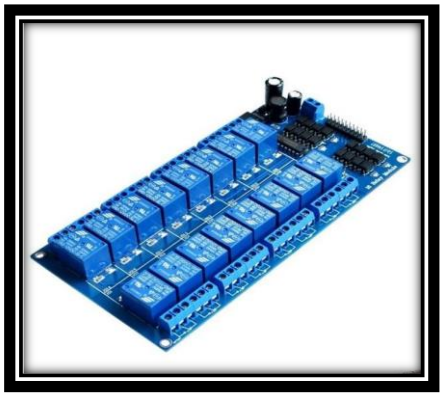


**Figura 1.13: Modulo relé 16 canales 5V para Arduino [12]**

## Cerradura Eléctrica

Cada puesto del sistema emplea una cerradura eléctrica fabricada de hierro y que posee una bobina de cobre, además soporta 12v DC (*Direct Current*) y es compatible para Arduino. Entre sus características básicas está su diseño delgado como se observa en la figura 1.14; una vez que se conecta a una fuente de alimentación, la lengüeta se abrirá, lo que hará que su consumo de energía sea muy bajo otorgando seguridad y estabilidad. [13]

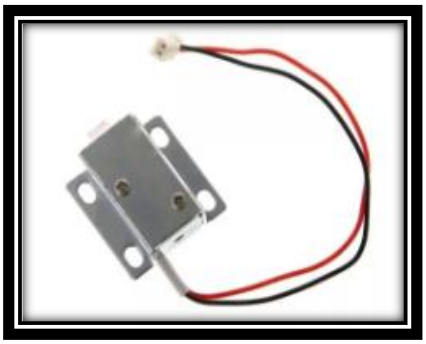


**Figura 1.14: Cerradura eléctrica 12v DC [13]**

## Sensores

Son dispositivos cuya función principal es detectar cualquier estímulo externo y dar una respuesta inmediata para enviar la información a un sistema central; es decir, capta una señal física y la transforma en una señal eléctrica con el fin de que un microprocesador procese los datos obtenidos. [14]

- **Sensor Infrarrojo**

Para detectar si el proyector se encuentra disponible para el préstamo, se optó por un sensor infrarrojo pero que fuese industrial, el cual se muestra en la figura 1.15 esto debido a que ofrece mejor calibración por su amplio rango de detección, sin mencionar que fue muy fácil de usar y montarlo en la estructura. [15]

**Figura 1.15: Sensor infrarrojo Akozon E18-D80NK de 30cm [14]**

Además, sus características más importantes han sido ampliadas en la tabla 1.7 las cuales hacen que este sensor sea idóneo para aplicaciones con Arduino.

**Tabla 1.7: Características del sensor E18-D80NK [16]**

| CARACTERÍSTICAS TÉCNICAS SENSOR INFRARROJO | |
|---|---|
| **Rango de detección** | 3 a 80cm (se puede calibrar) |
| **Voltaje con el que opera** | 5V DC. |
| **Corriente máxima soportada** | 100mA. |
| **Salida** | Tipo NPN normalmente abierto. |
| **Medidas** | 18x45mm. |
| **Longitud del cable** | 50cm. |
| **Material del que está hecho** | Plástico |
| **Temperatura de operación** | -25 a 70°C. |
| **Conexión cable rojo** | +5V |
| **Conexión cable negro** | GND (tierra) |
| **Conexión cable amarrillo** | Señal |

Este dispositivo, pertenece a los sensores ópticos, los cuales se basan en el principio de reflexión para múltiples aplicaciones de detección de objetos e incluso para conteo de personas.

- **Sensor contacto magnético**

Luego de realizar un análisis, se llegó a la conclusión de que era necesario usar un sensor de contacto magnético, el cual fue ubicado en cada puerta de los *lockers*. Esto, con el fin de que este sensor fuera el encargado de enviar una señal a los Arduinos. En caso de que no se hubiere cerrado una puerta después de haber sido abierta para el retiro o devolución de un proyector, el programa cargado no continúa con las siguientes sentencias y así se evita errores de funcionamiento.

Un *reed switch* (sensor contacto magnético), posee diferentes encapsulados que varían según la forma y el material del que están fabricados y pueden ser usados en cualquier aplicación de exteriores, rejas, puertas o portones.

El sensor que se muestra en la figura 1.16 es un encapsulado con imán que consta de un material llamado álnico 5, el cual permite la magnetización entre los contactos. Cabe recalcar que dichos contactos se encuentran eléctricamente aislados entre sí y solamente se cierran cuando un campo magnético adecuado ha sido acercado. Por todas las características anteriormente mencionadas, este dispositivo es considerado pasivo ya que no necesita de alimentación propia. [17]

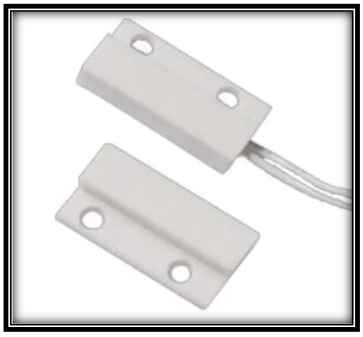


**Figura 1.16: Sensor contacto magnético [18]**

# 2. METODOLOGÍA

Para iniciar, se realizó un levantamiento de información de los proyectores y *kits* con los que cuenta la ESFOT, de modo que se garantice un correcto dimensionamiento de la estructura. Luego, se procedió con el diseño de *hardware* y *software* para continuar con la implementación y finalizar con las pruebas de funcionamiento y correcciones correspondientes. El proceso general se presenta en la figura 2.1.

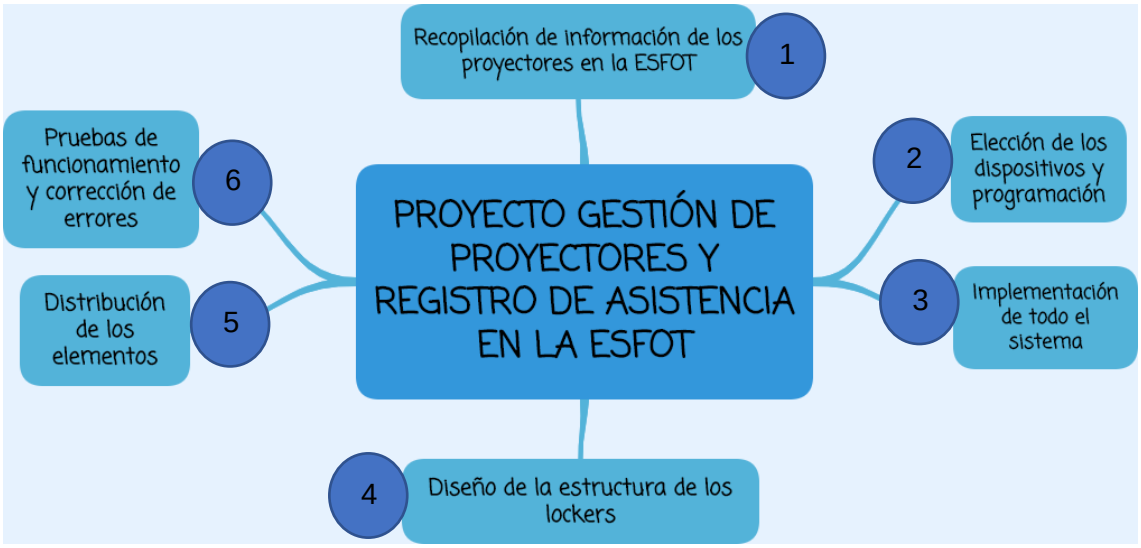



**Figura 2.1: Proceso general de la elaboración del proyecto**

## 2.1. Metodología Analítica

Se ha optado por la metodología analítica debido a que se ha partido de un todo para descomponerlo en sus partes para el estudio de cada elemento que lo conforman. Es así que, se ha empleado un estudio descriptivo para la recolección de datos sobre el número de proyectores que posee la ESFOT; de este punto se partió para el diseño del sistema y posterior elección de los elementos que lo compondrían.

Se consideraron todas las ventajas que poseen los dispositivos y sensores involucrados en el sistema. Además, se analizó las características técnicas que posee cada Arduino ya que, al tener una gran cantidad de puertos y una fácil programación, se logró integrar

el sistema de manera eficaz y a parte se consiguió la intercomunicación de los mismos mediante I2C.

Dentro de las características fundamentales que fueron analizadas, se optó por una comunicación serial ya que el envío de datos es *byte* a *byte*, lo que significa que se requiere una menor cantidad de líneas de transmisión que en la comunicación *Ethernet.*

### 2.2. Metodología Experimental

Para el desarrollo del código del presente proyecto, previamente se realizaron diagramas de flujo y una investigación sobre los sensores y demás elementos, con el fin de que cada parte realice su función. Se programó uno por uno los elementos y se fueron realizando pruebas del código de manera individual. Una vez que todos los dispositivos funcionaban correctamente, se integraron al sistema de automatización y es así que, se obtuvo un solo código.

Entre las partes esenciales que componen el sistema se encuentra un Arduino UNO como maestro para el manejo, mediante I2C, dos Arduinos MEGA que funcionan como esclavos y que son los encargados de enviar y recibir señales de los sensores infrarrojos y magnéticos. Además, se incluyó módulos relé para la parte de conmutación de: luces piloto como indicadores de estado de disponibilidad, cerradura magnética para abrir y cerrar los *lockers*, y la activación de los sensores magnéticos.

Finalmente, para la fase de pruebas, se realizaron modelos a pequeña escala para verificar el correcto funcionamiento del código con todas las partes integradas.

## 3. RESULTADOS Y DISCUSIÓN

### 3.1. Descripción General del Proyecto

En base al objetivo planteado en el presente proyecto, se ha desarrollado la programación para el sistema de automatización de préstamo de proyectores y registro de clases dictadas en la ESFOT.

El sistema general presentado en la figura 3.1, utiliza un servidor donde se encuentra albergado el *software* del proyecto, este *software* se encarga de la interacción entre el usuario y el sistema eléctronico para el préstamo o devolución de proyectores y registro de las clases de los profesores. Este programa es presentado en una *tablet* que se encuentra empotrada en la estructura y realiza la comunicación con el servidor mediante un enlace *Wi-Fi*; además, utiliza un biométrico encargado de verificar la identidad de los usuarios para permitir el ingreso al sistema general, este lector de huellas se encuentra conectado a un *router* mediante un cable de red.

Por otro lado, el sistema electrónico utiliza tres dispositivos de control, un Arduino maestro (Arduino UNO) y dos arduinos esclavos (Arduinos Mega). El Arduino maestro, es el encargado de recibir peticiones de préstamo y devolución de proyectores por parte del servidor, adicionalmente también se encarga de enviar toda la información del estado de cada *locker*, el cual se presenta en la *Tablet*. Para comunicarse con el servidor utiliza comunicación serial y para la comunicación con los Arduinos esclavos lo hace mediante el protocolo de comunicación I2C.

Cada esclavo es el encargado de controlar 10 *lockers*, como se puede observar en la figura 3.1. En cada *locker,* se encuentra dos luces piloto, un sensor infrarrojo, una cerradura magnética y un sensor magnético. Para controlar el encendido de las luces piloto que trabajan con 110v, se optó por utilizar módulos relé que conmutarán dependiendo el dato que procese el Arduino esclavo. Para la activación de los sensores infrarrojos y las cerraduras mágneticas también se utilizan módulos relé, de este modo los sensores infrarrojos no se encuentran sensando todo el tiempo sino solamente cuando sea requerido. La lectura del pin de datos del sensor infrarrojo y el control de los sensores magnéticos se lo realiza directamente desde la placa Arduino Mega.

Cuando el servidor tenga una orden de préstamo de un proyecto específico por parte del usuario, este enviará un dato único al Arduino maestro, a la vez procesará dicho dato y lo reenviará a uno de sus esclavos dependiento que dato sea. Una vez que el Arduino esclavo haya recibido este dato, lo procesará y mandará a abrir la cerradura del *locker* que el usuario quiere reservar para retirar el proyector, el programa no continuará mientras no se cierre la puerta del *locker* y esta acción será detectada gracias a los sensores magnéticos. Cuando la puerta sea cerrada se activará el sensor infrarrojo por el lapso de un segundo, para verificar si efectivamente se retiró o no el proyector y ese

nuevo dato será enviado al maestro para que este dispositivo se encargue de enviar la nueva información al servidor y se actualice el estado del *locker* en el sistema. Este proceso se cumple en los 20 *lockers* del módulo de proyectores.

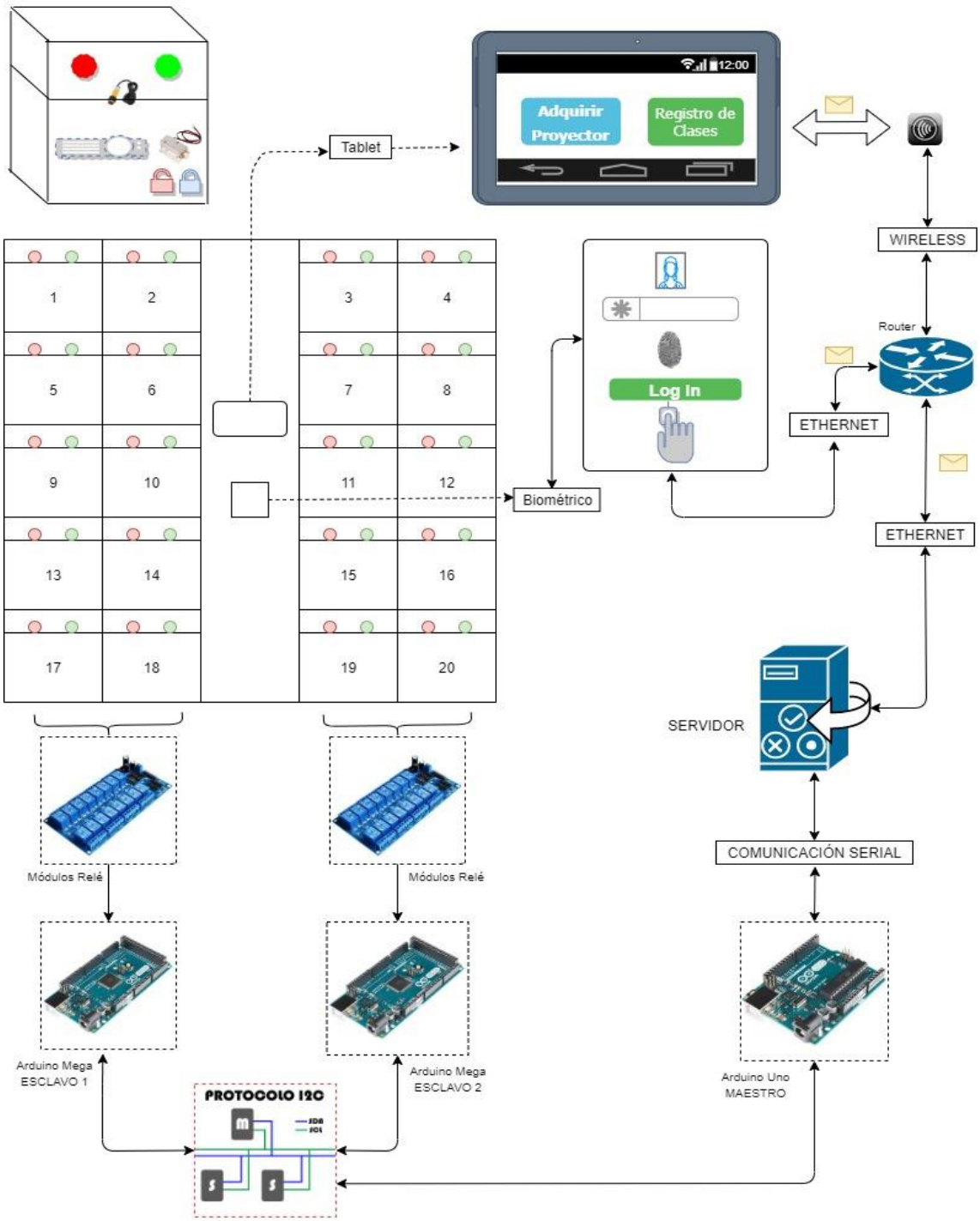



**Figura 3.1: Esquema General y manejo del protocolo I2C**

## 3.2. Diseño General del sistema

En primera instancia, se procedió a la elección de los dispositivos de control que fueron programados y que realizan la intercomunicación de todos los elementos. Para este proyecto se utilizó un Arduino Uno como maestro y dos Arduinos Mega como esclavos, debido a su fácil programación y múltiples puertos para el control de cada *locker*.

Una vez que se obtuvo esta información, también fue necesario seleccionar los demás dispositivos como sensores debido a que tenían que ser compatibles con el módulo de control elegido, es por eso por lo que se buscó la información a través de fuentes bibliográficas y *datasheets* para asegurarse que se acoplaran al sistema.

### Diagramas de flujo

Para el desarrollo del programa, primero se realizaron los diagramas de flujo presentados en las figuras de la 3.2 a la 3.4 que ayudaron a tener una mejor visión de lo que hace cada módulo de Arduino con los demás dispositivos elegidos. Se elaboraron 3 diagramas de flujo principales, en los que se muestran las funciones de los 3 Arduinos empleados (un maestro y 2 esclavos). Se optó por un diseño de programación modular con una técnica llamada *Top-Down* (análisis descendente), que permite dividir la programación en varios subprogramas con el fin de que sean mucho más fáciles de comprender; es decir que son programas pequeños independientes que a la vez trabajan en conjunto, una vez que se han unido al programa principal. Esta técnica es muy utilizada cuando la programación es compleja, hay varios dispositivos y además existe más de una persona que trabajará en el código, por lo que resulta más simple dividir en segmentos. [19]

- **Arduino maestro**

En la figura 3.2 se muestra el diagrama de flujo del módulo Arduino UNO que es el que trabaja como maestro. En este diagrama se muestra la función de inicio, en la cual se carga el dato en la variable destinada para la transmisión y que es enviada al esclavo 1, para que este pueda sensar por primera vez los 10 *lockers* que controla. Adicional a esto, se han incluido las librerías, se han declarado las variables y además se ha configurado la comunicación I2C con el uso de interrupciones y la comunicación serial entre dispositivos.

Para llevar a cabo todo lo anteriormente mencionado, se ha creado la función envio1 para la respectiva transmisión al dispositivo 1 (esclavo 1); paralelamente se crea la función envio2 para la transmisión de datos al esclavo 2. De igual manera, se han incorporado *LEDs* indicadores de envío de datos.

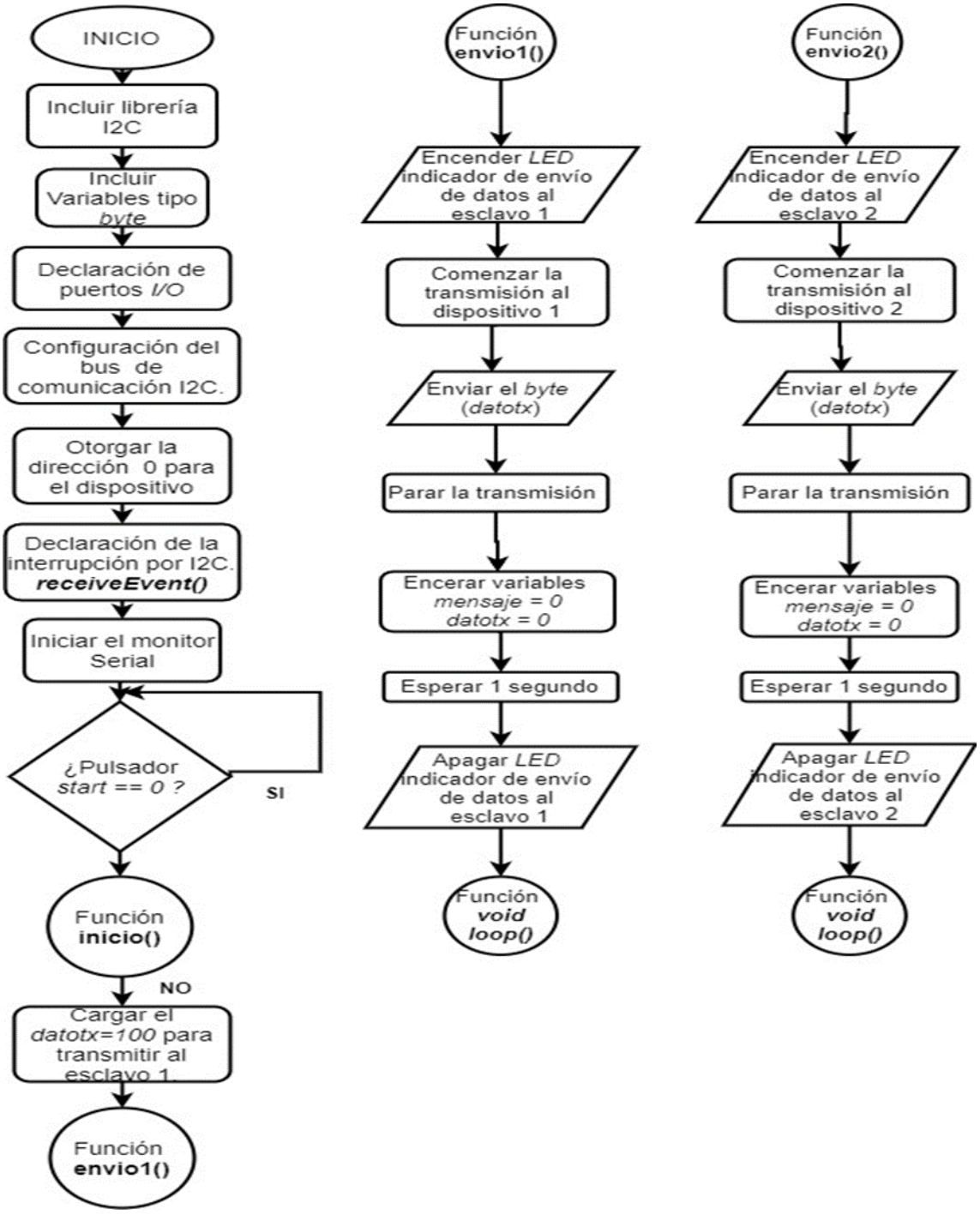



**Figura 3.2: Envío de datos a los esclavos**

En la figura 3.3 se observa la función *void loop ();* esta etapa es la de envío y recepción de datos por parte de los esclavos hacia el maestro. Aquí, también se ha incluido al servidor ya que es esencial para que el maestro pueda gestionar los datos recibidos y según ello, enviar dichos datos por medio de las funciones presentación o selección, según sea el mensaje recibido. Cabe recalcar que en esta fase será de gran relevancia la interrupción por I2C, ya que se ejecutará cuando se reciba datos de los esclavos.

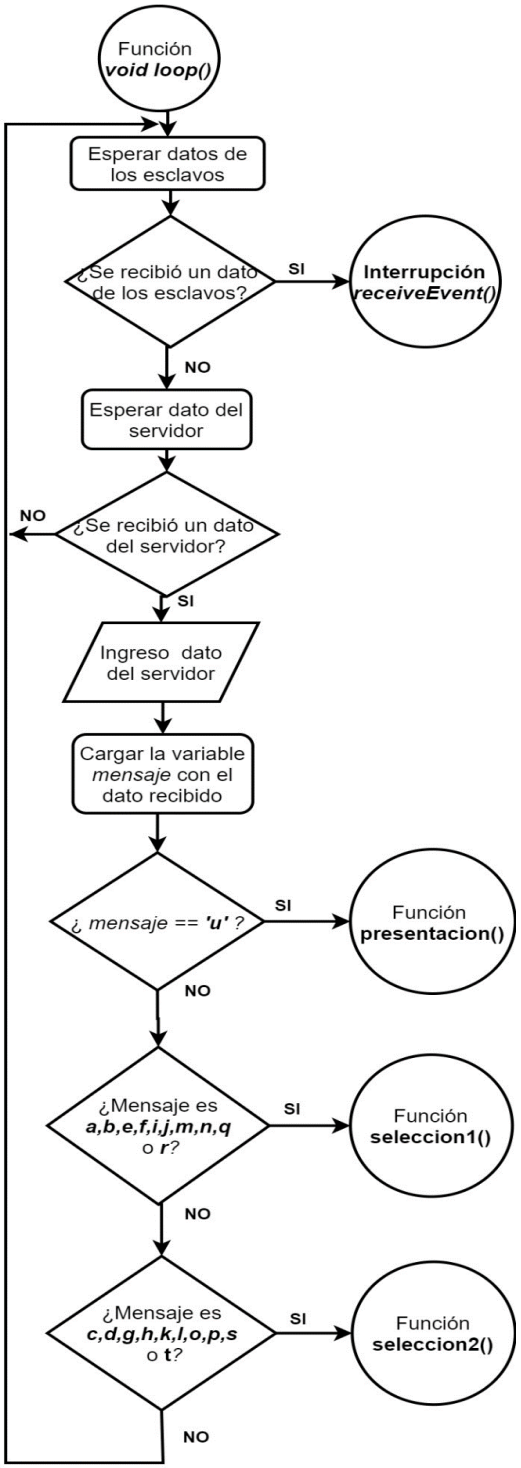


**Figura 3.3: Envío de datos al maestro**

Como se mostró en la etapa anterior, según sean los mensajes recibidos del servidor, se procederá a la función presentación o la función selección. Como se identifica en la figura 3.4, la función presentación, gracias a la comunicación serial, es la encargada de enviar al servidor los estados (ocupado o desocupado) de los 20 *lockers* que posee la estructura, así como la información de si alguna puerta se encuentra abierta.

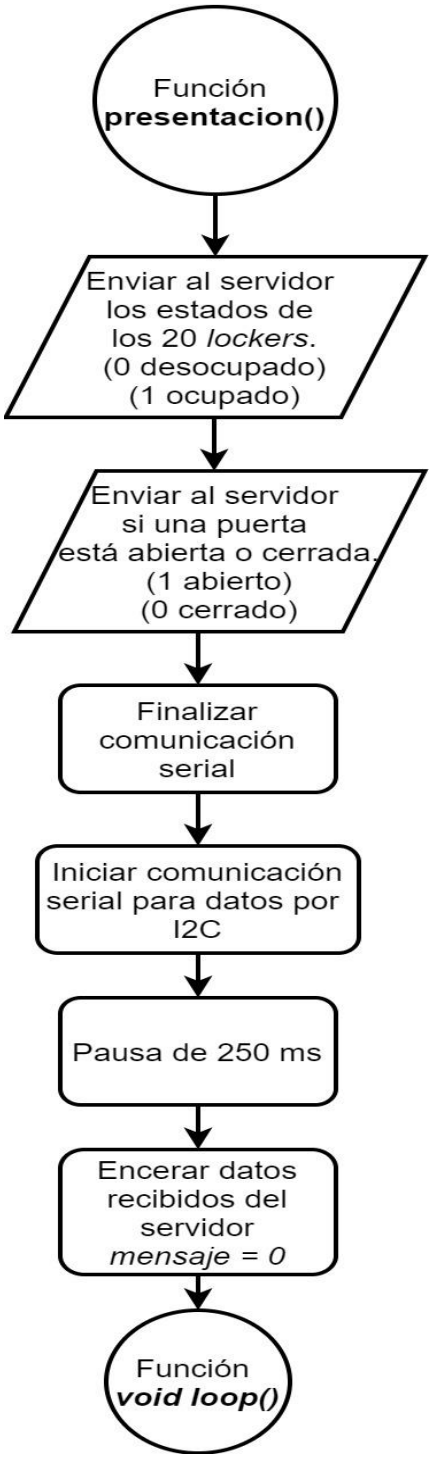



**Figura 3.4: Función presentación**

Por otro lado, está la función selección, en la cual el usuario podrá reservar el proyector que más se adapte a sus necesidades; para esta fase nuevamente el Arduino maestro enviará estos datos hacia los esclavos. En la figura 3.5 se presenta la función selección 1 y en la figura 3.6 la función selección 2.

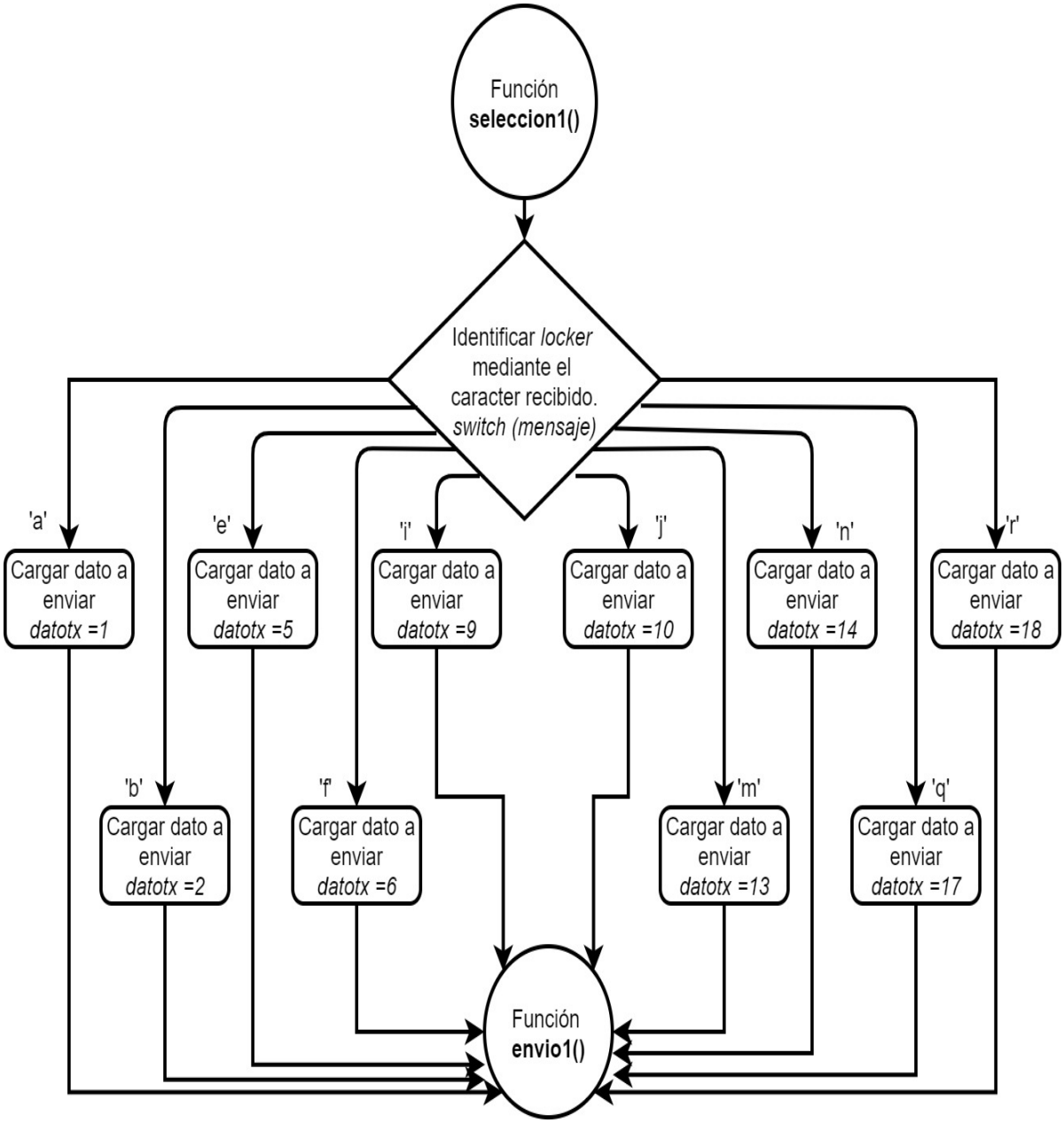



**Figura 3.5: Función selección 1**

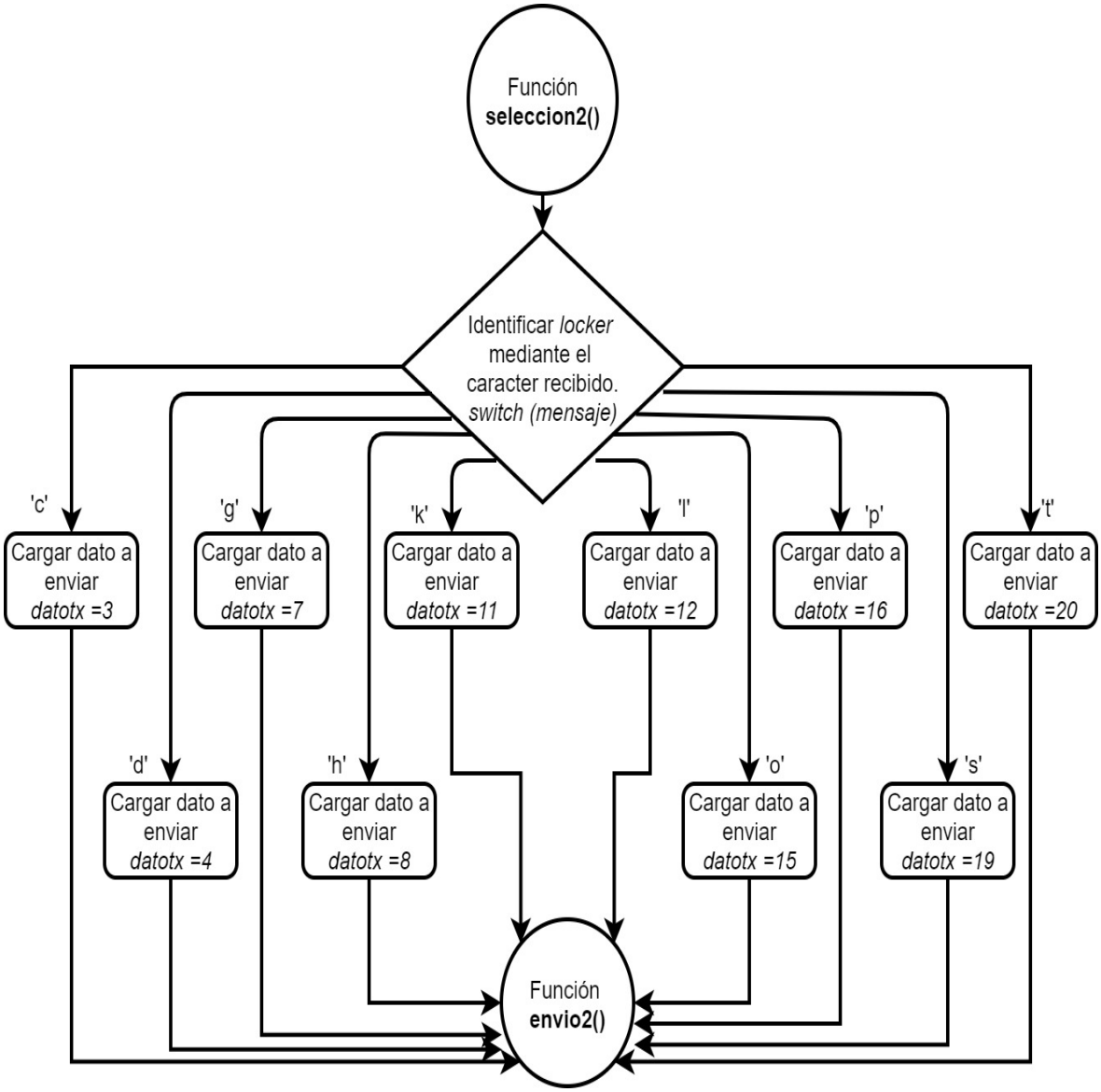



**Figura 3.6: Función selección 2**

Es importante destacar la interrupción que ocurre cuando ingresan datos de los esclavos al maestro, como en la figura 3.7, en este caso. Dicha interrupción se puede ejecutar en cualquier momento en la cual primero se verifica si hay un *byte* disponible para proceder a cargar la variable con el *byte* recibido. Una vez que haya ingresado el dato, aparecerán dos nuevas funciones llamadas par e impar para poder discriminar cuando hay y no hay proyectores.

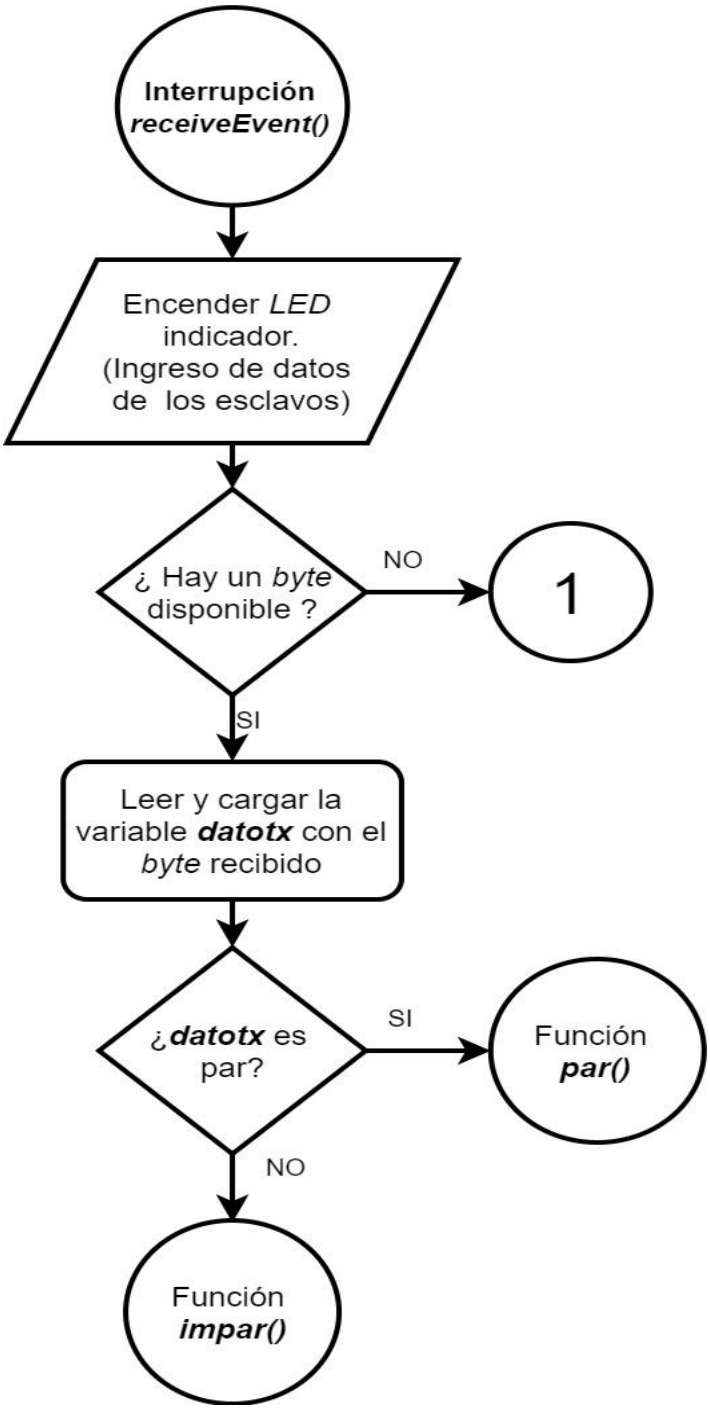



**Figura 3.7: Interrupción I2C**

Como se observa en la figura 3.8, la función par, identifica el valor recibido por I2C y procederá a cargar el valor 0 a las variables en las cuales conste que no hay proyector. Además, si se encuentra una puerta abierta, se cargará con un valor de 1 la variable asociada (estado), indicando que una puerta se encuentra abierta

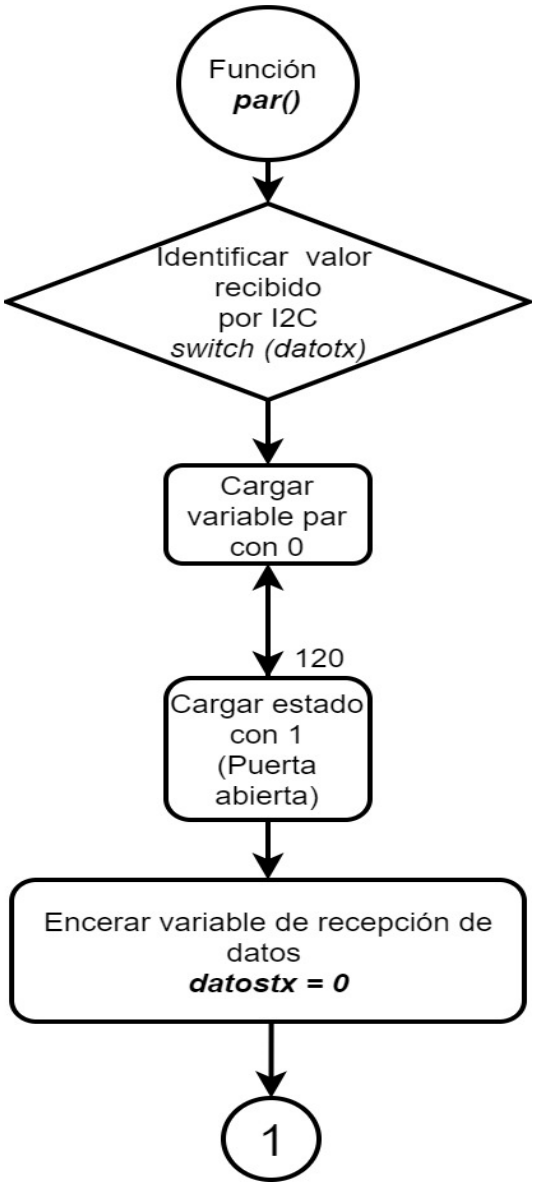



**Figura 3.8: Función Par**

En cambio, como se muestra en la figura 3.9, la función impar, cargará el valor de 1 a las variables de los *lockers* cuando conste que hay proyectores y del mismo modo a la función anterior, para indicar que la puerta está cerrada se procederá a cargar con un valor de 0 a su variable asociada.

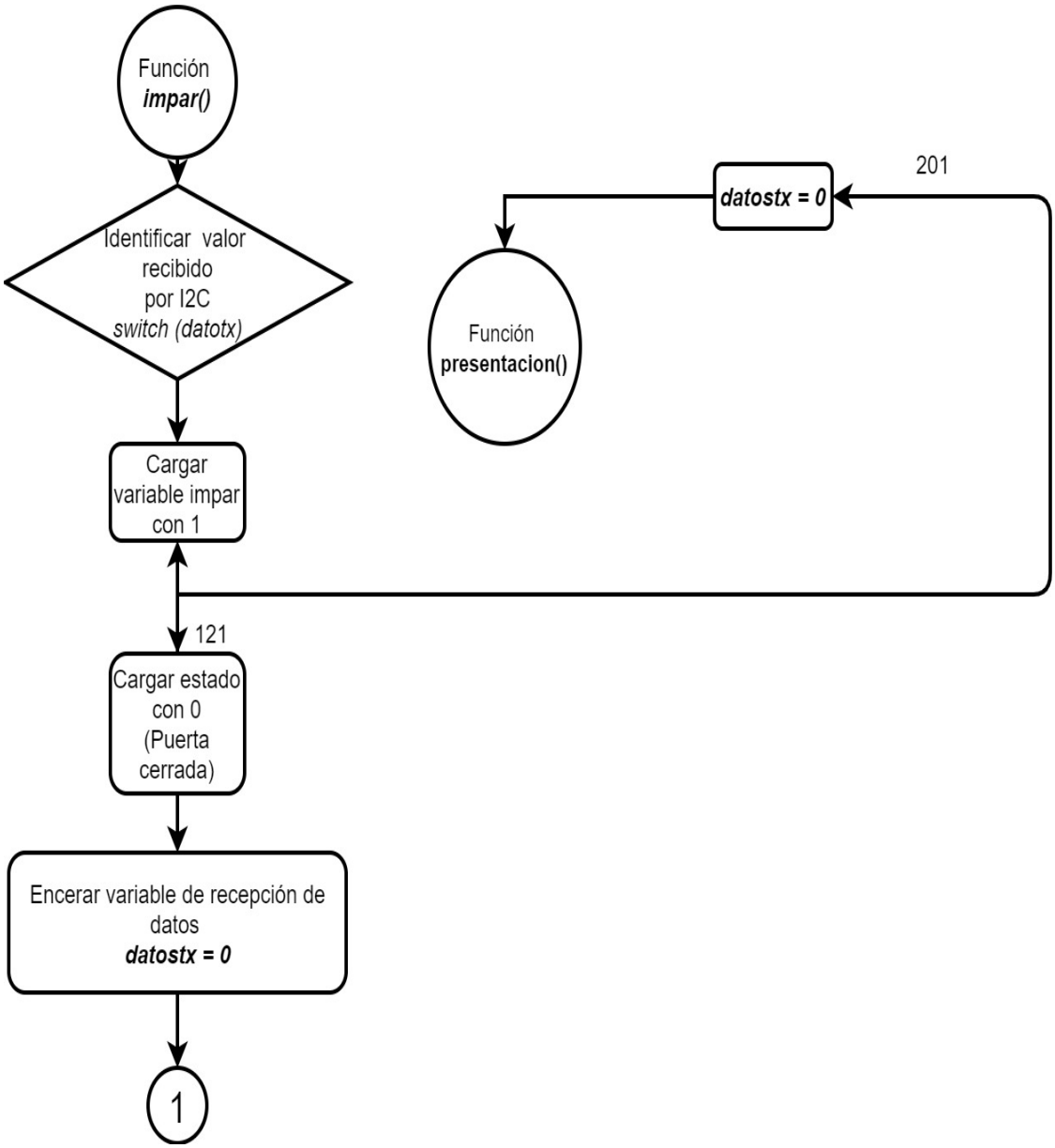



**Figura 3.9: Función Impar**

En adición, se tiene la figura 3.10, que muestra el conector denominado como “1”, el cual ocurrirá cuando no exista un *byte* disponible; en otras palabras, no se ha recibido el dato completo, en el conector se esperará 500ms para apagar el LED de ingreso de datos de los esclavos y se volverá a la función *void loop()* para la espera de nuevas peticiones.

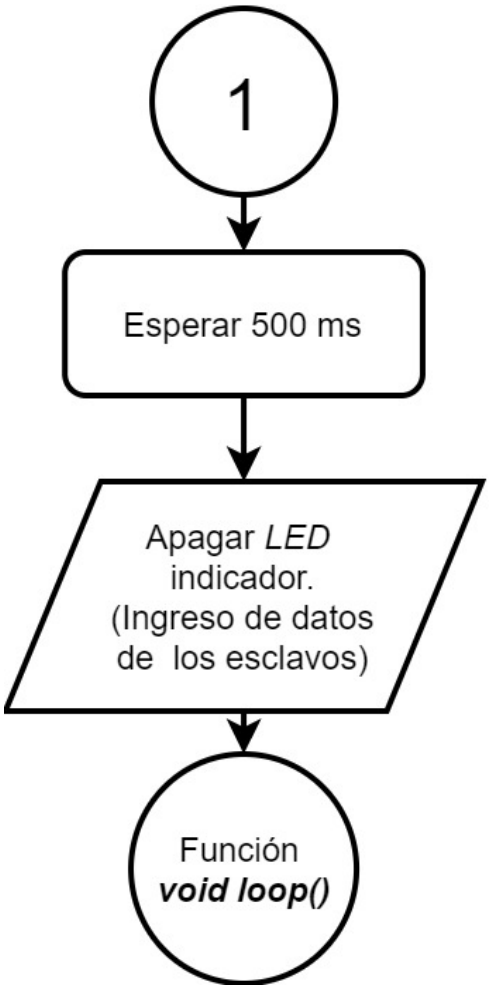



**Figura 3.10: Etapa "1"**

Para visualizar de manera conjunta el diagrama de flujo del Arduino maestro se adjunta el esquema en el Anexo A-1.

- **Arduino esclavo 1**

En la figura 3.11, se observa el mismo procedimiento que se realizó anteriormente en el Arduino maestro. Se incluyen librerías, se declaran variables y puertos de entrada y salida; además, se configura la interrupción por I2C y la comunicación serial. A parte, se observa la función *void loop ()*, que básicamente servirá para identificar el *locker* mediante el dato recibido. Al esclavo 1, se le han asignado los *lockers 1, 2, 5, 6, 9, 10, 13, 14, 17 y 18.* Si el dato recibido es 100, irá a la función inicio en la cual empieza a sensar los 10 *lockers* de forma consecutiva.

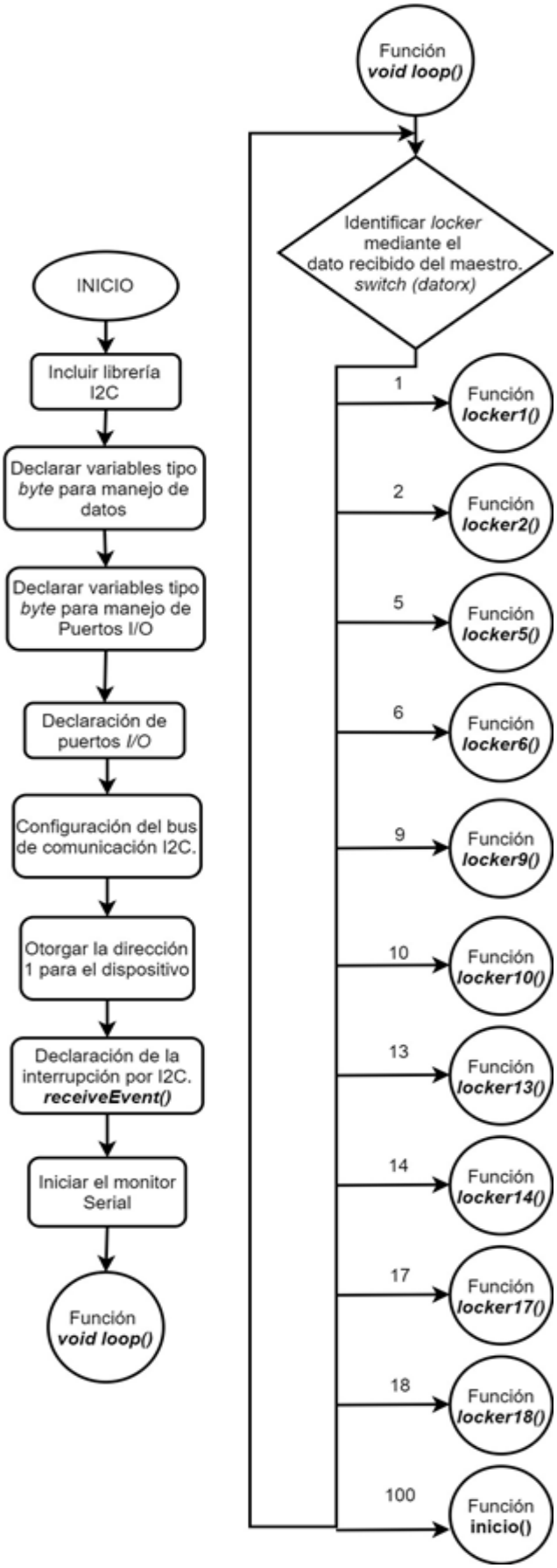



**Figura 3.11: Identificación de *lockers*, esclavo 1**

La función inicio, corresponde el primer estado de todo el sistema; como se aprecia en la figura 3.12, es la responsable de brindar la información de la presencia o ausencia del proyector por primera vez. Esta labor se lleva a cabo gracias a los sensores infrarrojos colocados en la estructura ya que en caso de que detecte a un proyector dentro del *locker*, se conmutará la luz verde; caso contrario, conmutará la luz roja. Este proceso se realizará a todos los *lockers* asignados al esclavo 1 y cuando concluya volverá a la función *void loop*.

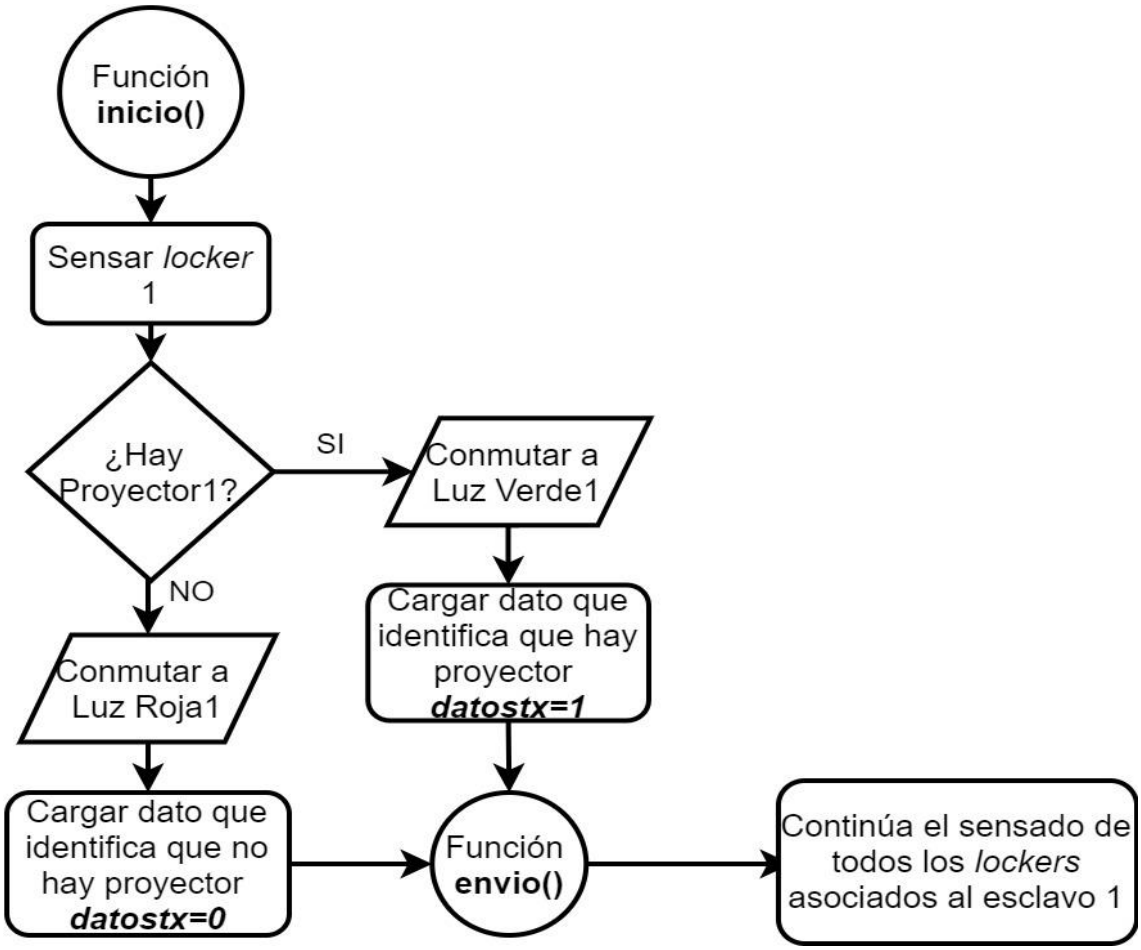



**Figura 3.12: Proceso del sensor infrarrojo y conmutación de luces**

En la figura 3.13 se observa la siguiente etapa del sistema que consiste en la parte en la que un usuario ha pedido un proyector o desea devolverlo. En ese caso, se realizará una conmutación en el módulo relé para abrir la cerradura eléctrica, inmediatamente el sensor magnético debe enviar el dato de si la puerta ha sido abierta para el retiro o devolución del proyector y pasado este proceso el sensor de contacto nuevamente deberá enviar el dato de si la puerta ha sido cerrada por el usuario ya que de este modo las luces conmutarán al color correspondiente al estado actual.

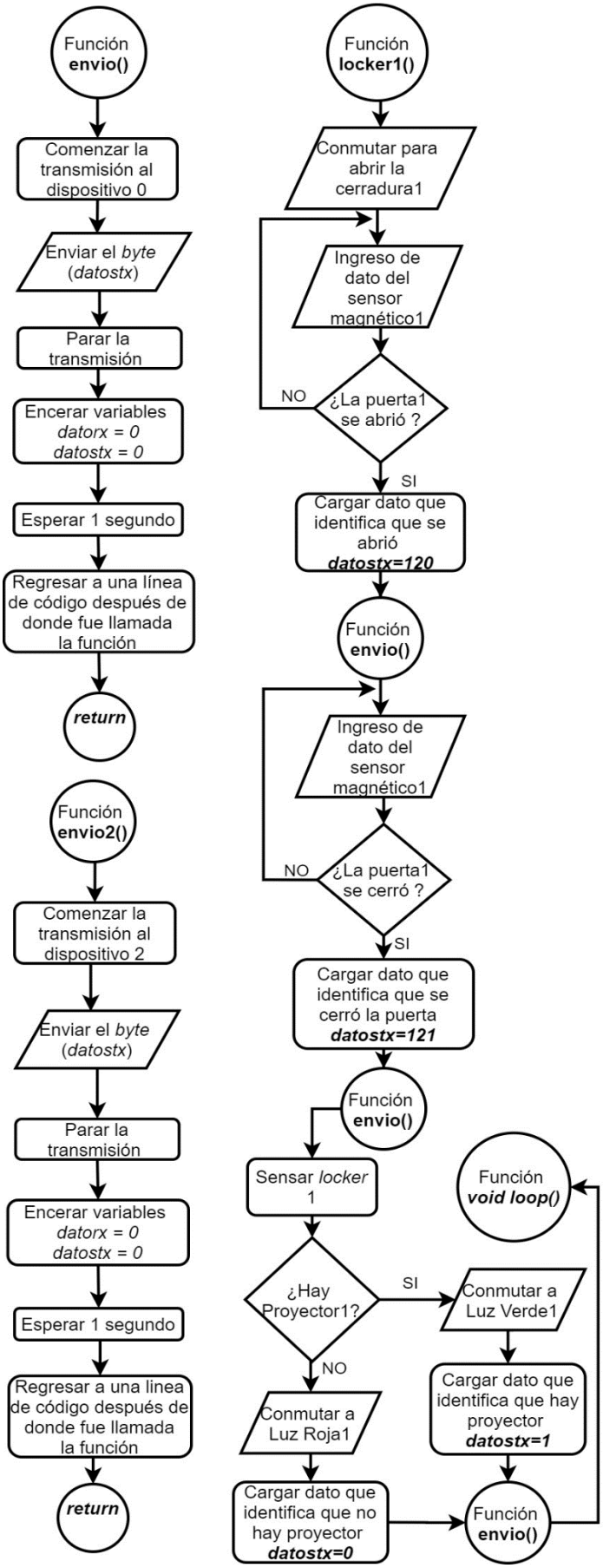



**Figura 3.13: Proceso de retiro o devolución de proyectores**

Este proceso ocurre en todos los *lockers* asignados al Arduino esclavo 1 y se puede observar en su totalidad en el Anexo A-2.

- **Arduino esclavo 2**

El proceso del Arduino esclavo 2 es exactamente el mismo al que se explicó en el Arduino esclavo 1. La diferencia radica en que para este esclavo se le han asignado los *lockers* 3, 4, 7, 8, 11, 12, 15, 16, 19 y 20; como se muestra en la figura 3.14.

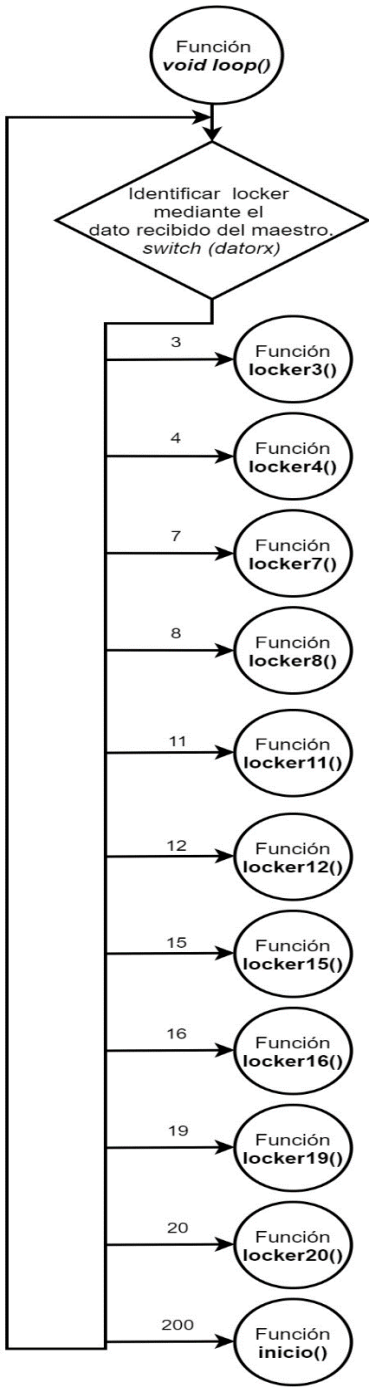



**Figura 3.14: Identificación de *lockers*, esclavo 2**

El diagrama de flujo total del esclavo 2 se presenta en el Anexo A-3.

## 3.3. Desarrollo del código para los dispositivos de control

Una vez que se concluyó con los esquemas, se procedió con la programación de los Arduinos elegidos utilizando variables y diferentes funciones. Hay que tener en cuenta que para esta parte se tuvo a la mano el *datasheet* correspondiente para analizar los puertos que se utilizaron para el manejo de los *lockers*. La programación se dividió en varios temas, entre los cuales se encuentran: Puertos de entrada y salida, Comunicación I2C, Comunicación serial y bucles de programación; donde cada uno tiene su propio código para comprobar su correcto funcionamiento y luego se unificaron en uno solo.

### Código del Arduino maestro

El desarrollo del código para el Arduino maestro se lo realizó en doce partes de importante relevancia; a continuación, se describe cada sección. Se puede revisar el código completo en el Anexo B-1.

- **Encabezado**

En la primera parte del programa, se agrega un pequeño encabezado tipo comentario para poder identificar el código correspondiente al dispositivo que se está programando. Además de incluir la única librería que se va a utilizar asociada al protocolo de comunicación I2C, como indica la figura 3.15.

```
/*
                    ESCUELA POLITÉCNICA NACIONAL - ESFOT
                          PROYECTO DE TITULACION
                            Control de Lockers
                                 Maestro
*/
//// LIBRERIAS
#include <Wire.h>       //Librería I2C
```

**Figura 3.15: Encabezado e inclusión de librería I2C**

- **Declaración de variables**

Se crean variables tipo *byte* debido a que se utilizarán valores no mayores a 255. Las tres primeras variables sirven para la comunicación I2C y serial, como se muestra en la figura 3.16.

```
//// VARIABLES
byte datotx = 0;        //Variable de enviar datos por i2c
byte datorx = 0;        //Variable para recibir datos por i2c
byte mensaje=0;         //Variable para recibir peticiones por comunicación
serial
```

**Figura 3.16: Variables para comunicación I2C y serial**

Las veinte siguientes, mostradas en la figura 3.17, son las variables donde se almacenará la información de la existencia o ausencia de proyectores, la variable "estado" almacenará la información de las puertas.

```
//Variables de estado de lockers (0 no hay proyector) (1 hay proyector):
byte var1=0;          byte var11=0;
byte var2=0;          byte var12=0;
byte var3=0;          byte var13=0;
byte var4=0;          byte var14=0;
byte var5=0;          byte var15=0;
byte var6=0;          byte var16=0;
byte var7=0;          byte var17=0;
byte var8=0;          byte var18=0;
byte var9=0;          byte var19=0;
byte var10=0;         byte var20=0;

byte estado=0;
```

**Figura 3.17: Variables de estado de *lockers***

Por último, la variable *"start"* es creada para el manejo del pulsador externo de inicio del sistema que se encuentra en el pin 5 del Arduino, como indica la línea de código de la figura 3.18.

```
byte start=5;          // Pin 5 del Arduino para pulsador de inicio
```

**Figura 3.18: Pulsador de inicio del sistema**

- **Declaración de Puertos I/O**

Se lo realiza en la función *void setup()* declarando tres pines (2, 3 y 4) como salidas digitales para colocar *LEDs* indicadores y el pin 5 como entrada activando su *pull-up* para el pulsador de inicio del sistema de control. Todo esto se muestra en la figura 3.19.

```
//// DECLARACION DE PUERTOS DE ENTRADA Y SALIDA
void setup() {
pinMode (2,OUTPUT);       // Led indicador cuando se transmite un dato al
esclavo1
pinMode (3,OUTPUT);       // Led indicador cuando se transmite un dato al
esclavo2
pinMode (4,OUTPUT);       // Led indicador cuando recibe un dato por i2c
pinMode (start,INPUT);    // Pulsador de inicio del sistema
digitalWrite(start,HIGH); // Activar el pull-up
```

**Figura 3.19: Declaración de puertos de entrada y salida**

- **Configuración del Protocolo de Comunicación I2C**

Esta configuración se lo realiza de igual modo en el *void setup()* en el cual primero se le da una dirección al dispositivo, en este caso el Arduino Maestro tiene la dirección cero. Después se declara la interrupción que se ejecutará cuando este dispositivo reciba datos y para finalizar se inicia el monitor serial con la configuración de los baudios a utilizar, tal y como se indica en la figura 3.20.

```
//// UNIR ESTE DISPOSITIVO AL BUS I2C CON DIRECCION 0
Wire.begin(0);
Wire.onReceive(receiveEvent); //Registrar el evento al recibir datos por
i2c
Serial.begin(9600);           //// Iniciamos el monitor serie para
monitorear la comunicación
```

**Figura 3.20: Dirección del Bus I2C**

- **Estado Inicial del sistema**

Como se puede observar en la figura 3.21, una vez que se haya configurado variables, puertos y protocolos de comunicación se procede con el estado inicial del sistema, en esta sección se espera a que se active el pulsador *"start".* Si no es pulsado, se queda en el bucle infinito *while*; pero si es activado, inmediatamente se ejecuta la función "inicio". Cabe recalcar que hasta aquí termina de ejecutarse el *void setup().*

```
//// ESTADO INICIAL DEL SISTEMA
while(digitalRead(start)==HIGH){}  //Esperar a que se presione el pulsador
de inicio
inicio();                          //Ir a la función inicio para sensar
todos los puestos
}
```

**Figura 3.21: Estado inicial del sistema**

- **Función “inicio”**

En la figura 3.22 se aprecia la función inicio donde se carga el valor de 100 en la variable de envío de datos del protocolo I2C y es enviado al dispositivo 1 por medio de la función “envio1”. El valor de 100 es interpretado por el esclavo 1 y empieza a sensar la presencia de proyectores.

```
//// FUNCION DE INICIO DEL SISTEMA
void inicio(){   // Función de inicio para que  el sistema empiece
sensando los lockers
datotx=100;      // Cargar  dato a transmitir al esclavo 1 para que
empiece a sensar
envio1();        // Llamar a la función de envío al esclavo 1
}
```

**Figura 3.22: Función de inicio del sistema**

- **Comunicación con el Servidor**

Se lo realiza en la función *void loop ().* En esta sección, mostrada en la figura 3.23, el Arduino maestro está constantemente a la espera de peticiones por parte del servidor con ciertos caracteres especiales descritos en la tabla 3.1. En un inicio se carga la variable “mensaje” con el caracter recibido y dependiendo su valor se ejecutarán las funciones “presentacion”, “seleccion1” o “seleccion2”.

```
//// COMUNICACIÓN CON EL SERVIDOR
void loop() {
  if(Serial.available()>0){        // Si se recibe un dato por comunicación
                                      serial con el servidor
     mensaje= Serial.read();       // Cargar la variable mensaje con el
dato recibido

// Si mensaje es u ir a la función presentación de variables
if(mensaje == 'u'){presentacion();}

//Si el mensaje es uno de los primeros 10 caracteres ir a seleccion1 (para
enviar dato al esclavo1)
     if(mensaje ==
'a'||'b'||'e'||'f'||'i'||'j'||'m'||'n'||'q'||'r'){seleccion1();}

//Si el mensaje es uno de los siguientes 10 caracteres ir a seleccion2
(para enviar dato al esclavo2)
     if(mensaje ==
'c'||'d'||'g'||'h'||'k'||'l'||'o'||'p'||'s'||'t'){seleccion2();}
  }
}
```

**Figura 3.23: Comunicación con el servidor**

**Tabla 3.1: Caracteres especiales para las peticiones de *lockers***

| CARACTER DEL SERVIDOR | *LOCKER* ASOCIADO | ESCLAVO ASOCIADO |
|---|---|---|
| **A** | 1 | Esclavo 1 |
| **B** | 2 | |
| **E** | 5 | |
| **F** | 6 | |
| **I** | 9 | |
| **J** | 10 | |
| **M** | 13 | |
| **N** | 14 | |
| **Q** | 17 | |
| **R** | 18 | |
| **C** | 3 | Esclavo 2 |
| **D** | 4 | |
| **G** | 6 | |
| **H** | 7 | |
| **K** | 11 | |
| **L** | 12 | |
| **O** | 15 | |
| **P** | 16 | |
| **S** | 19 | |
| **T** | 20 | |
| **U** | Presentación de las variables de disposición de *lockers* además de puertas abiertas o cerradas | |

- **Función “presentación”**

Si se recibe por comunicación serial el caracter “u” se ejecutará esta función. En esta sección, presentada en la figura 3.24 y 3.25, se envía la información de todas las variables al servidor por medio del *string* asociado a cada variable. Cada variable puede ser 0 o 1; para las 20 variables de los *lockers*, el 0 indica que no hay proyector y el 1 que sí lo hay. Por otro lado, en la variable “estado” el 1 indica que una puerta se encuentra abierta y el valor de 0 que está todo cerrado.

```
//// FUNCION DE PRESENTACION DE VARIABLES POR COMUNICACIÓN SERIAL
void presentacion(){  //Función para mostrar el estado de los lockers
                        (0 desocupado)   (1 ocupado)

// Presentar el estado de las puertas     (1 abierto) (0 cerrado)
   Serial.println((String)"Estado:"+estado);
   Serial.println((String)"Proyector1:"+var1);
   Serial.println((String)"Proyector2:"+var2);
   Serial.println((String)"Proyector5:"+var5);
```

**Figura 3.24: Función presentación de variables del esclavo 1**

```
//// FUNCION DE PRESENTACION DE VARIABLES POR COMUNICACIÓN SERIAL
void presentacion(){   //Función para mostrar el estado de los lockers
                         (0 desocupado)    (1 ocupado)

// Presentar el estado de las puertas      (1 abierto) (0 cerrado)
   Serial.println((String)"Estado:"+estado);
   Serial.println((String)"Proyector3:"+var3);
   Serial.println((String)"Proyector4:"+var4);
   Serial.println((String)"Proyector7:"+var7);
   Serial.println((String)"Proyector8:"+var8);
   Serial.println((String)"Proyector11:"+var11);
   Serial.println((String)"Proyector12:"+var12);
   Serial.println((String)"Proyector15:"+var15);
   Serial.println((String)"Proyector16:"+var16);
   Serial.println((String)"Proyector19:"+var19);
   Serial.println((String)"Proyector20:"+var20);
```

**Figura 3.25: Función presentación de variables del esclavo 2**

Una vez que se ha enviado toda la información, se finaliza la comunicación serial con el servidor; después, se inicializa el serial para la espera de nuevos datos que pueden mandar los esclavos y se encera la variable “mensaje”, como se indica en las líneas de código de la figura 3.26.

```
// Finalizar la comunicación serial después de enviar todas las variables al
servidor
Serial.end();
// Inicializar el serial para espera de datos por i2c
Serial.begin(9600);
delay(250);           // Pausa de 250 milisegundos
mensaje = 0;          // Encerar datos recibidos del servidor
       }
```

**Figura 3.26: Finalización de la comunicación serial**

- **Función “seleccion1”**

Si la petición del servidor fue uno de los 10 caracteres asociados al esclavo 1, se ejecutará esta función. Dependiendo del valor del caracter recibido, se selecciona el *locker* correspondiente cargando un código único de identificación de dicho *locker* para ser enviado al esclavo 1. En la tabla 3.2 se puede observar los valores únicos para su respectiva interpretación en el esclavo 1 y en la figura 3.27 se observan las líneas de código de algunos *lockers* del esclavo 1 que se han tomado como muestra.

```
//// FUNCION DE PETICION DE UNO DE LOS LOCKERS DEL ESCLAVO 1
void seleccion1(){     // Función
switch (mensaje) {     // Comparar a que caso pertenece el dato recibido del
servidor
      case 'a':        // Locker 1
      datotx=1;        // Cargar número único que identifica el locker 1
      envio1();        // Enviar dato al esclavo 1
      break;

      case 'b':        // Locker 2
      datotx=2;        // Cargar número único que identifica el locker 2
      envio1();        // Enviar dato al esclavo 1
      break;

      case 'e':        // Locker 5
      datotx=5;        // Cargar número único que identifica el locker 5
      envio1();        // Enviar dato al esclavo 1
      break;
      }
```

**Figura 3.27: Función petición "1" de lockers esclavo 1**

**Tabla 3.2: Caracteres para la identificación de *lockers* del esclavo 1**

| | CARACTER DEL SERVIDOR | *LOCKER* ASOCIADO | DATO QUE ENVÍA EL MAESTRO AL ESCLAVO 1 |
|---|---|---|---|
| ESCLAVO 1 | a | 1 | 1 |
| | b | 2 | 2 |
| | e | 5 | 5 |
| | f | 6 | 6 |
| | i | 9 | 9 |
| | j | 10 | 10 |
| | m | 13 | 13 |
| | n | 14 | 14 |
| | q | 17 | 17 |
| | r | 18 | 18 |

En otras palabras, el maestro manda al esclavo 1 el número del *locker* asociado.

- **Función "peticion2"**

Es muy similar a la función "seleccion1" con la particularidad de que aquí se envía el número de los *lockers* restantes asociados al esclavo 2. En la figura 3.28 se observan las líneas de código de algunos *lockers* del esclavo 2 que se han tomado como muestra. En la tabla 3.3 se puede observar los valores únicos para su respectiva interpretación en el esclavo 2.

```
//// FUNCION DE PETICION DE UNO DE LOS LOCKERS DEL ESCLAVO 2
void seleccion2(){     // Función
switch (mensaje) {     // Comparar a que caso pertenece el dato recibido del
servidor
      case 'c':        // Locker 3
      datotx=3;        // Cargar número único que identifica el locker 3
      envio2();        // Enviar dato al esclavo 2
      break;

      case 'd':        // Locker 4
      datotx=4;        // Cargar número único que identifica el locker 4
      envio2();        // Enviar dato al esclavo 2
      break;

      case 'g':        // Locker 7
      datotx=7;        // Cargar número único que identifica el locker 7
      envio2();        // Enviar dato al esclavo 2
      break;
```

**Figura 3.28: Función petición "2" de lockers esclavo 1**

**Tabla 3.3: Caracteres para la identificación de *lockers* del esclavo 2**

| | CARACTER DEL SERVIDOR | *LOCKER* ASOCIADO | DATO QUE ENVÍA EL MAESTRO AL ESCLAVO 2 |
|---|---|---|---|
| ESCLAVO 2 | C | 3 | 3 |
| | D | 4 | 4 |
| | G | 6 | 6 |
| | H | 7 | 7 |
| | K | 11 | 11 |
| | L | 12 | 12 |
| | O | 15 | 15 |
| | P | 16 | 16 |
| | S | 19 | 19 |
| | T | 20 | 20 |

- **Interrupción por I2C**

La función "*receiveEvent*" se ejecuta cada vez que se recibe un dato por el bus I2C, sin importar donde se encuentre el puntero de código ya que a esta sección se la trabaja como cualquier otra función; esto quiere decir que, después de finalizar regresa una línea después del código en donde se quedó el puntero. En esta sección primero se indica por medio de un LED que se recibió un dato por I2C. Posteriormente, se lee y se carga la variable de recepción de datos asociada al protocolo para que este sea verificado siendo par o impar.

Es muy importante entender la discriminación de datos que se realiza en esta sección de código presentada en la figura 3.29, ya que, por medio de esto, se cargan las variables de presentación con 0 y 1, recordando que con 0 se indica que no hay proyector y con 1 que sí existe. En otras palabras, si se recibe un número par de los esclavos, el maestro deduce que ese dato es de cualquiera de los 20 *lockers,* pero con la particularidad de que dicho dato ya indica que no hay proyector. Para poder saber a qué *locker* pertenece y poder cargar su variable con 0, se ejecuta la función “par”.

Del mismo modo, ocurre con la función “impar”, ya que se ejecutará cuando el dato recibido de cualquiera de los 20 *lockers* sea impar para posteriormente identificar a qué locker pertenece y cargar con 1 su variable asociada. Una vez que se hayan ejecutado las funciones “par” o “impar”, se regresa a la sección de código para realizar una espera de medio segundo, apagar el LED indicador y regresar a la parte de código en donde se encontraba el puntero que en la mayoría de los casos se encontrará en el *void loop.*

```
////FUNCION CUANDO SE RECIBE UN DATO POR I2C
void receiveEvent() {
digitalWrite(4,HIGH); //Encender led indicador de que se recibió un dato de
los esclavos
if (Wire.available() == 1){     // Si hay un byte disponible
    datorx = Wire.read();       // Leer  y cargar la variable
    if ( datorx % 2 == 0){      // Verificar si el dato que se recibió es par
o impar
        par();      // Si es par, ir a la función par para indicar que no hay
casilleros
    }
    else{
        impar();// Si es impar, ir a la función impar para indicar que si hay
casilleros
    }
}
delay(500);             // Pausa de medio segundo
digitalWrite(4,LOW);    // Apagar led indicador de que se recibió un dato de
los esclavos
```

**Figura 3.29: Interrupción por I2C**

Es importante mencionar que en la función “par” es donde se carga la variable “estado” con valor 1 indicando que una puerta está abierta, y que en la función “impar” esta variable será 0 indicando que todas las puertas están cerradas. Estas dos funciones se pueden apreciar en las figuras 3.30 a la 3.34; además la función “impar” es la encargada de interpretar el dato 201 que indica que el esclavo 2 terminó de sensar sus 10 *lockers* por primera vez y el maestro procede a ejecutar la presentación de las variables.

```
////FUNCION PARA CARGAR VARIABLES DE PRESENTACION CON EL SERVIDOR (SI NO HAY
PROYECTORES Y PUERTAS ABIERTAS)

void par(){
switch (datorx) {  //Identificar valor del dato que se recibió por i2c para
cargar con 0 la variable de presentación de cada locker
        ////////////////////////////////////1
        case 0:
        var1=0;
        datorx=0;
        break;
        ////////////////////////////////////2
        case 2:
        var2=0;
        datorx=0;
        break;
        ////////////////////////////////////3
```

**Figura 3.30: Función par (muestra de 3 *lockers*)**

```
////Estado=1 cuando una puerta está abierta
        case 120:
        estado=1;
        datorx=0;
        break;

   }
```

**Figura 3.31: Estado puerta abierta en función par**

```
//// FUNCION PARA CARGAR VARIABLES DE PRESENTACION CON EL SERVIDOR (SI HAY
PROYECTORES Y PUERTAS CERRADAS), ADEMAS DEL DATO DE FINILIZACION DE LA
PRIMERA VEZ QUE SE SENSA.

void impar(){
switch (datorx) {   //Identificar valor del dato que se recibió por i2c para
cargar con 1 la variable de presentación de cada locker
          ////////////////////////////////////1
          case 1:
          var1=1;
          datorx=0;
          break;
          ////////////////////////////////////2
          case 3:
          var2=1;
          datorx=0;
          break;
          ////////////////////////////////////3
          case 5:
          var3=1;
          datorx=0;
          break;
```

**Figura 3.32: Función impar (muestra de 3 *lockers*)**

```
////Estado=0 todo está cerrado
          case 121:
          estado=0;
          datorx=0;
          break;

   }
}
```

**Figura 3.33: Estado puerta abierta en función impar**

```
////Dato que envía el esclavo2 cuando termina de sensar por primera vez
          case 201:
          datorx=0;

////Presentar todas las variables del estado de los lockers por primera vez

          presentación ();
          break;
}
}
```

**Figura 3.34: Dato de finalización de la primera vez que se sensa**

Las tablas 3.4 y 3.5 muestran una mejor interpretación de los datos que puede recibir el maestro por parte de los esclavos.

**Tabla 3.4: Presencia o ausencia de proyectores, esclavo 1 y 2**

| ESCLAVO | *LOCKER* | DATO QUE ENVÍA EL ESCLAVO AL MAESTRO | DESCRIPCIÓN |
|---|---|---|---|
| **1** | 1 | 0 | No hay proyector |
| | | 1 | Sí hay proyector |
| | 2 | 2 | No hay proyector |
| | | 3 | Sí hay proyector |
| | 5 | 8 | No hay proyector |
| | | 9 | Sí hay proyector |
| | 6 | 10 | No hay proyector |
| | | 11 | Sí hay proyector |
| | 9 | 16 | No hay proyector |
| | | 17 | Sí hay proyector |
| | 10 | 18 | No hay proyector |
| | | 19 | Sí hay proyector |
| | 13 | 24 | No hay proyector |
| | | 25 | Sí hay proyector |
| | 14 | 26 | No hay proyector |
| | | 27 | Sí hay proyector |
| | 17 | 32 | No hay proyector |
| | | 33 | Sí hay proyector |
| | 18 | 34 | No hay proyector |
| | | 35 | Sí hay proyector |

| ESCLAVO | *LOCKER* | DATO QUE ENVÍA EL ESCLAVO AL MAESTRO | DESCRIPCIÓN |
|---|---|---|---|
| **2** | 3 | 4 | No hay proyector |
| | | 5 | Sí hay proyector |
| | 4 | 6 | No hay proyector |
| | | 7 | Sí hay proyector |
| | 7 | 12 | No hay proyector |
| | | 13 | Sí hay proyector |
| | 8 | 14 | No hay proyector |
| | | 15 | Sí hay proyector |
| | 11 | 20 | No hay proyector |
| | | 21 | Sí hay proyector |
| | 12 | 22 | No hay proyector |
| | | 23 | Sí hay proyector |
| | 15 | 28 | No hay proyector |
| | | 29 | Sí hay proyector |
| | 16 | 30 | No hay proyector |
| | | 31 | Sí hay proyector |
| | 19 | 36 | No hay proyector |
| | | 37 | Sí hay proyector |
| | 20 | 38 | No hay proyector |
| | | 39 | Sí hay proyector |
| | - | 201 | Finalizó el sensado de los últimos 10 *lockers* del inicio. |

**Tabla 3.5: Estados de las puertas de los *lockers***

| DATOS QUE ENVÍAN LOS ESCLAVOS DE CUALQUIER *LOCKER* | ESTADO DE PUERTA |
|---|---|
| **120** | Abierta |
| **121** | Cerrada |

- **Funciones de envío de datos por I2C**

Para poder enviar datos a los esclavos se crearon dos funciones, la primera es “envio1” y corresponde para el esclavo 1 y la segunda es “envio2” y corresponde para el esclavo 2, como se aprecia en las figuras 3.35. Estas dos funciones básicamente realizan lo mismo; primero se enciende el *LED* indicador correspondiente al esclavo, segundo se

ingresa la dirección del dispositivo al que se va a enviar el dato, el esclavo uno tiene la dirección 1 y el esclavo dos tiene la dirección 2. Después, se envía la variable con el dato que se cargó con anterioridad y una vez finalizado el envío se para la transmisión. Para finalizar se enceran variables, se da una pausa y finalmente se apaga el *LED* indicador de envío de datos.

```
////FUNCIONES DE ENVIO DE DATOS A LOS ESCLAVOS

void envio1(){      // FUNCION PARA ENVIAR DATOS AL ESCLAVO 1


    digitalWrite(2,HIGH);              // Encender led indicador de envío de
datos al esclavo 1
    Wire.beginTransmission(1);         // Comenzar la transmisión al
dispositivo 1
    Wire.write(datotx);                // Enviar un byte
    Wire.endTransmission();            // Parar la transmisión
    mensaje=0;                         // Encerar variables
    datotx=0;
    delay(1000);                       // Esperar 1 segundo
    digitalWrite(2,LOW);               // Apagar led indicador de envío de
datos al esclavo 1
}

void envio2(){                      // FUNCION PARA ENVIAR DATOS AL ESCLAVO 2


    digitalWrite(3,HIGH);              // Encender led indicador de envío de
datos al esclavo 2
    Wire.beginTransmission(2);         // Comenzar la transmisión al
dispositivo 2
    Wire.write(datotx);                // Enviar un byte
    Wire.endTransmission();            // Parar la transmisión
    mensaje=0;                         // Encerar variables
    datotx=0;
    delay(1000);                       // Esperar 1 segundo
    digitalWrite(3,LOW);               // Apagar led indicador de envío de
datos al esclavo 2
}
```

**Figura 3.35: Función de envío de datos a los esclavos**

## Código del Arduino esclavo 1

El Código de programación del esclavo 1, de la misma manera que en el maestro, se desarrolló por secciones, y que a continuación son descritas. El código completo se encuentra en el Anexo B-2.

- **Encabezado**

En la figura 3.36 muestra un pequeño título tipo comentario para poder identificar el código que se está desarrollando. Posteriormente, se empieza agregando la librería del Protocolo de Comunicación I2C.

```
/*
                    ESCUELA POLITÉCNICA NACIONAL - ESFOT
                          PROYECTO DE TITULACION
                            Control de Lockers
                                Esclavo 1
*/


//// LIBRERIAS
#include <Wire.h>
```

**Figura 3.36: Encabezado y librerías esclavo 1**

- **Declaración de Variables**

Se crean variables tipo *byte* debido a que no se van a exceder valores a 255. Las primeras dos variables se crean para enviar y recibir datos por medio del bus I2C. Las dos variables siguientes se crean para sensar los proyectores en cada *locker*, la variable “a” para crear un tiempo de conteo y la variable “respuesta” para guardar el dato de los sensores infrarrojos. Finalmente, se crea variables para poder especificar los pines del Arduino en la declaración de puertos I/O. Esto se realiza para un mejor manejo de los pines en el desarrollo del código; todas estas variables se pueden ver en las figuras 3.37 a la 3.39.

```
//// VARIABLES

byte datorx = 0;  //Variable para recibir datos por i2c
byte datostx=0;   //Variable de enviar datos por i2c
```

**Figura 3.37: Variables para envío y recepción de datos**

```
byte a=0;              //Variable a para establecer un contador (tiempo
para sensar con infrarrojo)
byte respuesta=0;      //Variable respuesta para leer datos de sensor
infrarrojo
```

**Figura 3.38: Variable contador y variable respuesta**

```
//Variables para manejar los puertos I/O con nombres propios para cada
locker:
//////////////////////////////////////////////////////////Locker 1
byte magnetopin1=4;     //Lectura del sensor magnético
byte infrapin1=53;      //Lectura del sensor infrarrojo
byte ledpin1=25;        //Activación de luces
byte cerradurapin1=28; //Activación de cerradura
byte activepin1=36;     //Activación del infrarrojo
//////////////////////////////////////////////////////////Locker 2
byte magnetopin2=5;     //Lectura del sensor magnético
byte infrapin2=51;      //Lectura del sensor infrarrojo
byte ledpin2=26;        //Activación de luces
byte cerradurapin2=27; //Activación de cerradura
byte activepin2=37;     //Activación del infrarrojo
```

**Figura 3.39: Manejo de puertos I/O para esclavo 1 (muestra de 2 *lockers*)**

En la tabla 3.6, se puede observar una mejor descripción y distribución de todos los pines del Arduino. Cabe recalcar que la distribución de pines no se realizó de manera secuencial debido al diseño de la tarjeta madre de la estructura general.

**Tabla 3.6: Distribución de pines del Arduino esclavo 1**

| PIN | ENTRADA /SALIDA | DESCRIPCIÓN | MÓDULO RELÉ | PINES MÓDULO RELÉ | *LOCKER* |
|---|---|---|---|---|---|
| **0** | - | - | - | - | - |
| **1** | - | - | - | - | - |
| **2** | Entrada | Datos sensor infrarrojo | - | - | 17 |
| **3** | Entrada | Datos sensor infrarrojo | - | - | 18 |
| **4** | Entrada | Sensor Magnético | - | - | 1 |
| **5** | Entrada | Sensor Magnético | - | - | 2 |
| **6** | Entrada | Sensor Magnético | - | - | 5 |
| **7** | Entrada | Sensor Magnético | - | - | 6 |
| **8** | Entrada | Sensor Magnético | - | - | 9 |
| **9** | Entrada | Sensor Magnético | - | - | 10 |
| **10** | Entrada | Sensor Magnético | - | - | 13 |
| **11** | Entrada | Sensor Magnético | - | - | 14 |
| **12** | Entrada | Sensor Magnético | - | - | 17 |
| **13** | Entrada | Sensor Magnético | - | - | 18 |
| **14** | Entrada | Datos sensor infrarrojo | - | - | 14 |
| **15** | Entrada | Datos sensor infrarrojo | - | - | 13 |
| **16** | Entrada | Datos sensor infrarrojo | - | - | 10 |
| **17** | Entrada | Datos sensor Infrarrojo | - | - | 9 |

| PIN | ENTRADA /SALIDA | DESCRIPCIÓN | MÓDULO RELÉ | PINES MÓDULO RELÉ | *LOCKER* |
|---|---|---|---|---|---|
| **18** | Entrada | Datos sensor infrarrojo | - | - | 6 |
| **19** | Entrada | Datos sensor infrarrojo | - | - | 5 |
| **20** | - | - | - | - | - |
| **21** | - | - | - | - | - |
| **22** | Salida | Luces | 1 | 16 | 9 |
| **23** | Salida | Luces | 1 | 15 | 5 |
| **24** | Salida | Luces | 1 | 14 | 6 |
| **25** | Salida | Luces | 1 | 13 | 1 |
| **26** | Salida | Luces | 1 | 12 | 2 |
| **27** | Salida | Cerradura | 1 | 11 | 2 |
| **28** | Salida | Cerradura | 1 | 10 | 1 |
| **29** | Salida | Cerradura | 1 | 9 | 6 |
| **30** | Salida | Cerradura | 1 | 8 | 5 |
| **31** | Salida | Cerradura | 1 | 8 | 9 |
| **33** | Salida | Polarización sensor Infrarrojo | 1 | 5 | 9 |
| **34** | Salida | Polarización sensor Infrarrojo | 1 | 5 | 5 |
| **35** | Salida | Polarización sensor Infrarrojo | 1 | 3 | 6 |
| **36** | Salida | Polarización sensor Infrarrojo | 1 | 2 | 1 |
| **37** | Salida | Polarización sensor Infrarrojo | 1 | 1 | 2 |
| **38** | Salida | Luces | 2 | 16 | 17 |
| **39** | Salida | Luces | 2 | 15 | 18 |
| **40** | Salida | Luces | 2 | 14 | 13 |
| **41** | Salida | Luces | 2 | 13 | 14 |
| **42** | Salida | Luces | 2 | 12 | 10 |
| **43** | Salida | Cerradura | 2 | 11 | 10 |
| **44** | Salida | Cerradura | 2 | 10 | 13 |
| **45** | Salida | Cerradura | 2 | 9 | 14 |
| **46** | Salida | Cerradura | 2 | 8 | 17 |
| **47** | Salida | Cerradura | 2 | 7 | 18 |
| **48** | Salida | Polarización sensor Infrarrojo | 2 | 6 | 13 |
| **49** | Salida | Polarización sensor Infrarrojo | 2 | 5 | 18 |
| **50** | Salida | Polarización sensor Infrarrojo | 2 | 4 | 17 |
| **51** | Entrada | Datos sensor Infrarrojo | - | - | 2 |
| **52** | Salida | Led indicador | - | - | Placa |
| **53** | Entrada | Datos sensor infrarrojo | - | - | 1 |

- **Declaración de Puertos I/O**

Se lo realiza en la función *setup* del Arduino. Para un mejor entendimiento, se lo separó por bloques tipo comentario según corresponda a cada *locker*. Además, en lugar de utilizar las variables descritas anteriormente para poder declararlas como salidas o como entradas, se activan los *pull-up* en todas las entradas debido a que se necesita detectar el cero lógico que los sensores mandan como respuesta al Arduino. Tanto el sensor magnético como el sensor infrarrojo emiten cero lógicos al momento en que se abre una puerta (magnético) y cuando detecta un objeto (infrarrojo). De igual manera, se activa el *pull-up* en las salidas ya que para activar un relé se debe mandar un cero lógico desde el dispositivo de control (Arduino) hasta los pines del módulo de los 16 relés. Finalmente, se declara el pin 52 como salida para colocar un *LED* indicador cuando se reciba un dato por I2C, esto se puede observar en la figura 3.40.

```
////DECLARACION DE PUERTOS DE ENTRADA Y SALIDA
void setup() {
////////////////////////////////////////////////////////Locker 1
pinMode (ledPin1,OUTPUT);
pinMode (cerraduraPin1,OUTPUT);
pinMode (activePin1,OUTPUT);
digitalWrite (ledPin1,HIGH);
digitalWrite (cerraduraPin1,HIGH);
digitalWrite (activePin1,HIGH);
pinMode (infraPin1,INPUT);
pinMode (magnetoPin1,INPUT);
digitalWrite (infraPin1,HIGH);
digitalWrite (magnetoPin1,HIGH);

pinMode (52,OUTPUT);          //Led indicador cuando recibe un dato por
i2c
```

**Figura 3.40: Declaración de puertos de entrada y salida, esclavo 1**

- **Protocolo de Comunicación I2C**

La configuración del bus I2C se lo realiza en el *setup,* como se evidencia en la figura 3.41. Primero se da la dirección 1 al dispositivo, después se registra la interrupción cuando se recibe un dato por I2C y finalmente se inicializa el monitor serial con la configuración de los baudios.

```
//// UNIR ESTE DISPOSITIVO AL BUS I2C CON DIRECCION 1
Wire.begin(1);
Wire.onReceive(receiveEvent); // Registrar el evento al recibir datos por
i2c
Serial.begin(9600);           // Iniciar el monitor serie para monitorear
la comunicación
}
```

**Figura 3.41: Comunicación I2C, esclavo 1**

- **Petición de *Locker***

Una vez configurados los puertos y variables necesarios, el Arduino ingresa al *void loop* y se queda constantemente en esta parte de código, a la espera de una petición de uno de los *lockers* que controla, por parte del Arduino maestro (Uno). Es importante mencionar que al momento que ingresa un dato por I2C, primero se cumple la estructura de código de su correspondiente interrupción por I2C para después ese dato recibido y almacenado en la variable "datorx" pueda ser comparado en el *switch* que se encuentra dentro del *void loop*. Los primeros diez casos corresponden a los diez *lockers* que controla este Arduino y si "datorx" corresponde a uno de estos casos, el código continúa en la respectiva función asociada a cada *locker* para poder realizar la petición. El último caso corresponde al dato inicial que manda el maestro para que el esclavo 1 empiece a sensar sus 10 *lockers*, mandando la información de cada uno al maestro por I2C, como se establece en la figura 3.42.

```
////IDENTIFICACION DE LOCKER MEDIANTE EL DATO RECIBIDO POR PARTE DEL
MAESTRO
void loop(){                  // Funcion loop
  switch (datorx){            // Comparar a que caso pertenece el dato
recibido del maestro
  case 1:                     // Número único que identifica el locker 1
    locker1();                // Ir a la función de petición del locker 1
    break;
  case 2:                     // Número único que identifica el locker 2
    locker2();                // Ir a la función de petición del locker 2
    break;
  case 5:                     // Número único que identifica el locker 5
    locker5();                // Ir a la función de petición del locker 5
    break;
  case 6:                     // Número único que identifica el locker 6
    locker6();                // Ir a la función de petición del locker 6
    break;
  case 9:                     // Número único que identifica el locker 9
    locker9();                // Ir a la función de petición del locker 9
    break;
  case 10:                    // Número único que identifica el locker 10
    locker10();               // Ir a la función de petición del locker 10
    break;
  case 13:                    // Número único que identifica el locker 13
    locker13();               // Ir a la función de petición del locker 13
    break;
  case 14:                    // Número único que identifica el locker 14
    locker14();               // Ir a la función de petición del locker 14
    break;
  case 17:                    // Número único que identifica el locker 17
    locker17();               // Ir a la función de petición del locker 17
    break;
  case 18:                    // Número único que identifica el locker 18
    locker18();               // Ir a la función de petición del locker 18
    break;
  case 100:                   // Dato que se recibe para sensar todos los
lockers.
    inicio();                 // Ir a la función inicio para sensar todos
los puestos
    break;
```

**Figura 3.42: Identificación de los *lockers* del esclavo 1**

- **Sensado de los diez *lockers* asociados al esclavo 1**

Si el dato que se recibió por I2C es 100, el esclavo 1 interpreta que el maestro requiere saber la información de sus 10 *lockers*; en otras palabras, el esclavo 1 sensa todos los 10 puestos y manda la información de cada uno al maestro si está ocupado o desocupado. Para realizar esto, se ejecuta la función inicio, en donde se sensa cada puesto; uno por uno se va activando los sensores infrarrojos, sensan por un tiempo de un segundo mientras guardan la respuesta del sensor en la variable "respuesta". Si existe un proyector, la respuesta es cero y si no existe un proyector la respuesta es 1. Después, se apaga el sensor infrarrojo y se procede a comparar la respuesta que se obtuvo para continuar con la conmutación de las luces (verde si hay proyector y rojo si no lo hay) y después se manda el número único de identificación que será interpretado por el maestro para presentar los datos al servidor. Esto se realiza en los 10 *lockers* que controla este Arduino y al finalizar, antes de acabar con la función inicio, el esclavo 1 manda al esclavo 2 el valor de 200 para que el dispositivo 2 empiece a sensar los 10 *lockers* que controla, como se indica en la figura 3.43. La interpretación de los números únicos que identifica cada *locker* ya se la describió en la sección de la interrupción por I2C en la parte del maestro, específicamente en las funciones par e impar.

```
//// FUNCION PARA SENSAR TODOS LOS LOCKERS
void inicio(){
////Locker 1
  digitalWrite(activePin1,LOW);             // Activar sensor
infrarrojo
  for(a=0;a<10000;a++){                     // Empieza a sensar  el
infrarrojo
      respuesta=digitalRead(infraPin1);}    // Guardar el valor en la
variable respuesta (0 o 1)
  digitalWrite(activePin1,HIGH);            // Apagar sensor
infrarrojo
  if(respuesta==LOW){                       // Si es 0 detecta proyector
      digitalWrite(ledPin1,LOW);            // Encender luz verde
      datostx=1;                           // Cargar dato que identifica que
hay proyector
      envio();}                            // Enviar dato que si hay
proyector al maestro
  else{                                     // Si es 1 no hay proyector
      digitalWrite(ledPin1,HIGH);           // Encender luz roja
      datostx=0;                         // Cargar dato que identifica que no
hay proyector
      envio();}                             // Enviar dato que no hay
proyector al maestro
//// Esclavo 2
datostx=200          // Cargar dato a transmitir al esclavo 2 para que
empiece a sensar
envio2();            // Llamar a la función de envío al esclavo 2
```

**Figura 3.43: Función para sensar los *lockers* del esclavo 1**

- **Funciones de Petición de cada *locker***

Si el dato que se recibió por I2C corresponde a uno de los casos del *void loop,* a excepción del dato de inicio, quiere decir que el maestro desea realizar una petición de uno de los *lockers* y según dependa el valor o número único de identificación de cada casillero, ingresará a su función correspondiente. Cada *locker* tiene su respectiva función muy similar a la función de inicio con la excepción de que aquí se incluye el funcionamiento del sensor de contacto magnético y de la cerradura eléctrica. En cada función primero se va a abrir la cerradura, y se quedará en un bucle hasta que la puerta sea abierta; si la puerta no es abierta, el programa no continúa y no se podrán realizar otras peticiones. Una vez que se abrió la puerta, el sensor magnético envía cero lógicos para que el programa salga del bucle *while*. Posteriormente, se envía el dato 120 al maestro que es un dato único para todos los *lockers* para indicar al maestro que su puerta se encuentra abierta. Es aquí donde el usuario procede a retirar su proyector y el código no continuará mientras no se cierre la puerta del *locker*, esto se expresa en el segundo bucle *while*, en donde se espera que la respuesta del sensor magnético sea 1. Cuando se ha cerrado la puerta con la cerradura correspondiente a ese *locker*, se manda el dato 121 al maestro para avisar que la puerta ya ha sido cerrada. Finalmente, una vez que se cerró la puerta, el programa continúa el proceso de detección del proyector; es decir que, se activa el infrarrojo y empieza a sensar, se cumple el proceso que se describió en la sección anterior y ahí culmina la función de petición de cada *locker*., todo esto se puede ver en la figura 3.44 y 3.45, en las que se ha tomado como ejemplo a la petición del *locker* 1.

```
//// FUNCIÓN DE PETICION DEL LOCKER 1
void locker1() {
  digitalWrite(cerraduraPin1, LOW);              // Abrir la cerradura
  while(digitalRead(magnetoPin1) ==LOW) {}       // Esperar hasta que se
separe la puerta

 // Cargar dato que identifica que está   abierta la puerta
 datostx=120;
  envio ();                                      // Enviar dato al maestro
  while(digitalRead(magnetoPin1) ==HIGH) {}      // Espera hasta que se
cierre la puerta
  digitalWrite(cerraduraPin1, HIGH);             // Cerrar la cerradura

 // Cargar dato que identifica que está cerrada la puerta
 datostx=121;
  envio();                                       // Enviar dato al maestro
  digitalWrite(activePin1,LOW);                  // Activar sensor infrarrojo
  for(a=0;a<10000;a++){                          // Empieza a sensar  el
infrarrojo
```

**Figura 3.44: Función de petición del *locker* 1**

```
// Guardar el valor en la variable respuesta (0 o 1)
 respuesta=digitalRead(infraPin1);}
  digitalWrite(activePin1,HIGH);                // Apagar sensor infrarrojo
  if(respuesta==LOW){                           // Si es 0 detecta
proyector
      digitalWrite(ledPin1,LOW);                // Encender luz verde

 // Cargar dato que identifica que hay proyector
 datostx=1;
 // Enviar dato que si hay proyector al maestro
      envio();}

  else{                                         // Si es 1 no hay proyector
      digitalWrite(ledPin1,HIGH);               // Encender luz roja

 // Cargar dato que identifica que no hay proyector
      datostx=0;
 // Enviar dato que no hay proyector al maestro
      envio();}
}
```

**Figura 3.45: Detección del proyector en el *locker* 1**

- **Interrupción por I2C**

Como se manifiesta en la figura 3.46, cuando se recibe cualquier dato por el bus I2C, se ingresa a esta sección de código que corresponde a la interrupción del protocolo I2C en donde primero se enciende un *LED* que indica que un dato ha ingresado por I2C, después se verifica si se ha recibido un *byte* y si esto se cumple se procede a cargar en la variable “datorx” con el dato recibido. Finalmente, se espera un tiempo de medio segundo y se apaga el *LED* para posteriormente regresar a la sección de código en donde se quedó el puntero.

```
////FUNCION CUANDO SE RECIBE UN DATO POR I2C

//Función que se ejecuta siempre que se reciben datos del master, siempre que en
el master  ejecute la sentencia endTransmission recibirá toda la información que
se transmitirá a través de la sentencia Wire.write
void receiveEvent() {
  digitalWrite(52,HIGH);              // Encender led indicador de que se
recibió un dato del maestro
  if (Wire.available() == 1){         // Si hay un byte disponible
   datorx = Wire.read();              // Leer y cargar la variable
    }
  delay(500);                         // Pausa de medio segundo
  digitalWrite(52,LOW);               // Apagar el led indicador de que se
recibió un dato del maestro
}
```

**Figura 3.46: Recepción de datos por I2C**

- **Funciones de envío de datos**

Estas funciones, mostradas en la figura 3.47, son llamadas cuando se requiere enviar un dato ya sea al dispositivo 0 (maestro) o al dispositivo 2 (esclavo 2); tienen el mismo fin, con la diferencia de que cambia la dirección de los dispositivos al que quieren

enviar el dato. Primero se identifica el dispositivo al que se va a enviar el dato, segundo se envía el dato que se encuentra en la variable “datostx”, después se para la trasmisión, se encera las variables y posteriormente después de una pausa de un segundo se regresa a la sección de código en donde se quedó el puntero.

```
//// FUNCIONES DE ENVIO DE DATOS A LOS DEMÁS DISPOSITIVOS

void envio(){                          // FUNCION PARA ENVIAR DATOS AL MAESTRO
    Wire.beginTransmission(0);         // Comenzar la transmisión al
dispositivo 0
    Wire.write(datostx);               // Enviar un byte
    Wire.endTransmission();            // Parar la transmisión
    datorx=0;                          // Encerar variables
    datostx=0;
    delay(1000);                       // Esperar 1 segundo
}
void envio2(){                         // FUNCION PARA ENVIAR DATOS AL
ESCLAVO 2
    Wire.beginTransmission(2);         // Comenzar la transmisión al
dispositivo 2
    Wire.write(datostx);               // Enviar un byte
    Wire.endTransmission();            // Parar la transmisión
    datostx=0;                         // Encerar variables
    datorx=0;
    delay(1000);                       // Esperar 1 segundo

}
```

**Figura 3.47: Envío de datos por I2C**

## Código Arduino esclavo 2

El código del esclavo 2 se realizó en nueve secciones, al igual que el esclavo 1 ya que tiene la misma estructura de programación y lo único en que se diferencia al esclavo uno es en que este dispositivo controla los otros diez *lockers* restantes (3,4,7,8,11,12,15,16,19 y 20). Para controlar estos *lockers*, lo que cambia son los datos que identifican a cada *locker* al momento de recibir peticiones, además de los datos de cuando exista o no proyectores, estos datos se encuentran en la tabla 3.4. Para ver el código completo se puede revisar el Anexo B-3, ahí se puede observar que este dispositivo tiene la dirección 2 en el bus de comunicación por I2C, además de que solo posee una función de envío; esto quiere decir que solo se puede comunicar con el Arduino maestro y como ya se mencionó, solo se cambian los datos de identificación de cada *locker*. En la tabla 3.7 se puede observar una mejor descripción de los pines de este dispositivo.

**Tabla 3.7: Distribución de pines del Arduino esclavo 2**

| PIN | ENTRADA/SALIDA | DESCRIPCIÓN | MÓDULO RELÉ | PINES MÓDULO RELÉ | LOCKER |
|---|---|---|---|---|---|
| **0** | - | - | - | - | - |
| **1** | - | - | - | - | - |
| **2** | Entrada | Datos sensor infrarrojo | - | - | 17 |
| **3** | Entrada | Datos sensor infrarrojo | - | - | 18 |
| **4** | Entrada | Sensor Magnético | - | - | 1 |
| **5** | Entrada | Sensor Magnético | - | - | 2 |
| **6** | Entrada | Sensor Magnético | - | - | 5 |
| **7** | Entrada | Sensor Magnético | - | - | 6 |
| **8** | Entrada | Sensor Magnético | - | - | 9 |
| **9** | Entrada | Sensor Magnético | - | - | 10 |
| **10** | Entrada | Sensor Magnético | - | - | 13 |
| **11** | Entrada | Sensor Magnético | - | - | 14 |
| **12** | Entrada | Sensor Magnético | - | - | 17 |
| **13** | Entrada | Sensor Magnético | - | - | 18 |
| **14** | Entrada | Datos sensor infrarrojo | - | - | 14 |
| **15** | Entrada | Datos sensor infrarrojo | - | - | 13 |
| **16** | Entrada | Datos sensor infrarrojo | - | - | 10 |
| **17** | Entrada | Datos sensor Infrarrojo | - | - | 9 |
| **18** | Entrada | Datos sensor infrarrojo | - | - | 6 |
| **19** | Entrada | Datos sensor infrarrojo | - | - | 5 |
| **20** | - | - | - | - | - |
| **21** | - | - | - | - | - |
| **22** | Salida | Luces | 3 | 16 | 12 |
| **23** | Salida | Luces | 3 | 15 | 7 |
| **24** | Salida | Luces | 3 | 14 | 8 |
| **25** | Salida | Luces | 3 | 13 | 3 |
| **26** | Salida | Luces | 3 | 12 | 4 |
| **27** | Salida | Cerradura | 3 | 11 | 12 |

| PIN | ENTRADA/SALIDA | DESCRIPCIÓN | MÓDULO RELÉ | PINES MÓDULO RELÉ | *LOCKER* |
|---|---|---|---|---|---|
| **29** | Salida | Cerradura | 3 | 9 | 8 |
| **30** | Salida | Cerradura | 3 | 8 | 3 |
| **31** | Salida | Cerradura | 3 | 8 | 4 |
| **33** | Salida | Polarización sensor Infrarrojo | 3 | 5 | 4 |
| **34** | Salida | Polarización sensor Infrarrojo | 3 | 5 | 3 |
| **35** | Salida | Polarización sensor Infrarrojo | 3 | 3 | 8 |
| **36** | Salida | Polarización sensor Infrarrojo | 3 | 2 | 7 |
| **37** | Salida | Polarización sensor Infrarrojo | 3 | 1 | 12 |
| **38** | Salida | Luces | 4 | 16 | 19 |
| **39** | Salida | Luces | 4 | 15 | 20 |
| **40** | Salida | Luces | 4 | 14 | 15 |
| **41** | Salida | Luces | 4 | 13 | 16 |
| **42** | Salida | Luces | 4 | 12 | 11 |
| **43** | Salida | Cerradura | 4 | 11 | 19 |
| **44** | Salida | Cerradura | | 10 | 20 |
| **45** | Salida | Cerradura | 4 | 9 | 15 |
| **46** | Salida | Cerradura | 4 | 8 | 16 |
| **47** | Salida | Cerradura | 4 | 7 | 11 |
| **48** | Salida | Polarización sensor Infrarrojo | 4 | 6 | 11 |
| **49** | Salida | Polarización sensor Infrarrojo | 4 | 5 | 20 |
| **50** | Salida | Polarización sensor Infrarrojo | 4 | 4 | 16 |
| **51** | Entrada | Datos sensor Infrarrojo | - | - | 3 |
| **52** | Salida | Led indicador | - | - | Placa |
| **53** | Entrada | Datos sensor Infrarrojo | - | - | 4 |

## 3.4. Realización de pruebas del funcionamiento del programa

Para la fase de pruebas, se conectó todos los elementos adquiridos con los Arduinos, además se realizó una simulación de los *lockers* con cajas de cartón y circuitos a pequeña escala.

En la figura 3.48 se aprecia a los Arduinos interconectados entre sí y con los demás elementos adquiridos; además se simularon las peticiones de préstamo de proyectores con pulsadores y LEDs.

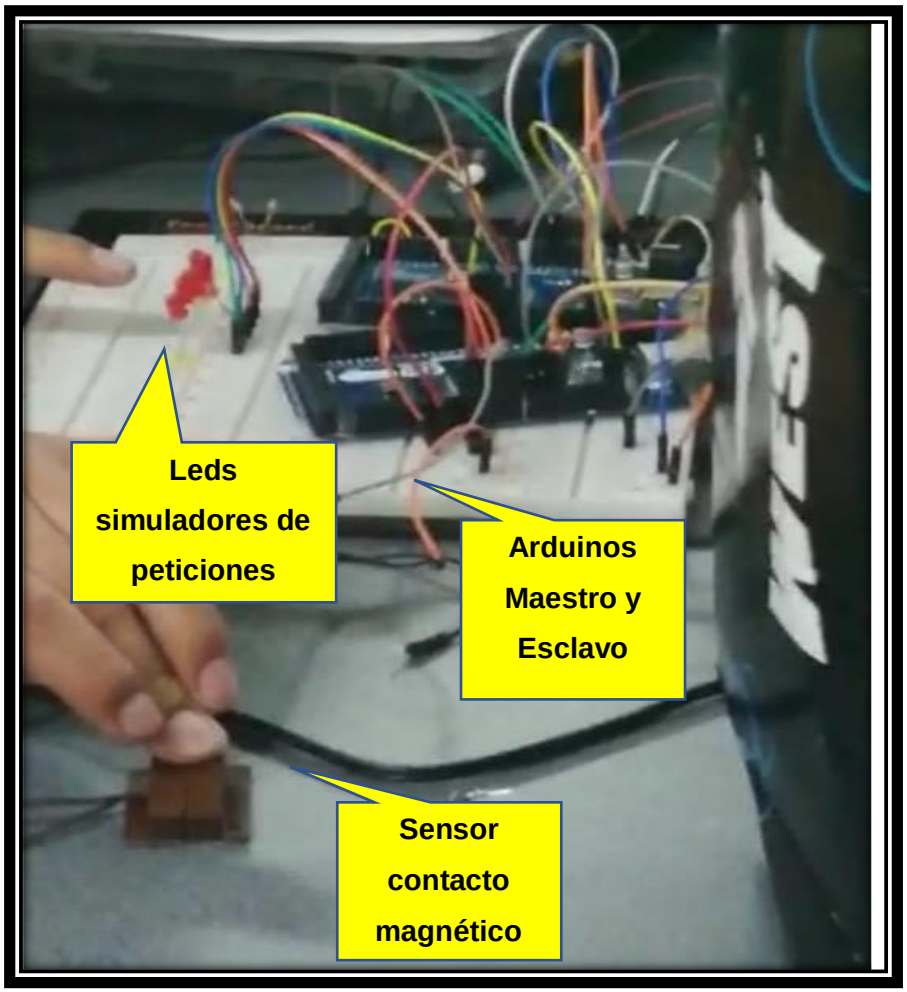



**Figura 3.48: Simulación de peticiones con los Arduinos**

En la Figura 3.49 y 3.50, se observa la conmutación entre las luces gracias a los relés, al momento de cargar el programa y que los sensores envíen la señal.

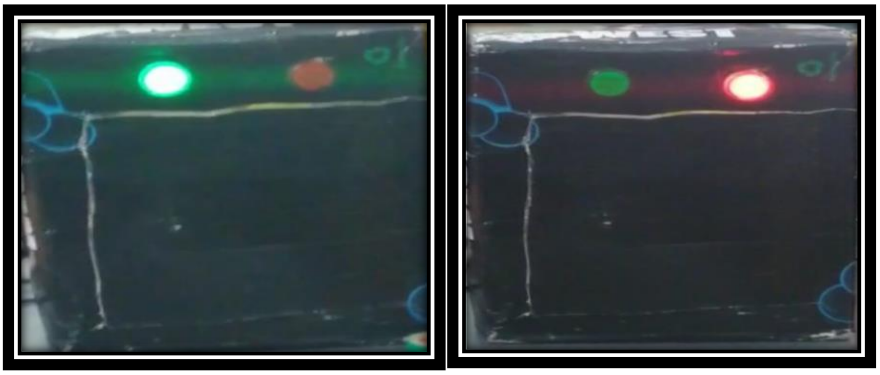


**Figura 3.49: Conmutación entre las luces piloto**

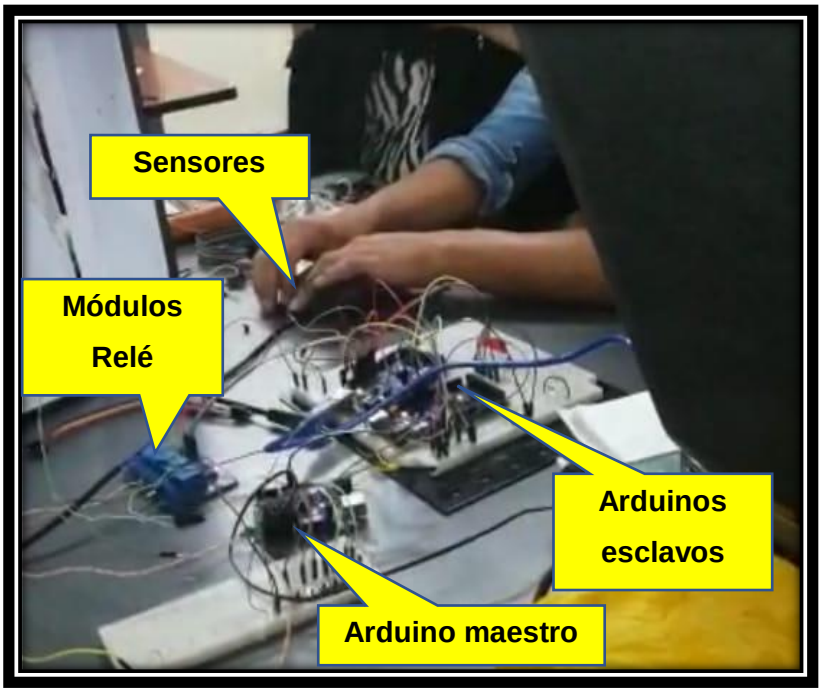



**Figura 3.50: Simulación del código de los sensores**

## 3.5. Construcción de la placa general

Se partió del diseño para la realización de la placa y sus respectivas pruebas de funcionamiento ya que en ella se encuentran todos los Arduinos conectados. Cabe indicar que no se profundiza en dicho diseño, puesto que otros estudiantes lo realizarán en su documento. Para este proceso se utilizó el *Software Proteus* con sus extensiones ISIS para la conexión de elementos y ARES para el diseño de la PCB. Posteriormente, se adquirieron todos los elementos necesarios para la elaboración de la placa, que, en este caso, fueron en tecnología SMD *(Surface Mounted Device)* debido a que fue hecha en una baquelita a doble lado.

## Diseño de la Placa general de Arduinos

El diseño de todas las placas se tratará en el documento de otros estudiantes, sin embargo, en la figura 3.51 y 3.52 se muestra la placa general de los Arduinos ya que esta placa fue esencial para realizar las pruebas de funcionamiento de los códigos descritos y posteriormente la integración de todo el sistema.

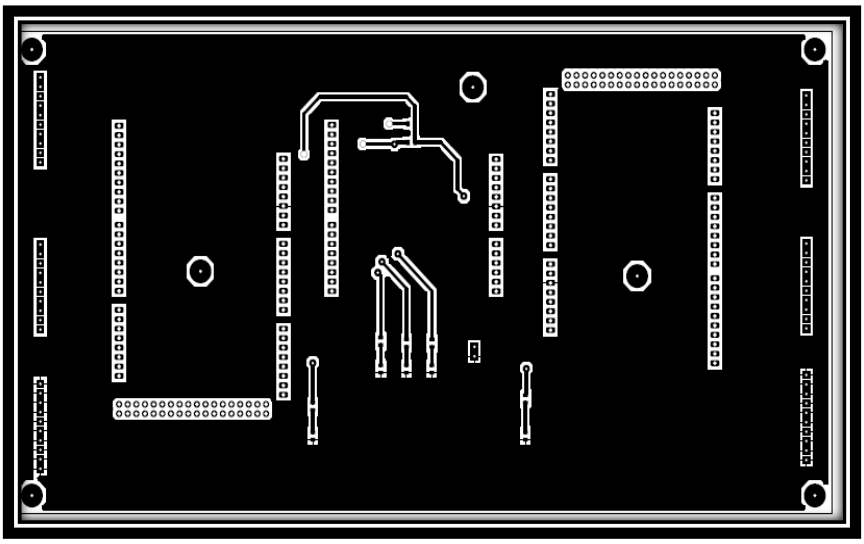


**Figura 3.51: Placa general de los Arduinos parte frontal**

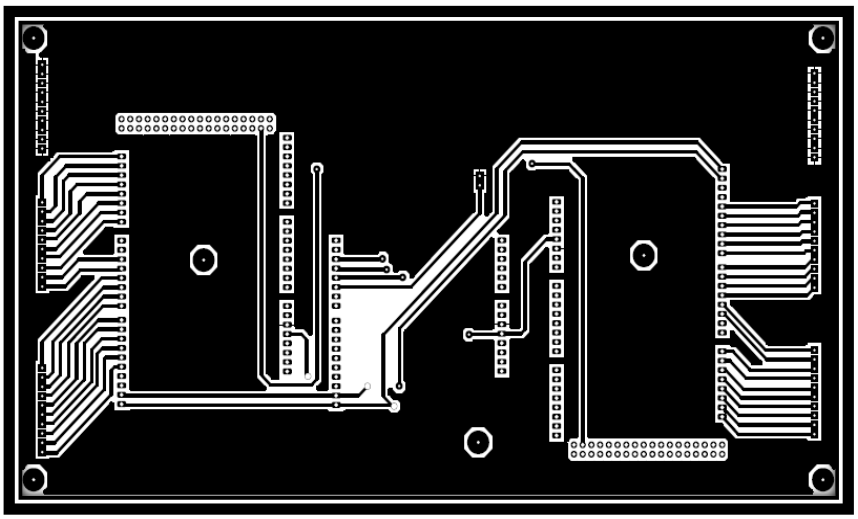


**Figura 3.52: Placa general de los Arduinos parte trasera**

## Elaboración de la Placa

En la figura 3.53 y 3.54 se presenta el proceso de limpieza y planchado de la baquelita, respectivamente.

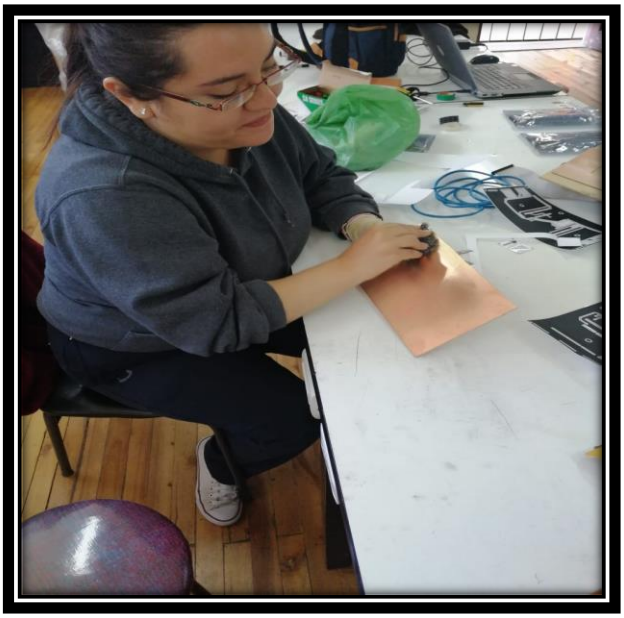


**Figura 3.53: Limpieza de baquelita**

**Figura 3.54: Planchado del diseño en la placa.**

### Ensamblaje de elementos en la placa

Al tener listo el diseño en la placa, se realizó el ensamblaje de los elementos y de los Arduinos en la placa, y además se verificó continuidad entre las pistas, se procedió a la soldadura, como se muestra en la figura 3.55 y 3.56.

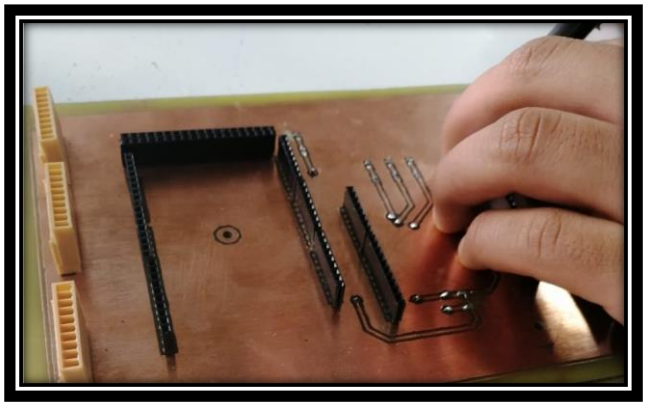


**Figura 3.55: Colocación de los elementos en la placa**

**Figura 3.56: Soldadura en tecnología SMD**

### Pruebas de funcionamiento de la placa

Se verificó la comunicación I2C entre Arduinos y la comunicación serial entre el Arduino maestro y el servidor, como se observa en la figura 3.57.

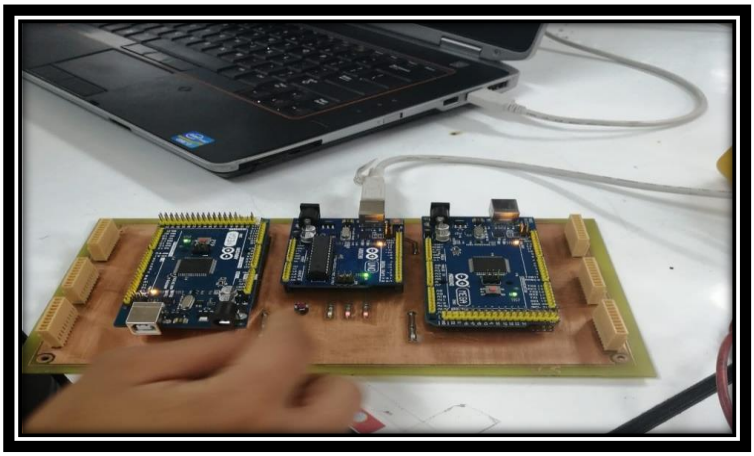


**Figura 3.57: Protocolo I2C y Comunicación Serial**

## 3.6. Verificación del correcto funcionamiento de la programación

La fase de pruebas fue primordial en cada paso; así mismo, cuando se finalizó con todo lo anterior, se hizo una verificación global con todos los *lockers* conectados a los Arduinos, los cuales ya estaban en su respectiva placa. Además, como se puede apreciar en la figura 3.58 a la 3.60, fue necesario realizar el cableado correspondiente de los dispositivos electrónicos, los cuales ya estaban dispuestos en su respectivo lugar dentro de la estructura.

**Figura 3.58: Cableado y ubicación de elementos**

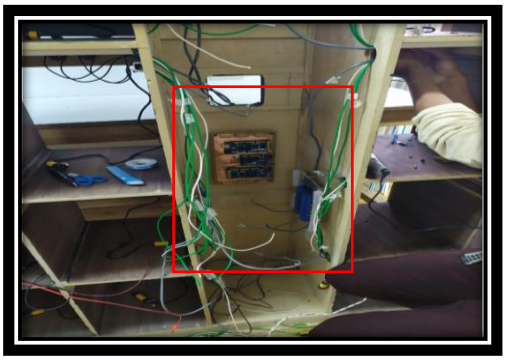


**Figura 3.59: Ubicación de la placa general de los Arduinos**

**Figura 3.60: Cableado de los módulos relé**

Una vez que se realizó todo el cableado interno de los elementos con su respectiva alimentación, se procedió a encenderlo para comprobar la funcionalidad del código. En la figura 3.61, se observa las luces de la estructura ya encendidas y muestran el estado de que no hay disponibilidad de proyectores en los *lockers* gracias al sensor infrarrojo.

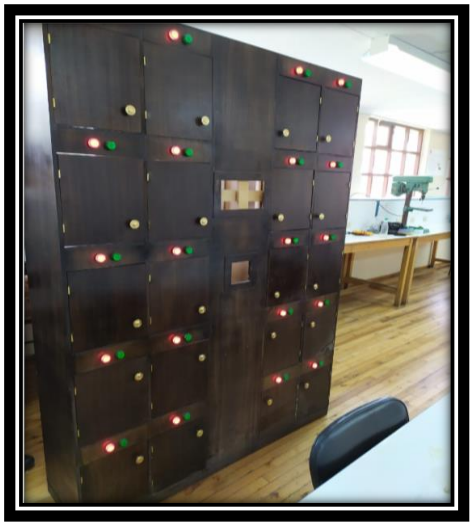


**Figura 3.61: Funcionalidad del código en luces y sensores**

En la figura 3.62 se aprecia la etapa de control en la que el sistema ya se encontraba integrado en su totalidad y se realiza su presentación gracias a un servidor que tiene albergada la interfaz correspondiente, desarrollada por una estudiante de la carrera de Análisis de Sistemas Informáticos.

**Figura 3.62: Etapa de control del sistema integrado en su totalidad**

# 4. CONCLUSIONES Y RECOMENDACIONES

## 4.1. Conclusiones

- Se constató la necesidad de un sistema de registro de clases de los profesores y a su vez para el proceso de gestión de préstamo de proyectores, debido a que esto ocasionaba una considerable pérdida de tiempo y de papel para realizar estos registros.

- Se realizó la investigación sobre los dispositivos de control necesarios para el desarrollo del código y la posterior implementación de todo el sistema; es así que, se dedujo los elementos principales tales como: sensores infrarrojos, cerraduras eléctricas, sensores de contacto magnético, módulos relé y de manera indispensable dos Arduinos MEGA y un Arduino UNO, ya que son esenciales para llevar a cabo la programación y funcionamiento de este proyecto.

- Se desarrolló el código correspondiente para cada elemento y además para la comunicación entre los dispositivos de control (Arduinos), donde se utilizó el protocolo I2C; y entre el Arduino maestro y el servidor, se manejó comunicación serial. Además, se optó por realizar un código para el Arduino maestro y un código para cada Arduino esclavo, así se facilitó el uso de funciones a lo largo de la programación y fue necesario para la distribución de pines.

- Se hicieron pruebas de funcionamiento del programa del Arduino maestro y de los Arduinos esclavos con los elementos elegidos para el sistema, donde al inicio se presentaron problemas ya que cuando se procedió a alimentar el sistema, las luces no encendieron en su totalidad, esto fue debido a que las conexiones estaban flojas.

- En los Arduinos Mega no se pudieron utilizar los pines 0, 1, 20 y 21, debido a que estos pines están relacionados con los dos tipos de comunicación que se utilizaron para el desarrollo del proyecto y al momento de ser declarados estos pines como entradas o salidas se ven afectados y presentan interferencias. Los pines 0 y 1 se encuentran mapeados al monitor serie como al USB y los pines 20 y 21 están mapeados con el Protocolo de Comunicación I2C utilizados para

el SDA y SCL, quitando flexibilidad al proyecto ya que en un instante se contaba con el uso de estos pines para el desarrollo del código y del Hardware.

- Se optó por una comunicación serial, en lugar de *Ethernet*, debido a que ofrece una mejor sincronización y su cableado es mucho más simple; sobre todo, se aprovechó la ventaja de la velocidad de transmisión, es decir el número de bits transmitidos en un segundo, de esta manera el envío de datos se realiza bit a bit y por una misma línea de comunicación.

- Debido a la gran cantidad de elementos electrónicos que se requería conectar, se optó por utilizar una segmentación en dos partes del control de los 20 *lockers,* diez para un Arduino Mega y los otros diez restantes para el segundo Arduino Mega. Para tener conexiones adecuadas y ordenadas, se optó por utilizar el protocolo de comunicación I2C, debido a que este protocolo soporta múltiples maestros y esclavos. Este protocolo es muy beneficioso ya que posee un *hardware* menos complicado y además se utilizó datos no mayores a 8 bits y este protocolo maneja ese tamaño de paquete.

- Al momento de compilar los programas el *sketch* en el Arduino MEGA (esclavos) utiliza el 24% de la memoria FLASH y el 20% de su memoria SRAM. Por otro lado, el *sketch* en el Arduino UNO (maestro) ocupa el 25% de la memoria FLASH y el 34% de su memoria SRAM. Concluyendo que, a pesar de la extensión de los códigos de programación, los dispositivos trabajan sin efectos inesperados cuando no se superan el 70% al 75% de la SRAM debido a que esta memoria volátil es un recurso escaso y si se lo sobrecarga, el usuario tendrá problemas con la velocidad de uso.

- Gracias al entorno de programación simple y directo de Arduino, las primeras pruebas de sensores se las realizaban con pulsadores, logrando economizar recursos en el proyecto. Además de la alta flexibilidad de Arduino con otras plataformas informaticas, en el proyecto se logró una comunicación eficaz entre el Arduino Maestro y el Servidor por medio del Serial.

- Pese a que el Arduino utiliza dispositivos AVR y que a estos también se los puede programar en un lenguaje C, hubiera sido un proceso más largo y complejo desarrollar este proyecto en estos microcontroladores ya que se requiere un amplio conocimiento del funcionamiento de sus registros.

- Para el desarrollo de una librería I2C y comunicación serial, gracias al entorno del lenguaje de programación que presenta Arduino, este proceso se pudo simplificar solo añadiendo las librerías del protocolo de comunicación que ya Arduino posee.

- Debido a la gran cantidad de elementos que se iban a utilizar, los primeros diagramas de flujo y códigos de programación se los realizaban solo para diez lockers, cinco en cada esclavo para ser exactos, permitiendo tener una idea más clara del cómo iba a ser el desarrollo del programa completo y cómo se íba a implementar los códigos o números únicos de identificación de cada locker en cada caso ya sea de peticiones, devoluciones de lockers o los datos que informaban al servidor si un locker tenía o no un proyector.

## 4.2. Recomendaciones

- En caso de alguna falla en el sistema, se recomienda utilizar el manual de usuario presentado en el Anexo C, en este documento se presentan las consideraciones más importantes para una mejor administración de todo el sistema, así como de sus elementos y placas.

- Verificar el estado del cable para la comunicación serial entre la placa de los Arduinos y los módulos relé, puesto que con el tiempo puede existir daños en el mismo.

- Es importante que el personal administrativo de la ESFOT presione siempre el botón de reinicio de la placa de los Arduinos en caso de que el sistema se haya apagado.

- Es necesario que el personal de la ESFOT verifique el buen funcionamiento del sistema y evitar que los estudiantes hagan un mal uso o manipulen los dispositivos, ocasionando daños en los sensores, *tablet* o biométrico.

# 5. REFERENCIAS BIBLIOGRÁFICAS


[1] C. Edu, «Csulb Edu,» 2015. [En línea]. Available: http://web.csulb.edu/~hill/ee400d/Technical%20Training%20Series/02%20Intro%20to%20Arduino.pdf. [Último acceso: Enero 2020].

[2] Arduino, «What is Arduino?,» [En línea]. Available: https://www.arduino.cc/en/Guide/Introduction. [Último acceso: Enero 2020].

[3] Arduino, «Arduino Software IDE,» 7 September 2015. [En línea]. Available: https://www.arduino.cc/en/Guide/Environment. [Último acceso: 3 Enero 2020].

[4] Arduino, «Arduino Mega 2560,» [En línea]. Available: https://store.arduino.cc/usa/arduino-mega-2560-rev3. [Último acceso: 5 Enero 2019].

[5] J. DIAZ, «PLACA ARDUINO UNO,» 21 Enero 2016. [En línea]. Available: http://www.iescamp.es/miarduino/2016/01/21/placa-arduino-uno/. [Último acceso: 12 Febrero 2020].

[6] N. Ledesma, «Modelo de Aprendizaje para Arduino Uno Básico,» *Revista de Cómputo Aplicado,* vol. 3, nº 10, pp. 15-22, 2019.

[7] «Aprendiendo Arduino,» [En línea]. Available: https://aprendiendoarduino.wordpress.com/2017/01/23/programacion-arduino-5/. [Último acceso: 4 Abril 2020].

[8] PROMETEC, «Introducción a la programación del Arduino,» [En línea]. Available: https://www.prometec.net/intro-programacion/. [Último acceso: 4 Abril 2020].

[9] «Manual de programación de Arduino,» [En línea]. Available: http://dfists.ua.es/~jpomares/arduino/page_10.htm. [Último acceso: 4 Abril 2020].

[10] E. J. Carletti, «Comunicación - Bus I2C,» [En línea]. Available: http://www.bolanosdj.com.ar/MOVIL/ARDUINO2/ComunicacionBusI2C.pdf. [Último acceso: 4 Abril 2020].

[11] PROMETEC, «Bus I2C,» [En línea]. Available: https://www.prometec.net/bus-i2c/. [Último acceso: 4 Abril 2020].

[12] «Electrónia y Robótica,» [En línea]. Available: https://www.e-ika.com/modulo-de-reles-16-canales-para-arduino. [Último acceso: 2020 Mayo 23].

[13] AMAZON, «AMAZON BUSINESS,» [En línea]. Available: https://www.amazon.es/DyNamic-Cerradura-El%C3%A9ctrica-Montaje-Solenoide/dp/B07QV8H3TW. [Último acceso: 20 ENERO 2020].

[14] Lab-Volt, «Sensores,» Febrero 2001. [En línea]. Available: http://biblio3.url.edu.gt/Publi/Libros/2013/ManualesIng/FluidosySensores-O.pdf. [Último acceso: 23 Mayo 2020].

[15] «AMAZON,» [En línea]. Available: https://www.amazon.es/Infrarrojo-Obst%C3%A1culo-Akozon-Obst%C3%A1culos-Infrarrojos/dp/B07DKGGHCD. [Último acceso: 25 ENERO 2020].

[16] «Geek Factory,» [En línea]. Available: https://www.geekfactory.mx/tienda/sensores/e18-d80nk-sensor-de-proximidad-infrarrojo/. [Último acceso: 6 Abril 2020].

[17] TECNOSEGURO, «¿Qué es un detector magnético de apertura?,» [En línea]. Available: https://www.tecnoseguro.com/faqs/alarma/que-es-un-detector-magnetico-de-apertura. [Último acceso: 6 Abril 2020].

[18] MERCADO LIBRE, «Seguridad en el Hogar,» [En línea]. Available: https://articulo.mercadolibre.com.ec/MEC-424308815-sensor-contacto-magnetico-para-alarma-en-puertas-o-arduino-_JM?quantity=1. [Último acceso: 25 Enero 2020].

[19] C. A. y. F. Beltrán, PROGRAMACION MODULAR, La Paz: Universal Salesiana de Bolivia, 2011.

[20] ALIEXPRESS, «Electrical Equipment,» [En línea]. Available: https://es.aliexpress.com/item/32374697528.html. [Último acceso: 20 Enero 2020].

[21] A. Alhambra, «Tecnología, Robótica y algo más,» [En línea]. Available: https://www.scoop.it/topic/tecnologia-robotica-y-algo-mas/?&tag=Arduino+domotica. [Último acceso: 6 Abril 2020].

# ANEXOS

ANEXO A-1: Diagrama de flujo Arduino maestro

ANEXO A-2: Diagrama de flujo Arduino esclavo 1

ANEXO A-3: Diagrama de flujo Arduino esclavo 2

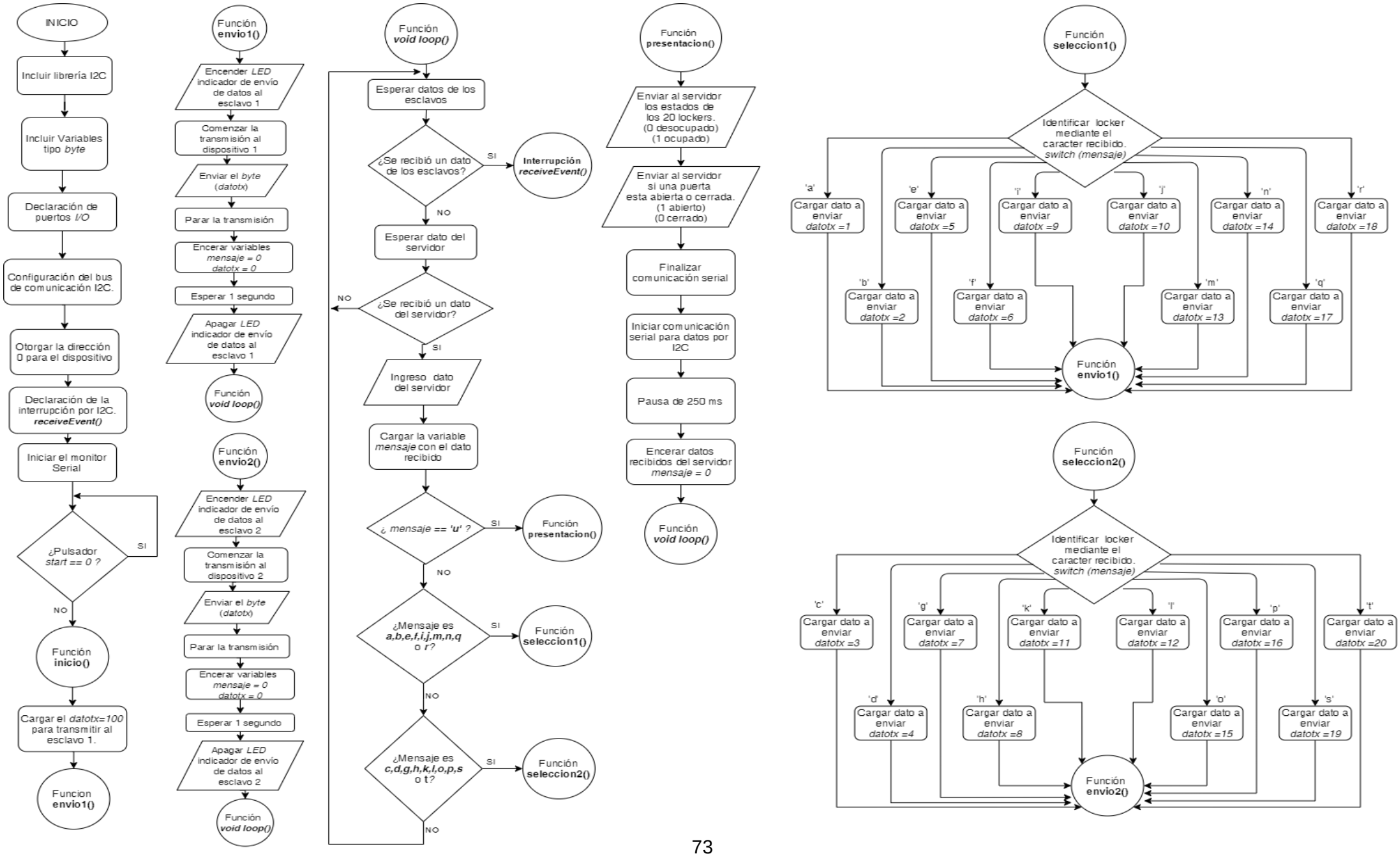
ANEXO B-1: Código completo Arduino maestro

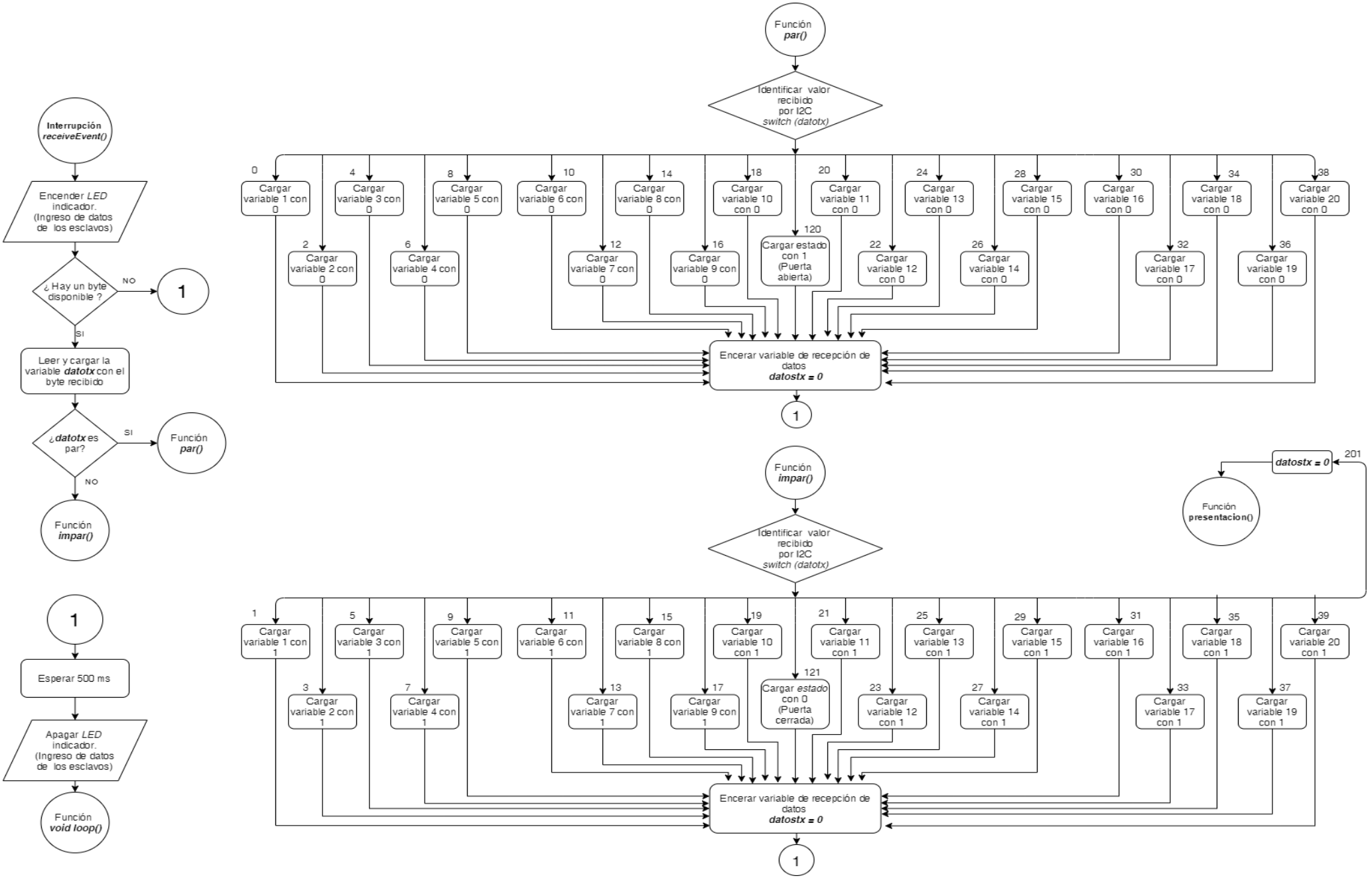
ANEXO B-2: Código completo Arduino esclavo 1

ANEXO B-3: Código complete Arduino esclavo 2

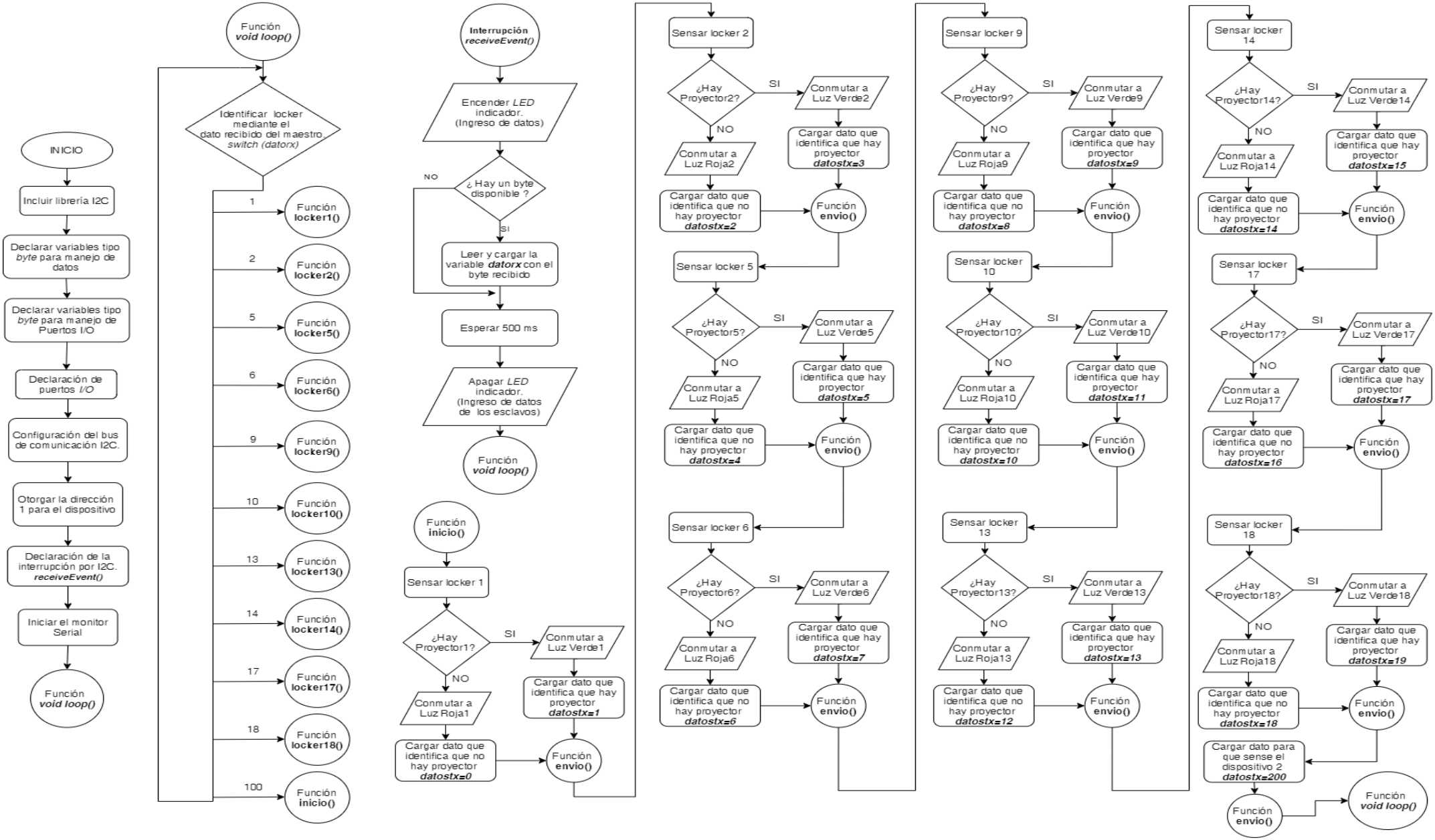
ANEXO C: Manual de usuario

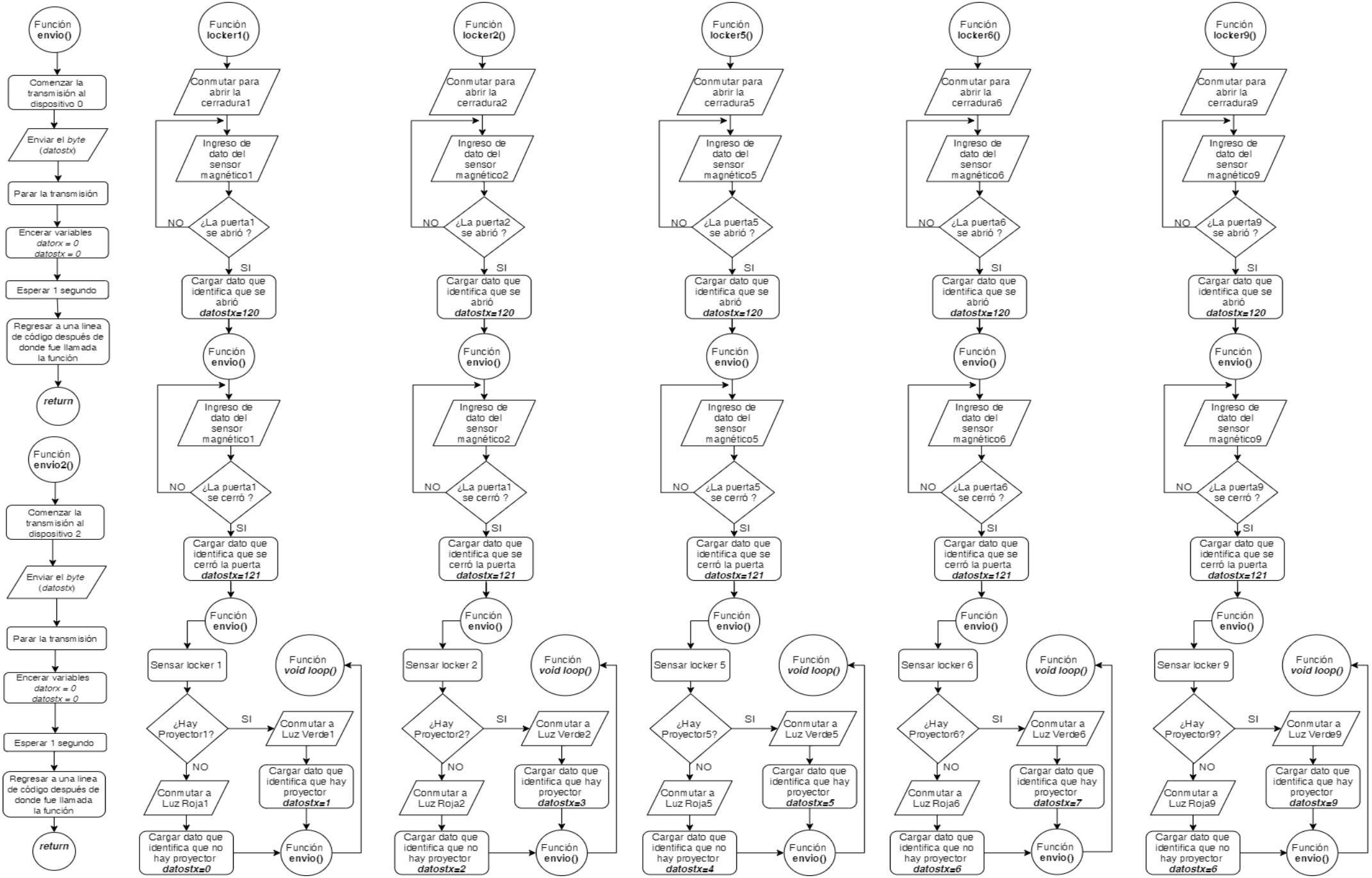
## ANEXO A-1: Diagrama de flujo Arduino maestro

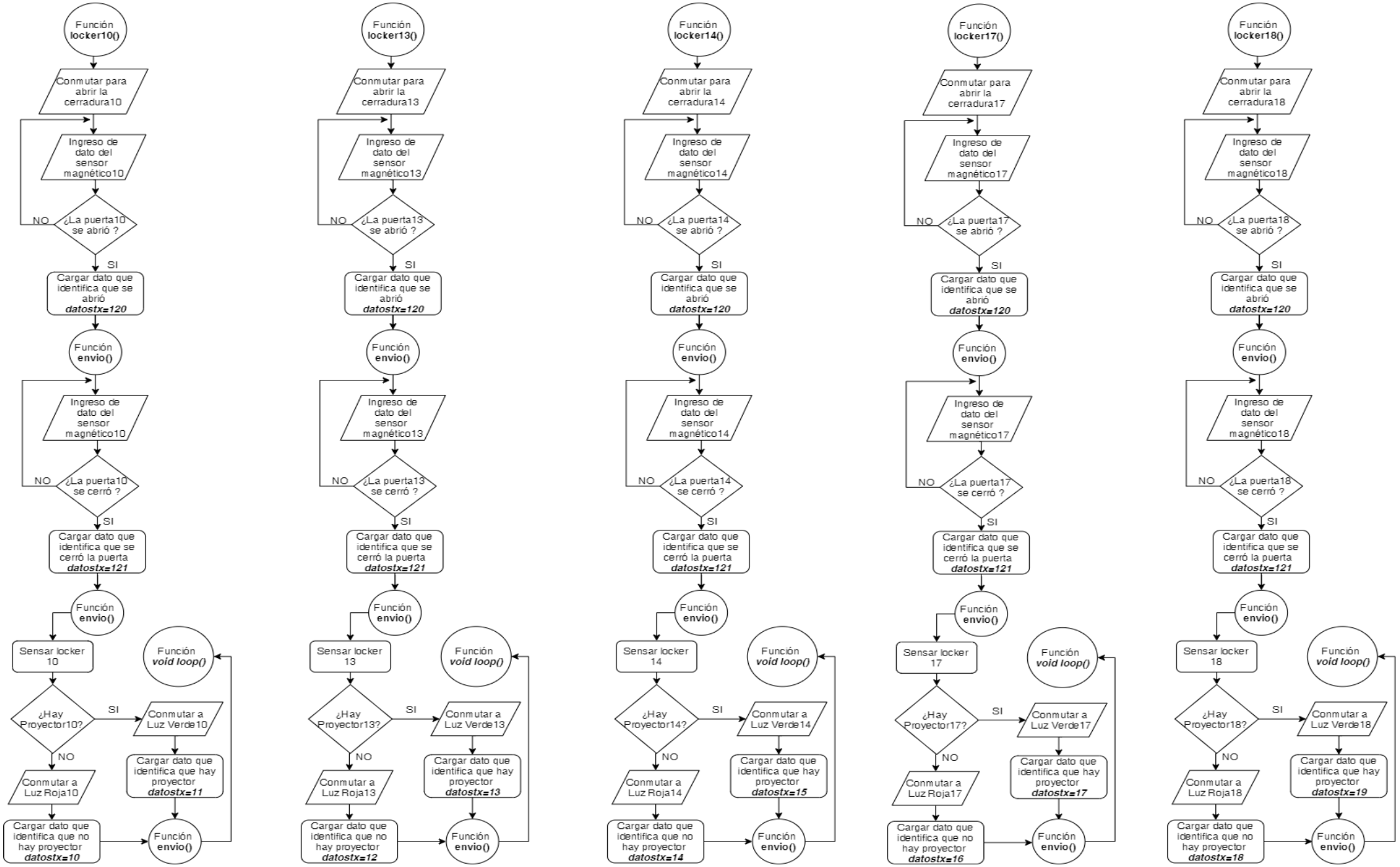



Interrupción
receiveEvent()
Encender LED indicador. (Ingreso de datos de los esclavos)
¿ Hay un byte disponible ?
NO
1
SI
Leer y cargar la variable datotx con el byte recibido
¿datotx es par?
SI
Función par()
NO
Función impar()
1
Esperar 500 ms
Apagar LED indicador. (Ingreso de datos de los esclavos)
Función void loop()
Función par()
Identificar valor recibido por I2C switch (datotx)
0 Cargar variable 1 con 0
2 Cargar variable 2 con 0
4 Cargar variable 3 con 0
6 Cargar variable 4 con 0
8 Cargar variable 5 con 0
10 Cargar variable 6 con 0
12 Cargar variable 7 con 0
14 Cargar variable 8 con 0
16 Cargar variable 9 con 0
18 Cargar variable 10 con 0
20 Cargar variable 11 con 0
120 Cargar estado con 1 (Puerta abierta)
22 Cargar variable 12 con 0
24 Cargar variable 13 con 0
26 Cargar variable 14 con 0
28 Cargar variable 15 con 0
30 Cargar variable 16 con 0
32 Cargar variable 17 con 0
34 Cargar variable 18 con 0
36 Cargar variable 19 con 0
38 Cargar variable 20 con 0
Encerar variable de recepción de datos datostx = 0
1
Función impar()
Identificar valor recibido por I2C switch (datotx)
1 Cargar variable 1 con 1
3 Cargar variable 2 con 1
5 Cargar variable 3 con 1
7 Cargar variable 4 con 1
9 Cargar variable 5 con 1
11 Cargar variable 6 con 1
13 Cargar variable 7 con 1
15 Cargar variable 8 con 1
17 Cargar variable 9 con 1
19 Cargar variable 10 con 1
21 Cargar variable 11 con 1
121 Cargar estado con 0 (Puerta cerrada)
23 Cargar variable 12 con 1
25 Cargar variable 13 con 1
27 Cargar variable 14 con 1
29 Cargar variable 15 con 1
31 Cargar variable 16 con 1
33 Cargar variable 17 con 1
35 Cargar variable 18 con 1
37 Cargar variable 19 con 1
39 Cargar variable 20 con 1
Encerar variable de recepción de datos datostx = 0
1
Función presentacion()
datostx = 0
201

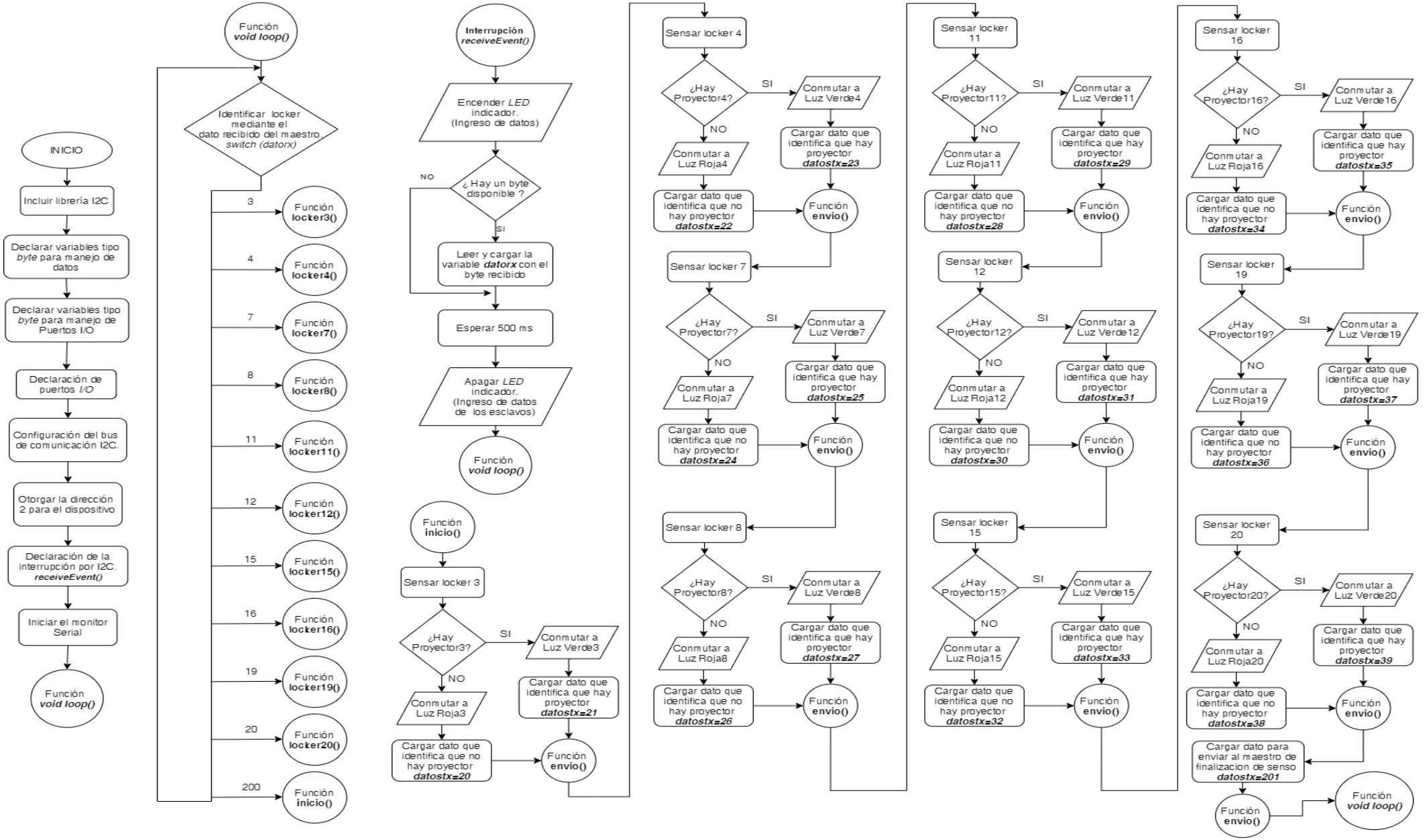
## ANEXO A-2: Diagrama de flujo Arduino esclavo 1

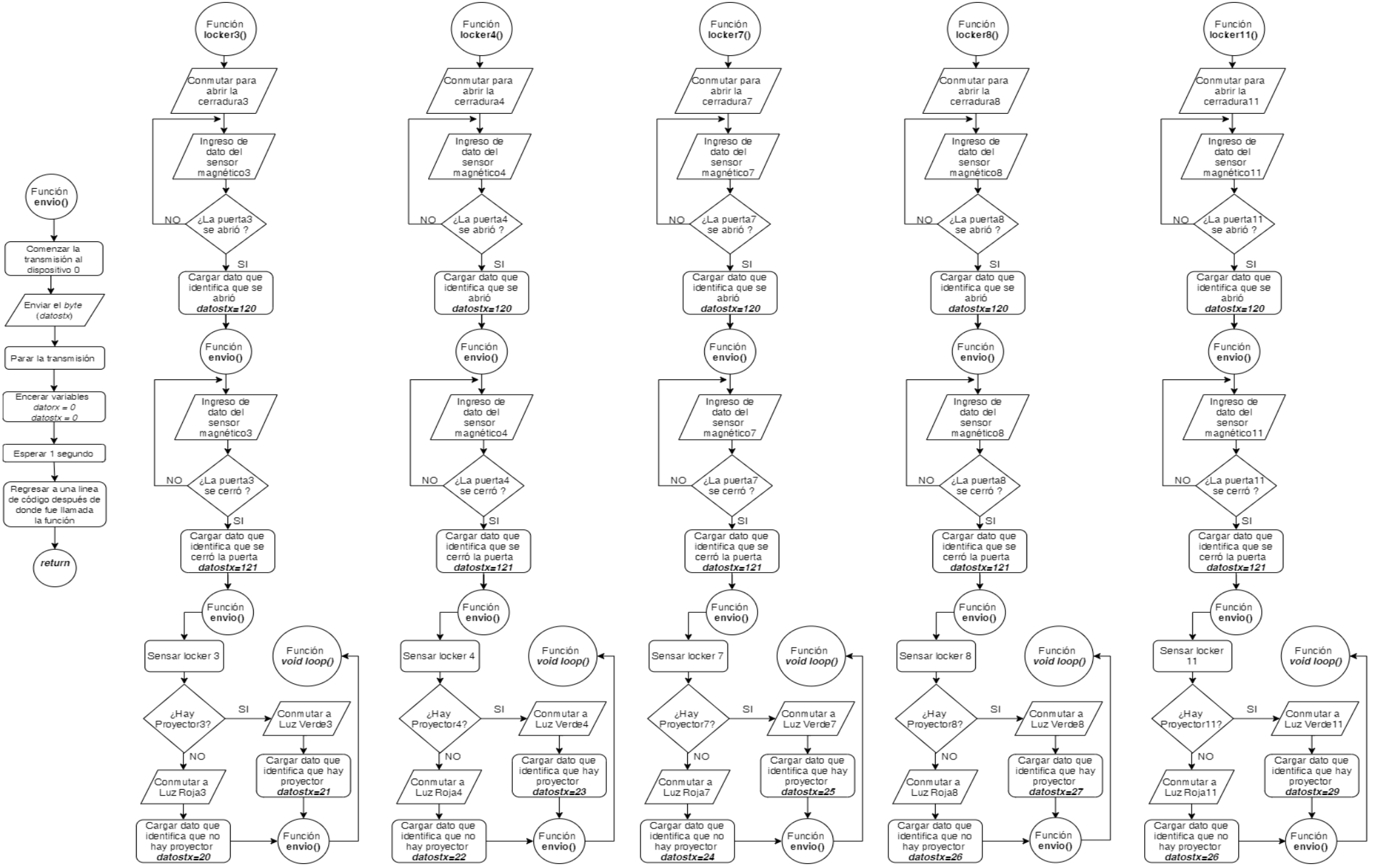


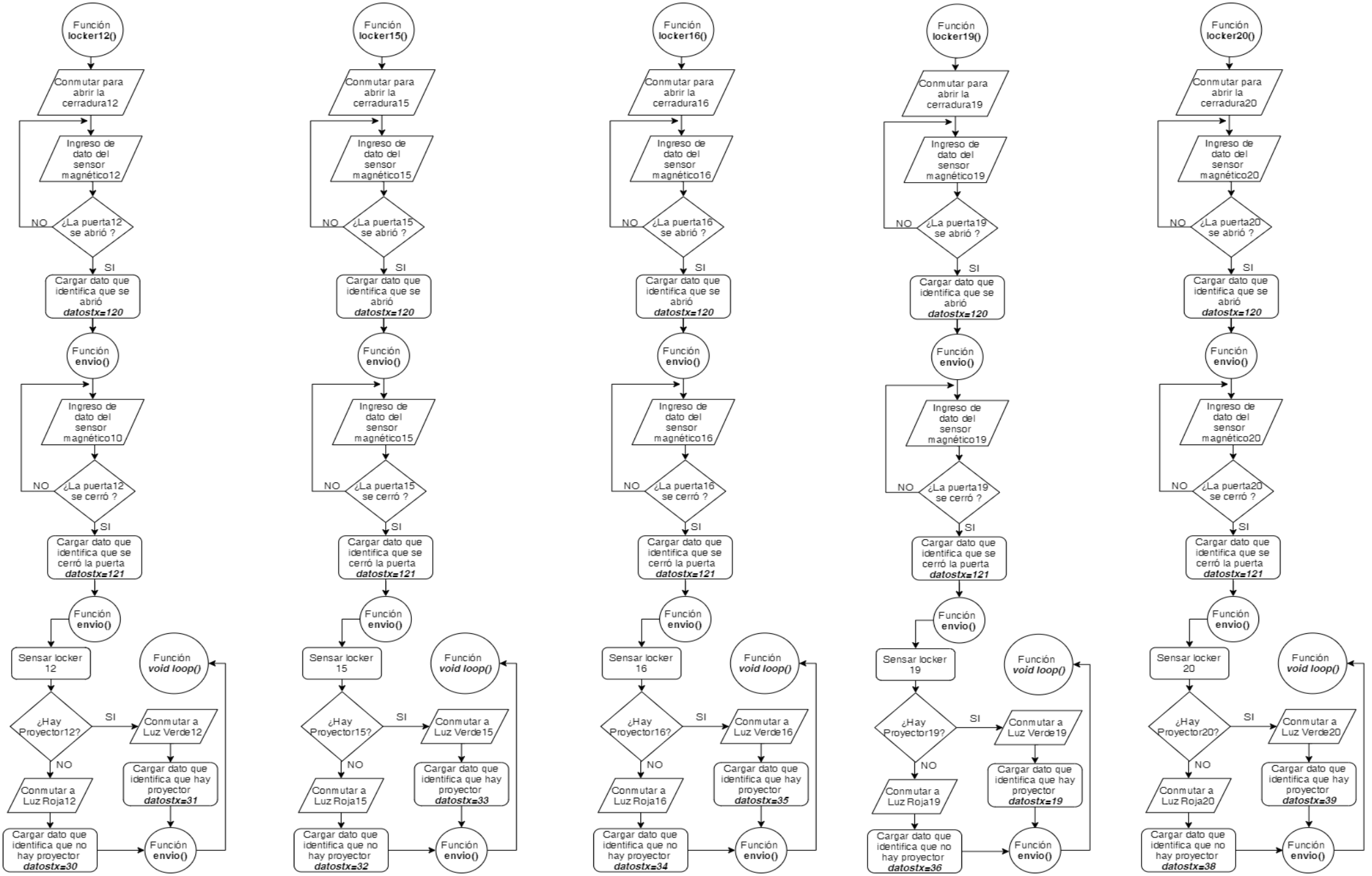

Función envio()
Comenzar la transmisión al dispositivo 0
Enviar el byte (datostx)
Parar la transmisión
Encerar variables datorx = 0 datostx = 0
Esperar 1 segundo
Regresar a una linea de código después de donde fue llamada la función
return
Función envio2()
Comenzar la transmisión al dispositivo 2
Enviar el byte (datostx)
Parar la transmisión
Encerar variables datorx = 0 datostx = 0
Esperar 1 segundo
Regresar a una linea de código después de donde fue llamada la función
return
Función locker1()
Conmutar para abrir la cerradura1
Ingreso de dato del sensor magnético1
NO
¿La puerta1 se abrió ?
SI
Cargar dato que identifica que se abrió datostx=120
Función envio()
Ingreso de dato del sensor magnético1
NO
¿La puerta1 se cerró ?
SI
Cargar dato que identifica que se cerró la puerta datostx=121
Función envio()
Sensar locker 1
¿Hay Proyector1?
SI
Conmutar a Luz Verde1
Cargar dato que identifica que hay proyector datostx=1
NO
Conmutar a Luz Roja1
Cargar dato que identifica que no hay proyector datostx=0
Función envio()
Función void loop()
Función locker2()
Conmutar para abrir la cerradura2
Ingreso de dato del sensor magnético2
NO
¿La puerta2 se abrió ?
SI
Cargar dato que identifica que se abrió datostx=120
Función envio()
Ingreso de dato del sensor magnético2
NO
¿La puerta1 se cerró ?
SI
Cargar dato que identifica que se cerró la puerta datostx=121
Función envio()
Sensar locker 2
¿Hay Proyector2?
SI
Conmutar a Luz Verde2
Cargar dato que identifica que hay proyector datostx=3
NO
Conmutar a Luz Roja2
Cargar dato que identifica que no hay proyector datostx=2
Función envio()
Función void loop()
Función locker5()
Conmutar para abrir la cerradura5
Ingreso de dato del sensor magnético5
NO
¿La puerta5 se abrió ?
SI
Cargar dato que identifica que se abrió datostx=120
Función envio()
Ingreso de dato del sensor magnético5
NO
¿La puerta5 se cerró ?
SI
Cargar dato que identifica que se cerró la puerta datostx=121
Función envio()
Sensar locker 5
¿Hay Proyector5?
SI
Conmutar a Luz Verde5
Cargar dato que identifica que hay proyector datostx=5
NO
Conmutar a Luz Roja5
Cargar dato que identifica que no hay proyector datostx=4
Función envio()
Función void loop()
Función locker6()
Conmutar para abrir la cerradura6
Ingreso de dato del sensor magnético6
NO
¿La puerta6 se abrió ?
SI
Cargar dato que identifica que se abrió datostx=120
Función envio()
Ingreso de dato del sensor magnético6
NO
¿La puerta6 se cerró ?
SI
Cargar dato que identifica que se cerró la puerta datostx=121
Función envio()
Sensar locker 6
¿Hay Proyector6?
SI
Conmutar a Luz Verde6
Cargar dato que identifica que hay proyector datostx=7
NO
Conmutar a Luz Roja6
Cargar dato que identifica que no hay proyector datostx=6
Función envio()
Función void loop()
Función locker9()
Conmutar para abrir la cerradura9
Ingreso de dato del sensor magnético9
NO
¿La puerta9 se abrió ?
SI
Cargar dato que identifica que se abrió datostx=120
Función envio()
Ingreso de dato del sensor magnético9
NO
¿La puerta9 se cerró ?
SI
Cargar dato que identifica que se cerró la puerta datostx=121
Función envio()
Sensar locker 9
¿Hay Proyector9?
SI
Conmutar a Luz Verde9
Cargar dato que identifica que hay proyector datostx=9
NO
Conmutar a Luz Roja9
Cargar dato que identifica que no hay proyector datostx=6
Función envio()
Función void loop()


Función locker10()
Conmutar para abrir la cerradura10
Ingreso de dato del sensor magnético10
NO
¿La puerta10 se abrió ?
SI
Cargar dato que identifica que se abrió datostx=120
Función envio()
Ingreso de dato del sensor magnético10
NO
¿La puerta10 se cerró ?
SI
Cargar dato que identifica que se cerró la puerta datostx=121
Función envio()
Sensar locker 10
¿Hay Proyector10?
SI
Conmutar a Luz Verde10
Cargar dato que identifica que hay proyector datostx=11
NO
Conmutar a Luz Roja10
Cargar dato que identifica que no hay proyector datostx=10
Función envio()
Función void loop()
Función locker13()
Conmutar para abrir la cerradura13
Ingreso de dato del sensor magnético13
NO
¿La puerta13 se abrió ?
SI
Cargar dato que identifica que se abrió datostx=120
Función envio()
Ingreso de dato del sensor magnético13
NO
¿La puerta13 se cerró ?
SI
Cargar dato que identifica que se cerró la puerta datostx=121
Función envio()
Sensar locker 13
¿Hay Proyector13?
SI
Conmutar a Luz Verde13
Cargar dato que identifica que hay proyector datostx=13
NO
Conmutar a Luz Roja13
Cargar dato que identifica que no hay proyector datostx=12
Función envio()
Función void loop()
Función locker14()
Conmutar para abrir la cerradura14
Ingreso de dato del sensor magnético14
NO
¿La puerta14 se abrió ?
SI
Cargar dato que identifica que se abrió datostx=120
Función envio()
Ingreso de dato del sensor magnético14
NO
¿La puerta14 se cerró ?
SI
Cargar dato que identifica que se cerró la puerta datostx=121
Función envio()
Sensar locker 14
¿Hay Proyector14?
SI
Conmutar a Luz Verde14
Cargar dato que identifica que hay proyector datostx=15
NO
Conmutar a Luz Roja14
Cargar dato que identifica que no hay proyector datostx=14
Función envio()
Función void loop()
Función locker17()
Conmutar para abrir la cerradura17
Ingreso de dato del sensor magnético17
NO
¿La puerta17 se abrió ?
SI
Cargar dato que identifica que se abrió datostx=120
Función envio()
Ingreso de dato del sensor magnético17
NO
¿La puerta17 se cerró ?
SI
Cargar dato que identifica que se cerró la puerta datostx=121
Función envio()
Sensar locker 17
¿Hay Proyector17?
SI
Conmutar a Luz Verde17
Cargar dato que identifica que hay proyector datostx=17
NO
Conmutar a Luz Roja17
Cargar dato que identifica que no hay proyector datostx=16
Función envio()
Función void loop()
Función locker18()
Conmutar para abrir la cerradura18
Ingreso de dato del sensor magnético18
NO
¿La puerta18 se abrió ?
SI
Cargar dato que identifica que se abrió datostx=120
Función envio()
Ingreso de dato del sensor magnético18
NO
¿La puerta18 se cerró ?
SI
Cargar dato que identifica que se cerró la puerta datostx=121
Función envio()
Sensar locker 18
¿Hay Proyector18?
SI
Conmutar a Luz Verde18
Cargar dato que identifica que hay proyector datostx=19
NO
Conmutar a Luz Roja18
Cargar dato que identifica que no hay proyector datostx=18
Función envio()
Función void loop()

## ANEXO A-3: Diagrama de flujo Arduino esclavo 2




Función envio()
Comenzar la transmisión al dispositivo 0
Enviar el byte (datostx)
Parar la transmisión
Encerar variables datorx = 0 datostx = 0
Esperar 1 segundo
Regresar a una linea de código después de donde fue llamada la función
return
Función locker3()
Conmutar para abrir la cerradura3
Ingreso de dato del sensor magnético3
¿La puerta3 se abrió ?
NO
SI
Cargar dato que identifica que se abrió datostx=120
Función envio()
Ingreso de dato del sensor magnético3
¿La puerta3 se cerró ?
NO
SI
Cargar dato que identifica que se cerró la puerta datostx=121
Función envio()
Sensar locker 3
¿Hay Proyector3?
SI
NO
Conmutar a Luz Verde3
Cargar dato que identifica que hay proyector datostx=21
Conmutar a Luz Roja3
Cargar dato que identifica que no hay proyector datostx=20
Función envio()
Función void loop()
Función locker4()
Conmutar para abrir la cerradura4
Ingreso de dato del sensor magnético4
¿La puerta4 se abrió ?
NO
SI
Cargar dato que identifica que se abrió datostx=120
Función envio()
Ingreso de dato del sensor magnético4
¿La puerta4 se cerró ?
NO
SI
Cargar dato que identifica que se cerró la puerta datostx=121
Función envio()
Sensar locker 4
¿Hay Proyector4?
SI
NO
Conmutar a Luz Verde4
Cargar dato que identifica que hay proyector datostx=23
Conmutar a Luz Roja4
Cargar dato que identifica que no hay proyector datostx=22
Función envio()
Función void loop()
Función locker7()
Conmutar para abrir la cerradura7
Ingreso de dato del sensor magnético7
¿La puerta7 se abrió ?
NO
SI
Cargar dato que identifica que se abrió datostx=120
Función envio()
Ingreso de dato del sensor magnético7
¿La puerta7 se cerró ?
NO
SI
Cargar dato que identifica que se cerró la puerta datostx=121
Función envio()
Sensar locker 7
¿Hay Proyector7?
SI
NO
Conmutar a Luz Verde7
Cargar dato que identifica que hay proyector datostx=25
Conmutar a Luz Roja7
Cargar dato que identifica que no hay proyector datostx=24
Función envio()
Función void loop()
Función locker8()
Conmutar para abrir la cerradura8
Ingreso de dato del sensor magnético8
¿La puerta8 se abrió ?
NO
SI
Cargar dato que identifica que se abrió datostx=120
Función envio()
Ingreso de dato del sensor magnético8
¿La puerta8 se cerró ?
NO
SI
Cargar dato que identifica que se cerró la puerta datostx=121
Función envio()
Sensar locker 8
¿Hay Proyector8?
SI
NO
Conmutar a Luz Verde8
Cargar dato que identifica que hay proyector datostx=27
Conmutar a Luz Roja8
Cargar dato que identifica que no hay proyector datostx=26
Función envio()
Función void loop()
Función locker11()
Conmutar para abrir la cerradura11
Ingreso de dato del sensor magnético11
¿La puerta11 se abrió ?
NO
SI
Cargar dato que identifica que se abrió datostx=120
Función envio()
Ingreso de dato del sensor magnético11
¿La puerta11 se cerró ?
NO
SI
Cargar dato que identifica que se cerró la puerta datostx=121
Función envio()
Sensar locker 11
¿Hay Proyector11?
SI
NO
Conmutar a Luz Verde11
Cargar dato que identifica que hay proyector datostx=29
Conmutar a Luz Roja11
Cargar dato que identifica que no hay proyector datostx=26
Función envio()
Función void loop()


Función locker12()
Conmutar para abrir la cerradura12
Ingreso de dato del sensor magnético12
NO
¿La puerta12 se abrió ?
SI
Cargar dato que identifica que se abrió datostx=120
Función envio()
Ingreso de dato del sensor magnético10
NO
¿La puerta12 se cerró ?
SI
Cargar dato que identifica que se cerró la puerta datostx=121
Función envio()
Sensar locker 12
¿Hay Proyector12?
SI
Conmutar a Luz Verde12
Cargar dato que identifica que hay proyector datostx=31
NO
Conmutar a Luz Roja12
Cargar dato que identifica que no hay proyector datostx=30
Función envio()
Función void loop()
Función locker15()
Conmutar para abrir la cerradura15
Ingreso de dato del sensor magnético15
NO
¿La puerta15 se abrió ?
SI
Cargar dato que identifica que se abrió datostx=120
Función envio()
Ingreso de dato del sensor magnético15
NO
¿La puerta15 se cerró ?
SI
Cargar dato que identifica que se cerró la puerta datostx=121
Función envio()
Sensar locker 15
¿Hay Proyector15?
SI
Conmutar a Luz Verde15
Cargar dato que identifica que hay proyector datostx=33
NO
Conmutar a Luz Roja15
Cargar dato que identifica que no hay proyector datostx=32
Función envio()
Función void loop()
Función locker16()
Conmutar para abrir la cerradura16
Ingreso de dato del sensor magnético16
NO
¿La puerta16 se abrió ?
SI
Cargar dato que identifica que se abrió datostx=120
Función envio()
Ingreso de dato del sensor magnético16
NO
¿La puerta16 se cerró ?
SI
Cargar dato que identifica que se cerró la puerta datostx=121
Función envio()
Sensar locker 16
¿Hay Proyector16?
SI
Conmutar a Luz Verde16
Cargar dato que identifica que hay proyector datostx=35
NO
Conmutar a Luz Roja16
Cargar dato que identifica que no hay proyector datostx=34
Función envio()
Función void loop()
Función locker19()
Conmutar para abrir la cerradura19
Ingreso de dato del sensor magnético19
NO
¿La puerta19 se abrió ?
SI
Cargar dato que identifica que se abrió datostx=120
Función envio()
Ingreso de dato del sensor magnético19
NO
¿La puerta19 se cerró ?
SI
Cargar dato que identifica que se cerró la puerta datostx=121
Función envio()
Sensar locker 19
¿Hay Proyector19?
SI
Conmutar a Luz Verde19
Cargar dato que identifica que hay proyector datostx=19
NO
Conmutar a Luz Roja19
Cargar dato que identifica que no hay proyector datostx=36
Función envio()
Función void loop()
Función locker20()
Conmutar para abrir la cerradura20
Ingreso de dato del sensor magnético20
NO
¿La puerta20 se abrió ?
SI
Cargar dato que identifica que se abrió datostx=120
Función envio()
Ingreso de dato del sensor magnético20
NO
¿La puerta20 se cerró ?
SI
Cargar dato que identifica que se cerró la puerta datostx=121
Función envio()
Sensar locker 20
¿Hay Proyector20?
SI
Conmutar a Luz Verde20
Cargar dato que identifica que hay proyector datostx=39
NO
Conmutar a Luz Roja20
Cargar dato que identifica que no hay proyector datostx=38
Función envio()
Función void loop()

## ANEXO B-1: Código completo Arduino maestro

```
/*
                                ESCUELA POLITÉCNICA NACIONAL - ESFOT
                                PROYECTO DE TITULACION
                                Control de Lockers
                                Maestro
*/
//////////////////////////////////////////////////////////////////////////////////////////////// LIBRERIAS
#include <Wire.h>       //Librería i2c
//////////////////////////////////////////////////////////////////////////////////////////////// VARIABLES
byte datotx = 0;       //Variable de enviar datos por i2c
byte datorx = 0;       //Variable para recibir datos por i2c
byte mensaje=0;        //Variable para recibir peticiones por comunicación serial
//Variables de estado de lockers (0 no hay proyector) (1 hay proyector):
byte var1=0;
byte var2=0;
byte var3=0;
byte var4=0;
byte var5=0;
byte var6=0;
byte var7=0;
byte var8=0;
byte var9=0;
byte var10=0;
byte var11=0;
byte var12=0;
byte var13=0;
byte var14=0;
byte var15=0;
byte var16=0;
byte var17=0;
byte var18=0;
byte var19=0;
byte var20=0;
byte estado=0;
byte start=5;         // Pin 5 del Arduino para pulsador de inicio del sistema
//////////////////////////////////////////////////////////////////////DECLARACION DE PUERTOS DE ENTRADA Y SALIDA
void setup() {
pinMode (2,OUTPUT);       // Led indicador cuando se transmite un dato al esclavo1
pinMode (3,OUTPUT);       // Led indicador cuando se transmite un dato al esclavo2
pinMode (4,OUTPUT);       // Led indicador cuando recibe un dato por i2c
pinMode (start,INPUT);    // Pulsador de inicio del sistema
digitalWrite(start,HIGH); // Activar el pull-up
////////////////////////////////////////////////////////// UNIR ESTE DISPOSITIVO AL BUS I2C CON DIRECCION 0
Wire.begin(0);
Wire.onReceive(receiveEvent); //Registrar el evento al recibir datos por i2c
Serial.begin(9600);           //// Iniciamos el monitor serie para monitorear la comunicación
////////////////////////////////////////////////////////////////////////////////// ESTADO INICIAL DEL SISTEMA
while(digitalRead(start)==HIGH){}  //Esperar a que se presione el pulsador de inicio
inicio();                          //Ir a la función inicio para sensar todos los puestos
}
////////////////////////////////////////////////////////////////////////////////// COMUNICACIÓN CON SERVIDOR
void loop() {
  if(Serial.available()>0){        // Si se recibe un dato por comunicación serial con el servidor
     mensaje= Serial.read();       // Cargar la variable mensaje con el dato recibido
     if(mensaje == 'u'){presentacion();} // Si mensaje es u ir a la función presentación de variables
     if(mensaje == 'a'||'b'||'e'||'f'||'i'||'j'||'m'||'n'||'q'||'r'){seleccion1();} //Si  mensaje es uno de
los primeros 10 caracteres ir a  seleccion1 (para enviar dato al esclavo1)
     if(mensaje == 'c'||'d'||'g'||'h'||'k'||'l'||'o'||'p'||'s'||'t'){seleccion2();} //Si  mensaje es uno de
los siguientes 10 caracteres ir a  seleccion2 (para enviar dato al esclavo2)
  }
}

////////////////////////////////////////////////////////////////////////////// FUNCION DE INICIO DEL SISTEMA
void inicio(){   // Función de inicio para que  el sistema empiece sensando los lockers
datotx=100;      // Cargar  dato a transmitir al esclavo 1 para que empiece a sensar
envio1();        // Llamar a la función de envío al esclavo 1
}
////////////////////////////////////////////// FUNCION DE PRESENTACION DE VARIABLES POR COMUNICACIÓN SERIAL
void presentacion(){  //Función para mostrar el estado de los lockers (0 desocupado) (1 ocupado)
   Serial.println((String)"Estado:"+estado);  // Presentar el estado de las puertas (1 abierto) (0 cerrado)
   Serial.println((String)"Proyector1:"+var1);
   Serial.println((String)"Proyector2:"+var2);
   Serial.println((String)"Proyector5:"+var5);
   Serial.println((String)"Proyector6:"+var6);
   Serial.println((String)"Proyector9:"+var9);
   Serial.println((String)"Proyector10:"+var10);
```

```
   Serial.println((String)"Proyector13:"+var13);
   Serial.println((String)"Proyector14:"+var14);
   Serial.println((String)"Proyector17:"+var17);
   Serial.println((String)"Proyector18:"+var18);
   Serial.println((String)"Proyector3:"+var3);
   Serial.println((String)"Proyector4:"+var4);
   Serial.println((String)"Proyector7:"+var7);
   Serial.println((String)"Proyector8:"+var8);
   Serial.println((String)"Proyector11:"+var11);
   Serial.println((String)"Proyector12:"+var12);
   Serial.println((String)"Proyector15:"+var15);
   Serial.println((String)"Proyector16:"+var16);
   Serial.println((String)"Proyector19:"+var19);
   Serial.println((String)"Proyector20:"+var20);

   Serial.end();        // Finalizar la comunicación serial después de enviar todas las variables al servidor
   Serial.begin(9600);  // Inicializar el serial para espera de datos por i2c
   delay(250);          // Pausa de 250 milisegundos
   mensaje = 0;         // Encerar datos recibidos del servidor
}
////////////////////////////////////////////////// FUNCION DE PETICION DE UNO DE LOS LOCKERS DEL ESCLAVO 1
void seleccion1(){    // Función
switch (mensaje) {    // Comparar a que caso pertenece el dato recibido del servidor
      case 'a':       // Locker 1
      datotx=1;       // Cargar número único que identifica el locker 1
      envio1();       // Enviar dato al esclavo 1
      break;

      case 'b':       // Locker 2
      datotx=2;       // Cargar número único que identifica el locker 2
      envio1();       // Enviar dato al esclavo 1
      break;

      case 'e':       // Locker 5
      datotx=5;     // Cargar número único que identifica el locker 5
      envio1();       // Enviar dato al esclavo 1
      break;

      case 'f':       // Locker 6
      datotx=6;       // Cargar número único que identifica el locker 6
      envio1();       // Enviar dato al esclavo 1
      break;

      case 'i':       // Locker 9
      datotx=9;       // Cargar número único que identifica el locker 9
      envio1();       // Enviar dato al esclavo 1
      break;

      case 'j':       // Locker 10
      datotx=10;      // Cargar número único que identifica el locker 10
      envio1();       // Enviar dato al esclavo 1
      break;

      case 'm':       // Locker 13
      datotx=13;      // Cargar número único que identifica el locker 13
      envio1();       // Enviar dato al esclavo 1
      break;

      case 'n':       // Locker 14
      datotx=14;      // Cargar número único que identifica el locker 14
      envio1();       // enviar dato al esclavo 1
      break;

      case 'q':       // Locker 17
      datotx=17;      // Cargar número único que identifica el locker 17
      envio1();       // Enviar dato al esclavo 1
      break;

      case 'r':       // Locker 18
      datotx=18;      // Cargar número único que identifica el locker 18
      envio1();       // Enviar dato al esclavo 1
      break;
      }
  }
////////////////////////////////////////////////// FUNCION DE PETICION DE UNO DE LOS LOCKERS DEL ESCLAVO 2
void seleccion2(){    // Función
switch (mensaje) {    // Comparar a que caso pertenece el dato recibido del servidor
      case 'c':       // Locker 3
      datotx=3;       // Cargar número único que identifica el locker 3
      envio2();       // Enviar dato al esclavo 2
      break;
```

```
      case 'd':       // Locker 4
      datotx=4;       // Cargar número único que identifica el locker 4
      envio2();       // Enviar dato al esclavo 2
      break;

      case 'g':       // Locker 7
      datotx=7;       // Cargar número único que identifica el locker 7
      envio2();       // Enviar dato al esclavo 2
      break;

      case 'h':       // Locker 8
      datotx=8;       // Cargar número único que identifica el locker 8
      envio2();       // Enviar dato al esclavo 2
      break;

      case 'k':       // Locker 11
      datotx=11;      // Cargar número único que identifica el locker 11
      envio2();       // Enviar dato al esclavo 2
      break;

      case 'l':       // Locker 12
      datotx=12;      // Cargar número único que identifica el locker 12
      envio2();       // Enviar dato al esclavo 2
      break;

      case 'o':       // Locker 15
      datotx=15;      // Cargar número único que identifica el locker 15
      envio2();       // Enviar dato al esclavo 2
      break;

      case 'p':       // Locker 16
      datotx=16;      // Cargar número único que identifica el locker 16
      envio2();       // Enviar dato al esclavo 2
      break;

      case 's':       // Locker 19
      datotx=19;      // Cargar número único que identifica el locker 19
      envio2();       // Enviar dato al esclavo 2
      break;

      case 't':       // Locker 20
      datotx=20;      // Cargar número único que identifica el locker 20
      envio2();       // Enviar dato al esclavo 2
      break;
 }
}
////////////////////////////////////////////////////////////////////////FUNCION CUANDO SE RECIBE UN DATO POR I2C
void receiveEvent() {
digitalWrite(4,HIGH);           //Encender led indicador de que se recibió un dato de los esclavos
if (Wire.available() == 1){     // Si hay un byte disponible
    datorx = Wire.read();       // Leer  y cargar la variable
    if ( datorx % 2 == 0){      // Verificar si el dato que se recibió es par o impar
        par();                  // Si es par, ir a la función par para indicar que no hay casilleros
    }
    else{
        impar();                // Si es impar, ir a la función impar para indicar que si hay casilleros
    }
}
delay(500);                     // Pausa de medio segundo
digitalWrite(4,LOW);            // Apagar led indicador de que se recibió un dato de los esclavos
}
//// FUNCION PARA CARGAR VARIABLES DE PRESENTACION CON EL SERVIDOR (SI NO HAY PROYECTORES Y PUERTAS ABIERTAS)
void par(){
switch (datorx) {  //Identificar valor del dato que se recibió por i2c para cargar con 0 la variable de
presentación de cada locker
        ////////////////////////////////////1
        case 0:
        var1=0;
        datorx=0;
        break;
        ////////////////////////////////////2
        case 2:
        var2=0;
        datorx=0;
        break;
        ////////////////////////////////////3
        case 4:
        var3=0;
        datorx=0;
        break;
        ////////////////////////////////////4
        case 6:
```

```
var4=0;
datorx=0;
break;
////////////////////////////////////5
case 8:
var5=0;
datorx=0;
break;
////////////////////////////////////6
case 10:
var6=0;
datorx=0;
break;
////////////////////////////////////7
case 12:
var7=0;
datorx=0;
break;
////////////////////////////////////8
case 14:
var8=0;
datorx=0;
break;
////////////////////////////////////9
case 16:
var9=0;
datorx=0;
break;
////////////////////////////////////10
case 18:
var10=0;
datorx=0;
break;
////////////////////////////////////11
case 20:
var11=0;
datorx=0;
break;
////////////////////////////////////12
case 22:
var12=0;
datorx=0;
break;
////////////////////////////////////13
case 24:
var13=0;
datorx=0;
break;
////////////////////////////////////14
case 26:
var14=0;
datorx=0;
break;
////////////////////////////////////15
case 28:
var15=0;
datorx=0;
break;
////////////////////////////////////16
case 30:
var16=0;
datorx=0;
break;
////////////////////////////////////17
case 32:
var17=0;
datorx=0;
break;
////////////////////////////////////18
case 34:
var18=0;
datorx=0;
break;
////////////////////////////////////19
case 36:
var19=0;
datorx=0;
break;
////////////////////////////////////20
case 38:
var20=0;
datorx=0;
```

```
        break;
        ////////////////////////////////////estado=1 cuando una puerta está abierta
        case 120:
        estado=1;
        datorx=0;
        break;

}
}
//// FUNCION PARA CARGAR VARIABLES DE PRESENTACION CON EL SERVIDOR (SI HAY PROYECTORES Y PUERTAS CERRADAS),
ADEMAS DEL DATO DE FINILIZACION DE LA PRIMERA VEZ QUE SE SENSA.
void impar(){
switch (datorx) {   //Identificar valor del dato que se recibió por i2c para cargar con 1 la variable de
presentación de cada locker
          ////////////////////////////////////1
          case 1:
          var1=1;
          datorx=0;
          break;
          ////////////////////////////////////2
          case 3:
          var2=1;
          datorx=0;
          break;
          ////////////////////////////////////3
          case 5:
          var3=1;
          datorx=0;
          break;
          ////////////////////////////////////4
          case 7:
          var4=1;
          datorx=0;
          break;
          ////////////////////////////////////5
          case 9:
          var5=1;
          datorx=0;
          break;
          ////////////////////////////////////6
          case 11:
          var6=1;
          datorx=0;
          break;
          ////////////////////////////////////7
          case 13:
          var7=1;
          datorx=0;
          break;
          ////////////////////////////////////8
          case 15:
          var8=1;
          datorx=0;
          break;
          ////////////////////////////////////9
          case 17:
          var9=1;
          datorx=0;
          break;
          ////////////////////////////////////10
          case 19:
          var10=1;
          datorx=0;
          break;
          ////////////////////////////////////11
          case 21:
          var11=1;
          datorx=0;
          break;
          ////////////////////////////////////12
          case 23:
          var12=1;
          datorx=0;
          break;
          ////////////////////////////////////13
          case 25:
          var13=1;
          datorx=0;
          break;
          ////////////////////////////////////14
          case 27:
          var14=1;
```

```
          datorx=0;
          break;
          ////////////////////////////////////15
          case 29:
          var15=1;
          datorx=0;
          break;
          ////////////////////////////////////16
          case 31:
          var16=1;
          datorx=0;
          break;
          ////////////////////////////////////17
          case 33:
          var17=1;
          datorx=0;
          break;
          ////////////////////////////////////18
          case 35:
          var18=1;
          datorx=0;
          break;
          ////////////////////////////////////19
          case 37:
          var19=1;
          datorx=0;
          break;
          ////////////////////////////////////20
          case 39:
          var20=1;
          datorx=0;
          break;
          ////////////////////////////////////estado=0 todo está cerrado
          case 121:
          estado=0;
          datorx=0;
          break;
          ////////////////////////////Dato que envía el esclavo2 cuando termina de sensar por primera vez
          case 201:
          datorx=0;
          presentacion(); // Presentar todas las variables del estado de los lockers por primera vez
          break;
}
}

/////////////////////////////////////////////////////////////////// FUNCIONES DE ENVIO DE DATOS A LOS ESCLAVOS
void envio1(){                          // FUNCION PARA ENVIAR DATOS AL ESCLAVO 1
    digitalWrite(2,HIGH);               // Encender led indicador de envío de datos al esclavo 1
    Wire.beginTransmission(1);          // Comenzar la transmisión al dispositivo 1
    Wire.write(datotx);                 // Enviar un byte
    Wire.endTransmission();             // Parar la transmisión
    mensaje=0;                          // Encerar variables
    datotx=0;
    delay(1000);                        // Esperar 1 segundo
    digitalWrite(2,LOW);                // Apagar led indicador de envío de datos al esclavo 1
}

void envio2(){                          // FUNCION PARA ENVIAR DATOS AL ESCLAVO 2
    digitalWrite(3,HIGH);               // Encender led indicador de envío de datos al esclavo 2
    Wire.beginTransmission(2);          // Comenzar la transmisión al dispositivo 2
    Wire.write(datotx);                 // Enviar un byte
    Wire.endTransmission();             // Parar la transmisión
    mensaje=0;                          // Encerar variables
    datotx=0;
    delay(1000);                        // Esperar 1 segundo
    digitalWrite(3,LOW);                // Apagar led indicador de envío de datos al esclavo 2
}
```

## ANEXO B-2: Código completo Arduino esclavo 1

```
/*
                                    ESCUELA POLITÉCNICA NACIONAL - ESFOT
                                          PROYECTO DE TITULACION
                                            Control de Lockers
                                                Esclavo 1
*/
////////////////////////////////////////////////////////////////////////////////////////////////////////// LIBRERIAS
#include <Wire.h>
////////////////////////////////////////////////////////////////////////////////////////////////////////// VARIABLES
byte datorx = 0;        //Variable para recibir datos por i2c
byte datostx=0;         //Variable de enviar datos por i2c
byte a=0;               //Variable a para establecer un contador (tiempo para sensar con infrarrojo)
byte respuesta=0;       //Variable respuesta para leer datos de sensor infrarrojo
//Variables para manejar los puertos I/O con nombres propios para cada locker:
//////////////////////////////////////////////////////////////Locker 1
byte magnetopin1=4;     //Lectura del sensor magnético
byte infrapin1=53;      //Lectura del sensor infrarrojo
byte ledpin1=25;        //Activación de luces
byte cerradurapin1=28; //Activación de cerradura
byte activepin1=36;     //Activación del infrarrojo
//////////////////////////////////////////////////////////////Locker 2
byte magnetopin2=5;     //Lectura del sensor magnético
byte infrapin2=51;      //Lectura del sensor infrarrojo
byte ledpin2=26;        //Activación de luces
byte cerradurapin2=27; //Activación de cerradura
byte activepin2=37;     //Activación del infrarrojo
//////////////////////////////////////////////////////////////Locker 5
byte magnetopin3=6;     //Lectura del sensor magnético
byte infrapin3=19;      //Lectura del sensor infrarrojo
byte ledpin3=23;        //Activación de luces
byte cerradurapin3=30; //Activación de cerradura
byte activepin3=34;     //Activación del infrarrojo
//////////////////////////////////////////////////////////////Locker 6
byte magnetopin4=7;     //Lectura del sensor magnético
byte infrapin4=18;      //Lectura del sensor infrarrojo
byte ledpin4=24;        //Activación de luces
byte cerradurapin4=29; //Activación de cerradura
byte activepin4=35;     //Activación del infrarrojo
//////////////////////////////////////////////////////////////Locker 9
byte magnetopin5=8;     //Lectura del sensor magnético
byte infrapin5=17;      //Lectura del sensor infrarrojo
byte ledpin5=22;        //Activación de luces
byte cerradurapin5=31; //Activación de cerradura
byte activepin5=33;     //Activación del infrarrojo
//////////////////////////////////////////////////////////////Locker 10
byte magnetopin6=9;     //Lectura del sensor magnético
byte infrapin6=16;      //Lectura del sensor infrarrojo
byte ledpin6=42;        //Activación de luces
byte cerradurapin6=43; //Activación de cerradura
//byte activepin6=49;  //Activación del infrarrojo
//////////////////////////////////////////////////////////////Locker 13
byte magnetopin7=10;    //Lectura del sensor magnético
byte infrapin7=15;      //Lectura del sensor infrarrojo
byte ledpin7=40;        //Activación de luces
byte cerradurapin7=44; //Activación de cerradura
byte activepin7=48;     //Activación del infrarrojo
//////////////////////////////////////////////////////////////Locker 14
byte magnetopin8=11;    //Lectura del sensor magnético
byte infrapin8=14;      //Lectura del sensor infrarrojo
byte ledpin8=41;        //Activación de luces
byte cerradurapin8=45; //Activación de cerradura
//byte activepin8=;     //Activación del infrarrojo
//////////////////////////////////////////////////////////////Locker 17
byte magnetopin9=12;    //Lectura del sensor magnético
byte infrapin9=2;       //Lectura del sensor infrarrojo
byte ledpin9=38;        //Activación de luces
byte cerradurapin9=46; //Activación de cerradura
byte activepin9=50;     //Activación del infrarrojo
//////////////////////////////////////////////////////////////Locker 18
byte magnetopin10=13;   //Lectura del sensor magnético
byte infrapin10=3;      //Lectura del sensor infrarrojo
byte ledpin10=39;       //Activación de luces
byte cerradurapin10=47; //Activación de cerradura
byte activepin10=49;    //Activación del infrarrojo
//////////////////////////////////////////////////////////////////////////DECLARACION DE PUERTOS DE ENTRADA Y
SALIDA
void setup() {
//////////////////////////////////////////////////////////////Locker 1
pinMode (ledPin1,OUTPUT);
```

```
pinMode (cerraduraPin1,OUTPUT);
pinMode (activePin1,OUTPUT);
digitalWrite (ledPin1,HIGH);
digitalWrite (cerraduraPin1,HIGH);
digitalWrite (activePin1,HIGH);
pinMode (infraPin1,INPUT);
pinMode (magnetoPin1,INPUT);
digitalWrite (infraPin1,HIGH);
digitalWrite (magnetoPin1,HIGH);
//////////////////////////////////////////////////////////Locker 2
pinMode (ledPin2,OUTPUT);
pinMode (cerraduraPin2,OUTPUT);
pinMode (activePin2,OUTPUT);
digitalWrite (ledPin2,HIGH);
digitalWrite (cerraduraPin2,HIGH);
digitalWrite (activePin2,HIGH);
pinMode (infraPin2,INPUT);
pinMode (magnetoPin2,INPUT);
digitalWrite (infraPin2,HIGH);
digitalWrite (magnetoPin2,HIGH);
//////////////////////////////////////////////////////////Locker 5
pinMode (ledPin3,OUTPUT);
pinMode (cerraduraPin3,OUTPUT);
pinMode (activePin3,OUTPUT);
digitalWrite (ledPin3,HIGH);
digitalWrite (cerraduraPin3,HIGH);
digitalWrite (activePin3,HIGH);
pinMode (infraPin3,INPUT);
pinMode (magnetoPin3,INPUT);
digitalWrite (infraPin3,HIGH);
digitalWrite (magnetoPin3,HIGH);
//////////////////////////////////////////////////////////Locker 6
pinMode (ledPin4,OUTPUT);
pinMode (cerraduraPin4,OUTPUT);
pinMode (activePin4,OUTPUT);
digitalWrite (ledPin4,HIGH);
digitalWrite (cerraduraPin4,HIGH);
digitalWrite (activePin4,HIGH);
pinMode (infraPin4,INPUT);
pinMode (magnetoPin4,INPUT);
digitalWrite (infraPin4,HIGH);
digitalWrite (magnetoPin4,HIGH);
//////////////////////////////////////////////////////////Locker 9
pinMode (ledPin5,OUTPUT);
pinMode (cerraduraPin5,OUTPUT);
pinMode (activePin5,OUTPUT);
digitalWrite (ledPin5,HIGH);
digitalWrite (cerraduraPin5,HIGH);
digitalWrite (activePin5,HIGH);
pinMode (infraPin5,INPUT);
pinMode (magnetoPin5,INPUT);
digitalWrite (infraPin5,HIGH);
digitalWrite (magnetoPin5,HIGH);
//////////////////////////////////////////////////////////Locker 10
pinMode (ledPin6,OUTPUT);
pinMode (cerraduraPin6,OUTPUT);
//pinMode (activePin6,OUTPUT);
digitalWrite (ledPin6,HIGH);
digitalWrite (cerraduraPin6,HIGH);
pinMode (infraPin6,INPUT);
pinMode (magnetoPin6,INPUT);
digitalWrite (infraPin6,HIGH);
digitalWrite (magnetoPin6,HIGH);
//////////////////////////////////////////////////////////Locker 13
pinMode (ledPin7,OUTPUT);
pinMode (cerraduraPin7,OUTPUT);
pinMode (activePin7,OUTPUT);
digitalWrite (ledPin7,HIGH);
digitalWrite (cerraduraPin7,HIGH);
digitalWrite (activePin7,HIGH);
pinMode (infraPin7,INPUT);
pinMode (magnetoPin7,INPUT);
digitalWrite (infraPin7,HIGH);
digitalWrite (magnetoPin7,HIGH);
//////////////////////////////////////////////////////////Locker 14
pinMode (ledPin8,OUTPUT);
pinMode (cerraduraPin8,OUTPUT);
//pinMode (activePin8,OUTPUT);
digitalWrite (ledPin8,HIGH);
digitalWrite (cerraduraPin8,HIGH);
pinMode (infraPin8,INPUT);
pinMode (magnetoPin8,INPUT);
```

```
digitalWrite (infraPin8,HIGH);
digitalWrite (magnetoPin8,HIGH);
////////////////////////////////////////////////////////Locker 17
pinMode (ledPin9,OUTPUT);
pinMode (cerraduraPin9,OUTPUT);
pinMode (activePin9,OUTPUT);
digitalWrite (ledPin9,HIGH);
digitalWrite (cerraduraPin9,HIGH);
digitalWrite (activePin9,HIGH);
pinMode (infraPin9,INPUT);
pinMode (magnetoPin9,INPUT);
digitalWrite (infraPin9,HIGH);
digitalWrite (magnetoPin9,HIGH);
////////////////////////////////////////////////////////Locker 18
pinMode (ledPin10,OUTPUT);
pinMode (cerraduraPin10,OUTPUT);
pinMode (activePin10,OUTPUT);
digitalWrite (ledPin10,HIGH);
digitalWrite (cerraduraPin10,HIGH);
digitalWrite (activePin10,HIGH);
pinMode (infraPin10,INPUT);
pinMode (magnetoPin10,INPUT);
digitalWrite (infraPin10,HIGH);
digitalWrite (magnetoPin10,HIGH);

pinMode (52,OUTPUT);          //Led indicador cuando recibe un dato por i2c
////////////////////////////////////////////////////////// UNIR ESTE DISPOSITIVO AL BUS I2C CON DIRECCION
1
Wire.begin(1);
Wire.onReceive(receiveEvent); // Registrar el evento al recibir datos por i2c
Serial.begin(9600);           // Iniciar el monitor serie para monitorear la comunicación
}
/////////////////////////////////////IDENTIFICACION DE LOCKER MEDIANTE EL DATO RECIBIDO POR PARTE DEL
MAESTRO
void loop(){                  // Funcion loop
  switch (datorx){            // Comparar a que caso pertenece el dato recibido del maestro
  case 1:                     // Número único que identifica el locker 1
    locker1();                // Ir a la función de petición del locker 1
    break;
  case 2:                     // Número único que identifica el locker 2
    locker2();                // Ir a la función de petición del locker 2
    break;
  case 5:                     // Número único que identifica el locker 5
    locker5();                // Ir a la función de petición del locker 5
    break;
  case 6:                     // Número único que identifica el locker 6
    locker6();                // Ir a la función de petición del locker 6
    break;
  case 9:                     // Número único que identifica el locker 9
    locker9();                // Ir a la función de petición del locker 9
    break;
  case 10:                    // Número único que identifica el locker 10
    locker10();               // Ir a la función de petición del locker 10
    break;
  case 13:                    // Número único que identifica el locker 13
    locker13();               // Ir a la función de petición del locker 13
    break;
  case 14:                    // Número único que identifica el locker 14
    locker14();               // Ir a la función de petición del locker 14
    break;
  case 17:                    // Número único que identifica el locker 17
    locker17();               // Ir a la función de petición del locker 17
    break;
  case 18:                    // Número único que identifica el locker 18
    locker18();               // Ir a la función de petición del locker 18
    break;
  case 100:                   // Dato que se recibe para sensar todos los lockers.
    inicio();                 // Ir a la función inicio para sensar todos los puestos
    break;
  }
}
////////////////////////////////////////////////////////////////////////// FUNCION PARA SENSAR TODOS LOS
LOCKERS
void inicio(){
////////////////////////////////////////////////////////////////////////////////////////////////Locker 1
  digitalWrite(activePin1,LOW);             // Activar sensor infrarrojo
  for(a=0;a<10000;a++){                     // Empieza a sensar  el infrarrojo
      respuesta=digitalRead(infraPin1);}    // Guardar el valor en la variable respuesta (0 o 1)
  digitalWrite(activePin1,HIGH);            // Apagar sensor infrarrojo
  if(respuesta==LOW){                       // Si es 0 detecta proyector
      digitalWrite(ledPin1,LOW);            // Encender luz verde
      datostx=1;                            // Cargar dato que identifica que hay proyector
```

```
      envio();}                              // Enviar dato que si hay proyector al maestro
  else{                                      // Si es 1 no hay proyector
      digitalWrite(ledPin1,HIGH);            // Encender luz roja
      datostx=0;                             // Cargar dato que identifica que no hay proyector
      envio();}                              // Enviar dato que no hay proyector al maestro
////////////////////////////////////////////////////////////////////////////////////////////////Locker 2
digitalWrite(activePin2,LOW);                // Activar sensor infrarrojo
  for(a=0;a<10000;a++){                      // Empieza a sensar  el infrarrojo
      respuesta=digitalRead(infraPin2);}     // Guardar el valor en la variable respuesta (0 o 1)
digitalWrite(activePin2,HIGH);               // Apagar sensor infrarrojo
  if(respuesta==LOW){                        // Si es 0 detecta proyector
      digitalWrite(ledPin2,LOW);             // Encender luz verde
      datostx=3;                             // Cargar dato que identifica que hay proyector
      envio();}                              // Enviar dato que si hay proyector al maestro
  else{                                      // Si es 1 no hay proyector
      digitalWrite(ledPin2,HIGH);            // Encender luz roja
      datostx=2;                             // Cargar dato que identifica que no hay proyector
      envio();}                              // Enviar dato que no hay proyector al maestro
////////////////////////////////////////////////////////////////////////////////////////////////Locker 5
digitalWrite(activePin3,LOW);                // Activar sensor infrarrojo
  for(a=0;a<10000;a++){                      // Empieza a sensar  el infrarrojo
      respuesta=digitalRead(infraPin3);}     // Guardar el valor en la variable respuesta (0 o 1)
digitalWrite(activePin3,HIGH);               // Apagar sensor infrarrojo
  if(respuesta==LOW){                        // Si es 0 detecta proyector
      digitalWrite(ledPin3,LOW);             // Encender luz verde
      datostx=5;                             // Cargar dato que identifica que hay proyector
      envio();}                              // Enviar dato que si hay proyector al maestro
  else{                                      // Si es 1 no hay proyector
      digitalWrite(ledPin3,HIGH);            // Encender luz roja
      datostx=4;                             // Cargar dato que identifica que no hay proyector
      envio();}                              // Enviar dato que no hay proyector al maestro
////////////////////////////////////////////////////////////////////////////////////////////////Locker 6
digitalWrite(activePin4,LOW);                // Activar sensor infrarrojo
  for(a=0;a<10000;a++){                      // Empieza a sensar  el infrarrojo
      respuesta=digitalRead(infraPin4);}     // Guardar el valor en la variable respuesta (0 o 1)
digitalWrite(activePin4,HIGH);               // Apagar sensor infrarrojo
  if(respuesta==LOW){                        // Si es 0 detecta proyector
      digitalWrite(ledPin4,LOW);             // Encender luz verde
      datostx=7;                             // Cargar dato que identifica que hay proyector
      envio();}                              // Enviar dato que si hay proyector al maestro
  else{                                      // Si es 1 no hay proyector
      digitalWrite(ledPin4,HIGH);            // Encender luz roja
      datostx=6;                             // Cargar dato que identifica que no hay proyector
      envio();}                              // Enviar dato que no hay proyector al maestro
////////////////////////////////////////////////////////////////////////////////////////////////Locker 9
digitalWrite(activePin5,LOW);                // Activar sensor infrarrojo
  for(a=0;a<10000;a++){                      // Empieza a sensar  el infrarrojo
      respuesta=digitalRead(infraPin5);}     // Guardar el valor en la variable respuesta (0 o 1)
digitalWrite(activePin5,HIGH);               // Apagar sensor infrarrojo
  if(respuesta==LOW){                        // Si es 0 detecta proyector
      digitalWrite(ledPin5,LOW);             // Encender luz verde
      datostx=9;                             // Cargar dato que identifica que hay proyector
      envio();}                              // Enviar dato que si hay proyector al maestro
  else{                                      // Si es 1 no hay proyector
      digitalWrite(ledPin5,HIGH);            // Encender luz roja
      datostx=8;                             // Cargar dato que identifica que no hay proyector
      envio();}                              // Enviar dato que no hay proyector al maestro
////////////////////////////////////////////////////////////////////////////////////////////////Locker 10
//digitalWrite(activePin6,HIGH);             // Activar sensor infrarrojo(siempre activo)
  for(a=0;a<10000;a++){                      // Empieza a sensar  el infrarrojo
      respuesta=digitalRead(infraPin6);}     // Guardar el valor en la variable respuesta (0 o 1)
//digitalWrite(activePin6,LOW);
  if(respuesta==LOW){                        // Si es 0 detecta proyector
      digitalWrite(ledPin6,LOW);             // Encender luz verde
      datostx=11;                            // Cargar dato que identifica que hay proyector
      envio();}                              // Enviar dato que si hay proyector al maestro
  else{                                      // Si es 1 no hay proyector
      digitalWrite(ledPin6,HIGH);            // Encender luz roja
      datostx=10;                            // Cargar dato que identifica que no hay proyector
      envio();}                              // Enviar dato que no hay proyector al maestro
////////////////////////////////////////////////////////////////////////////////////////////////Locker 13
digitalWrite(activePin7,LOW);                // Activar sensor infrarrojo
  for(a=0;a<10000;a++){                      // Empieza a sensar  el infrarrojo
      respuesta=digitalRead(infraPin7);}     // Guardar el valor en la variable respuesta (0 o 1)
digitalWrite(activePin7,HIGH);               // Apagar sensor infrarrojo
  if(respuesta==LOW){                        // Si es 0 detecta proyector
      digitalWrite(ledPin7,LOW);             // Encender luz verde
      datostx=13;                            // Cargar dato que identifica que hay proyector
      envio();}                              // Enviar dato que si hay proyector al maestro
  else{                                      // Si es 1 no hay proyector
      digitalWrite(ledPin7,HIGH);            // Encender luz roja
      datostx=12;                            // Cargar dato que identifica que no hay proyector
```

```
      envio();}                              // Enviar dato que no hay proyector al maestro
////////////////////////////////////////////////////////////////////////////////////////////////Locker 14
//digitalWrite(activePin8,LOW);              // Activar sensor infrarrojo (siempre activo)
  for(a=0;a<10000;a++){                        // Empieza a sensar  el infrarrojo
      respuesta=digitalRead(infraPin8);}       // Guardar el valor en la variable respuesta (0 o 1)
//digitalWrite(activePin8,HIGH);
  if(respuesta==LOW){                          // Si es 0 detecta proyector
      digitalWrite(ledPin8,LOW);               // Encender luz verde
      datostx=15;                              // Cargar dato que identifica que hay proyector
      envio();}                                // Enviar dato que si hay proyector al maestro
  else{                                        // Si es 1 no hay proyector
      digitalWrite(ledPin8,HIGH);              // Encender luz roja
      datostx=14;                              // Cargar dato que identifica que no hay proyector
      envio();}                                // Enviar dato que no hay proyector al maestro
////////////////////////////////////////////////////////////////////////////////////////////////Locker 17
digitalWrite(activePin9,LOW);                  // Activar sensor infrarrojo
  for(a=0;a<10000;a++){                        // Empieza a sensar  el infrarrojo
      respuesta=digitalRead(infraPin9);}       // Guardar el valor en la variable respuesta (0 o 1)
digitalWrite(activePin9,HIGH);                 // Apagar sensor infrarrojo
  if(respuesta==LOW){                          // Si es 0 detecta proyector
      digitalWrite(ledPin9,LOW);               // Encender luz verde
      datostx=17;                              // Cargar dato que identifica que hay proyector
      envio();}                                // Enviar dato que si hay proyector al maestro
  else{                                        // Si es 1 no hay proyector
      digitalWrite(ledPin9,HIGH);              // Encender luz roja
      datostx=16;                              // Cargar dato que identifica que no hay proyector
      envio();}                                // Enviar dato que no hay proyector al maestro
////////////////////////////////////////////////////////////////////////////////////////////////Locker 18
digitalWrite(activePin10,LOW);                 // Activar sensor infrarrojo
  for(a=0;a<10000;a++){                        // Empieza a sensar  el infrarrojo
      respuesta=digitalRead(infraPin10);}      // Guardar el valor en la variable respuesta (0 o 1)
digitalWrite(activePin10,HIGH);                // Apagar sensor infrarrojo
  if(respuesta==LOW){                          // Si es 0 detecta proyector
      digitalWrite(ledPin10,LOW);              // Encender luz verde
      datostx=19;                              // Cargar dato que identifica que hay proyector
      envio();}                                // Enviar dato que si hay proyector al maestro
  else{                                        // Si es 1 no hay proyector
      digitalWrite(ledPin10,HIGH);             // Encender luz roja
      datostx=18;                              // Cargar dato que identifica que no hay proyector
      envio();}                                // Enviar dato que no hay proyector al maestro
//////////////////////////////////////////////////////////////////////////////////////////////// Esclavo 2
datostx=200                    // Cargar  dato a transmitir al esclavo 2 para que empiece a sensar
envio2();                     // Llamar a la función de envío al esclavo 2
}
//////////////////////////////////////////////////////////////////////////////// FUNCIÓN DE PETICION DEL
LOCKER 1
void locker1(){
  digitalWrite(cerraduraPin1,LOW);             // Abrir la cerradura
  while(digitalRead(magnetoPin1)==LOW){}       // Esperar hasta que se separe la puerta
  datostx=120;                                 // Cargar dato que identifica que está abierta la puerta
  envio();                                     // Enviar dato al maestro
  while(digitalRead(magnetoPin1)==HIGH){}      // Espera hasta que se cierre la puerta
  digitalWrite(cerraduraPin1,HIGH);            // Cerrar la cerradura
  datostx=121;                                 // Cargar dato que identifica que está cerrada la puerta
  envio();                                     // Enviar dato al maestro
  digitalWrite(activePin1,LOW);                // Activar sensor infrarrojo
  for(a=0;a<10000;a++){                        // Empieza a sensar  el infrarrojo
      respuesta=digitalRead(infraPin1);}       // Guardar el valor en la variable respuesta (0 o 1)
  digitalWrite(activePin1,HIGH);               // Apagar sensor infrarrojo
  if(respuesta==LOW){                          // Si es 0 detecta proyector
      digitalWrite(ledPin1,LOW);               // Encender luz verde
      datostx=1;                               // Cargar dato que identifica que hay proyector
      envio();}                                // Enviar dato que si hay proyector al maestro
  else{                                        // Si es 1 no hay proyector
      digitalWrite(ledPin1,HIGH);              // Encender luz roja
      datostx=0;                               // Cargar dato que identifica que no hay proyector
      envio();}                                // Enviar dato que no hay proyector al maestro
}
//////////////////////////////////////////////////////////////////////////////// FUNCIÓN DE PETICION DEL
LOCKER 2
void locker2(){
  digitalWrite(cerraduraPin2,LOW);             // Abrir la cerradura
  while(digitalRead(magnetoPin2)==LOW){}       // Esperar hasta que se separe la puerta
  datostx=120;                                 // Cargar dato que identifica que está abierta la puerta
  envio();                                     // Enviar dato al maestro
  while(digitalRead(magnetoPin2)==HIGH){}      // Espera hasta que se cierre la puerta
  digitalWrite(cerraduraPin2,HIGH);            // Cerrar la cerradura
  datostx=121;                                 // Cargar dato que identifica que está cerrada la puerta
  envio();                                     // Enviar dato al maestro
  digitalWrite(activePin2,LOW);                // Activar sensor infrarrojo
  for(a=0;a<10000;a++){                        // Empieza a sensar  el infrarrojo
      respuesta=digitalRead(infraPin2);}       // Guardar el valor en la variable respuesta (0 o 1)
```

```
      digitalWrite(activePin2,HIGH);            // Apagar sensor infrarrojo
  if(respuesta==LOW){                           // Si es 0 detecta proyector
      digitalWrite(ledPin2,LOW);                // Encender luz verde
      datostx=3;                                // Cargar dato que identifica que hay proyector
      envio();}                                 // Enviar dato que si hay proyector al maestro
  else{                                         // Si es 1 no hay proyector
      digitalWrite(ledPin2,HIGH);               // Encender luz roja
      datostx=2;                                // Cargar dato que identifica que no hay proyector
      envio();}                                 // Enviar dato que no hay proyector al maestro
}
////////////////////////////////////////////////////////////////////////////// FUNCIÓN DE PETICION DEL
LOCKER 5
void locker5(){
 digitalWrite(cerraduraPin3,LOW);               // Abrir la cerradura
  while(digitalRead(magnetoPin3)==LOW){}        // Esperar hasta que se separe la puerta
  datostx=120;                                  // Cargar dato que identifica que está abierta la puerta
  envio();                                      // Enviar dato al maestro
  while(digitalRead(magnetoPin3)==HIGH){}       // Espera hasta que se cierre la puerta
  digitalWrite(cerraduraPin3,HIGH);             // Cerrar la cerradura
  datostx=121;                                  // Cargar dato que identifica que está cerrada la puerta
  envio();                                      // Enviar dato al maestro
  digitalWrite(activePin3,LOW);                 // Activar sensor infrarrojo
  for(a=0;a<10000;a++){                         // Empieza a sensar  el infrarrojo
      respuesta=digitalRead(infraPin3);}        // Guardar el valor en la variable respuesta (0 o 1)
      digitalWrite(activePin3,HIGH);            // Apagar sensor infrarrojo
  if(respuesta==LOW){                           // Si es 0 detecta proyector
      digitalWrite(ledPin3,LOW);                // Encender luz verde
      datostx=5;                                // Cargar dato que identifica que hay proyector
      envio();}                                 // Enviar dato que si hay proyector al maestro
  else{                                         // Si es 1 no hay proyector
      digitalWrite(ledPin3,HIGH);               // Encender luz roja
      datostx=4;                                // Cargar dato que identifica que no hay proyector
      envio();}                                 // Enviar dato que no hay proyector al maestro
}
////////////////////////////////////////////////////////////////////////////// FUNCIÓN DE PETICION DEL
LOCKER 6
void locker6(){
  digitalWrite(cerraduraPin4,LOW);              // Abrir la cerradura
  while(digitalRead(magnetoPin4)==LOW){}        // Esperar hasta que se separe la puerta
  datostx=120;                                  // Cargar dato que identifica que está abierta la puerta
  envio();                                      // Enviar dato al maestro
  while(digitalRead(magnetoPin4)==HIGH){}       // Espera hasta que se cierre la puerta
  digitalWrite(cerraduraPin4,HIGH);             // Cerrar la cerradura
  datostx=121;                                  // Cargar dato que identifica que está cerrada la puerta
  envio();                                      // Enviar dato al maestro
  digitalWrite(activePin4,LOW);                 // Activar sensor infrarrojo
  for(a=0;a<10000;a++){                         // Empieza a sensar  el infrarrojo
      respuesta=digitalRead(infraPin4);}        // Guardar el valor en la variable respuesta (0 o 1)
      digitalWrite(activePin4,HIGH);            // Apagar sensor infrarrojo
  if(respuesta==LOW){                           // Si es 0 detecta proyector
      digitalWrite(ledPin4,LOW);                // Encender luz verde
      datostx=7;                                // Cargar dato que identifica que hay proyector
      envio();}                                 // Enviar dato que si hay proyector al maestro
  else{                                         // Si es 1 no hay proyector
      digitalWrite(ledPin4,HIGH);               // Encender luz roja
      datostx=6;                                // Cargar dato que identifica que no hay proyector
      envio();}                                 // Enviar dato que no hay proyector al maestro
}
////////////////////////////////////////////////////////////////////////////// FUNCIÓN DE PETICION DEL
LOCKER 9
void locker9(){
  digitalWrite(cerraduraPin5,LOW);              // Abrir la cerradura
  while(digitalRead(magnetoPin5)==LOW){}        // Esperar hasta que se separe la puerta
  datostx=120;                                  // Cargar dato que identifica que está abierta la puerta
  envio();                                      // Enviar dato al maestro
  while(digitalRead(magnetoPin5)==HIGH){}       // Espera hasta que se cierre la puerta
  digitalWrite(cerraduraPin5,HIGH);             // Cerrar la cerradura
  datostx=121;                                  // Cargar dato que identifica que está cerrada la puerta
  envio();                                      // Enviar dato al maestro
  digitalWrite(activePin5,LOW);                 // Activar sensor infrarrojo
  for(a=0;a<10000;a++){                         // Empieza a sensar  el infrarrojo
      respuesta=digitalRead(infraPin5);}        // Guardar el valor en la variable respuesta (0 o 1)
      digitalWrite(activePin5,HIGH);            // Apagar sensor infrarrojo
  if(respuesta==LOW){                           // Si es 0 detecta proyector
      digitalWrite(ledPin5,LOW);                // Encender luz verde
      datostx=9;                                // Cargar dato que identifica que hay proyector
      envio();}                                 // Enviar dato que si hay proyector al maestro
  else{                                         // Si es 1 no hay proyector
      digitalWrite(ledPin5,HIGH);               // Encender luz roja
      datostx=8;                                // Cargar dato que identifica que no hay proyector
      envio();}                                 // Enviar dato que no hay proyector al maestro
}
```

```
//////////////////////////////////////////////////////////////////////////////// FUNCIÓN DE PETICION DEL LOCKER
10
void locker10(){
  digitalWrite(cerraduraPin6,LOW);              // Abrir la cerradura
  while(digitalRead(magnetoPin6)==LOW){}        // Esperar hasta que se separe la puerta
  datostx=120;                                  // Cargar dato que identifica que está abierta la puerta
  envio();                                      // Enviar dato al maestro
  while(digitalRead(magnetoPin6)==HIGH){}       // Espera hasta que se cierre la puerta
  digitalWrite(cerraduraPin6,HIGH);             // Cerrar la cerradura
  datostx=121;                                  // Cargar dato que identifica que está cerrada la puerta
  envio();                                      // Enviar dato al maestro
//  digitalWrite(activePin6,LOW);               // Activar sensor infrarrojo (siempre activo)
  for(a=0;a<10000;a++){                         // Empieza a sensar  el infrarrojo
      respuesta=digitalRead(infraPin6);}        // Guardar el valor en la variable respuesta (0 o 1)
//      digitalWrite(activePin6,HIGH);
  if(respuesta==LOW){                           // Si es 0 detecta proyector
      digitalWrite(ledPin6,LOW);                // Encender luz verde
      datostx=11;                               // Cargar dato que identifica que hay proyector
      envio();}                                 // Enviar dato que si hay proyector al maestro
  else{                                         // Si es 1 no hay proyector
      digitalWrite(ledPin6,HIGH);               // Encender luz roja
      datostx=10;                               // Cargar dato que identifica que no hay proyector
      envio();}                                 // Enviar dato que no hay proyector al maestro
}
//////////////////////////////////////////////////////////////////////////////// FUNCIÓN DE PETICION DEL LOCKER
11
void locker13(){
  digitalWrite(cerraduraPin7,LOW);              // Abrir la cerradura
  while(digitalRead(magnetoPin7)==LOW){}        // Esperar hasta que se separe la puerta
  datostx=120;                                  // Cargar dato que identifica que está abierta la puerta
  envio();                                      // Enviar dato al maestro
  while(digitalRead(magnetoPin7)==HIGH){}       // Espera hasta que se cierre la puerta
  digitalWrite(cerraduraPin7,HIGH);             // Cerrar la cerradura
  datostx=121;                                  // Cargar dato que identifica que está cerrada la puerta
  envio();                                      // Enviar dato al maestro
  digitalWrite(activePin7,LOW);                 // Activar sensor infrarrojo
  for(a=0;a<10000;a++){                         // Empieza a sensar  el infrarrojo
      respuesta=digitalRead(infraPin7);}        // Guardar el valor en la variable respuesta (0 o 1)
      digitalWrite(activePin7,HIGH);            // Apagar sensor infrarrojo
  if(respuesta==LOW){                           // Si es 0 detecta proyector
      digitalWrite(ledPin7,LOW);                // Encender luz verde
      datostx=13;                               // Cargar dato que identifica que hay proyector
      envio();}                                 // Enviar dato que si hay proyector al maestro
  else{                                         // Si es 1 no hay proyector
      digitalWrite(ledPin7,HIGH);               // Encender luz roja
      datostx=12;                               // Cargar dato que identifica que no hay proyector
      envio();}                                 // Enviar dato que no hay proyector al maestro
}
//////////////////////////////////////////////////////////////////////////////// FUNCIÓN DE PETICION DEL LOCKER
14
void locker14(){
  digitalWrite(cerraduraPin8,LOW);              // Abrir la cerradura
  while(digitalRead(magnetoPin8)==LOW){}        // Esperar hasta que se separe la puerta
  datostx=120;                                  // Cargar dato que identifica que está abierta la puerta
  envio();                                      // Enviar dato al maestro
  while(digitalRead(magnetoPin8)==HIGH){}       // Espera hasta que se cierre la puerta
  digitalWrite(cerraduraPin8,HIGH);             // Cerrar la cerradura
  datostx=121;                                  // Cargar dato que identifica que está cerrada la puerta
  envio();                                      // Enviar dato al maestro
// digitalWrite(activePin8,LOW);                // Activar sensor infrarrojo(siempre activo)
  for(a=0;a<10000;a++){                         // Empieza a sensar  el infrarrojo
      respuesta=digitalRead(infraPin8);}        // Guardar el valor en la variable respuesta (0 o 1)
// digitalWrite(activePin8,HIGH);
  if(respuesta==LOW){                           // Si es 0 detecta proyector
      digitalWrite(ledPin8,LOW);                // Encender luz verde
      datostx=15;                               // Cargar dato que identifica que hay proyector
      envio();}                                 // Enviar dato que si hay proyector al maestro
  else{                                         // Si es 1 no hay proyector
      digitalWrite(ledPin8,HIGH);               // Encender luz roja
      datostx=14;                               // Cargar dato que identifica que no hay proyector
      envio();}                                 // Enviar dato que no hay proyector al maestro
}
//////////////////////////////////////////////////////////////////////////////// FUNCIÓN DE PETICION DEL LOCKER
17
void locker17(){
  digitalWrite(cerraduraPin9,LOW);              // Abrir la cerradura
  while(digitalRead(magnetoPin9)==LOW){}        // Esperar hasta que se separe la puerta
  datostx=120;                                  // Cargar dato que identifica que está abierta la puerta
  envio();                                      // Enviar dato al maestro
  while(digitalRead(magnetoPin9)==HIGH){}       // Espera hasta que se cierre la puerta
  digitalWrite(cerraduraPin9,HIGH);             // Cerrar la cerradura
  datostx=121;                                  // Cargar dato que identifica que está cerrada la puerta
```

```
  envio();                                    // Enviar dato al maestro
  digitalWrite(activePin9,LOW);               // Activar sensor infrarrojo
  for(a=0;a<10000;a++){                       // Empieza a sensar  el infrarrojo
      respuesta=digitalRead(infraPin9);}      // Guardar el valor en la variable respuesta (0 o 1)
      digitalWrite(activePin9,HIGH);          // Apagar sensor infrarrojo
  if(respuesta==LOW){                         // Si es 0 detecta proyector
      digitalWrite(ledPin9,LOW);              // Encender luz verde
      datostx=17;                             // Cargar dato que identifica que hay proyector
      envio();}                               // Enviar dato que si hay proyector al maestro
  else{                                       // Si es 1 no hay proyector
      digitalWrite(ledPin9,HIGH);             // Encender luz roja
      datostx=16;                             // Cargar dato que identifica que no hay proyector
      envio();}                               // Enviar dato que no hay proyector al maestro
}
//////////////////////////////////////////////////////////////////////////////// FUNCIÓN DE PETICION DEL LOCKER
18
void locker18(){
 digitalWrite(cerraduraPin10,LOW);            // Abrir la cerradura
  while(digitalRead(magnetoPin10)==LOW){}     // Esperar hasta que se separe la puerta
  datostx=120;                                // Cargar dato que identifica que está abierta la puerta
  envio();                                    // Enviar dato al maestro
  while(digitalRead(magnetoPin10)==HIGH){}    // Espera hasta que se cierre la puerta
  digitalWrite(cerraduraPin10,HIGH);          // Cerrar la cerradura
  datostx=121;                                // Cargar dato que identifica que está cerrada la puerta
  envio();                                    // Enviar dato al maestro
  digitalWrite(activePin10,LOW);              // Activar sensor infrarrojo
  for(a=0;a<10000;a++){                       // Empieza a sensar  el infrarrojo
      respuesta=digitalRead(infraPin10);}     // Guardar el valor en la variable respuesta (0 o 1)
      digitalWrite(activePin10,HIGH);         // Apagar sensor infrarrojo
  if(respuesta==LOW){                         // Si es 0 detecta proyector
      digitalWrite(ledPin10,LOW);             // Encender luz verde
      datostx=19;                             // Cargar dato que identifica que hay proyector
      envio();}                               // Enviar dato que si hay proyector al maestro
  else{                                       // Si es 1 no hay proyector
      digitalWrite(ledPin10,HIGH);            // Encender luz roja
      datostx=18;                             // Cargar dato que identifica que no hay proyector
      envio();}                               // Enviar dato que no hay proyector al maestro
}
//////////////////////////////////////////////////////////////////////////////FUNCION CUANDO SE RECIBE UN DATO POR
I2C
//Función que se ejecuta siempre que se reciben datos del master, siempre que en el master se ejecute la
//sentencia endTransmission recibirá toda la información que se transmitirá a través de la sentencia
Wire.write
void receiveEvent() {
  digitalWrite(52,HIGH);              // Encender led indicador de que se recibió un dato del maestro
  if (Wire.available() == 1){         // Si hay un byte disponible
   datorx = Wire.read();              // Leer y cargar la variable
    }
  delay(500);                         // Pausa de medio segundo
  digitalWrite(52,LOW);               // Apagar led indicador de que se recibió un dato del maestro
}
////////////////////////////////////////////////////////// FUNCIONES DE ENVIO DE DATOS A LOS DEMÁS
DISPOSITIVOS
void envio(){                        // FUNCION PARA ENVIAR DATOS AL MAESTRO
    Wire.beginTransmission(0);       // Comenzar la transmisión al dispositivo 0
    Wire.write(datostx);             // Enviar un byte
    Wire.endTransmission();          // Parar la transmisión
    datorx=0;                        // Encerar variables
    datostx=0;
    delay(1000);                     // Esperar 1 segundo
}

void envio2(){                       // FUNCION PARA ENVIAR DATOS AL ESCLAVO 2
    Wire.beginTransmission(2);       // Comenzar la transmisión al dispositivo 2
    Wire.write(datostx);             // Enviar un byte
    Wire.endTransmission();          // Parar la transmisión
    datostx=0;                       // Encerar variables
    datorx=0;
    delay(1000);                     // Esperar 1 segundo

}
```

## ANEXO B-3: Código complete Arduino esclavo 2

```
/*
                                    ESCUELA POLITÉCNICA NACIONAL - ESFOT
                                          PROYECTO DE TITULACION
                                            Control de Lockers
                                                Esclavo 2
*/
////////////////////////////////////////////////////////////////////////////////////////////////// LIBRERIAS
#include <Wire.h>
////////////////////////////////////////////////////////////////////////////////////////////////// VARIABLES
byte datorx = 0;        //Variable para recibir datos por i2c
byte datostx=0;         //Variable de enviar datos por i2c
byte a=0;               //Variable a para establecer un contador (tiempo para sensar con infrarrojo)
byte respuesta=0;       //Variable respuesta para leer datos de sensor infrarrojo
//Variables para manejar los puertos I/O con nombres propios para cada locker:
//////////////////////////////////////////////////////////////Locker 3
byte magnetopin1=5;     //Lectura del sensor magnético
byte infrapin1=51;      //Lectura del sensor infrarrojo
byte ledpin1=25;        //Activación de luces
byte cerradurapin1=34; //Activación de cerradura
byte activepin1=30;     //Activación del infrarrojo
//////////////////////////////////////////////////////////////Locker 4
byte magnetopin2=4;     //Lectura del sensor magnético
byte infrapin2=53;      //Lectura del sensor infrarrojo
byte ledpin2=26;        //Activación de luces
byte cerradurapin2=33; //Activación de cerradura
byte activepin2=31;     //Activación del infrarrojo
//////////////////////////////////////////////////////////////Locker 7
byte magnetopin3=7;     //Lectura del sensor magnético
byte infrapin3=18;      //Lectura del sensor infrarrojo
byte ledpin3=23;        //Activación de luces
byte cerradurapin3=36; //Activación de cerradura
byte activepin3=28;     //Activación del infrarrojo
//////////////////////////////////////////////////////////////Locker 8
byte magnetopin4=6;     //Lectura del sensor magnético
byte infrapin4=19;      //Lectura del sensor infrarrojo
byte ledpin4=24;        //Activación de luces
byte cerradurapin4=35; //Activación de cerradura
byte activepin4=29;     //Activación del infrarrojo
//////////////////////////////////////////////////////////////Locker 11
byte magnetopin5=9;     //Lectura del sensor magnético
byte infrapin5=16;      //Lectura del sensor infrarrojo
byte ledpin5=42;        //Activación de luces
byte cerradurapin5=47; //Activación de cerradura
byte activepin5=49;     //Activación del infrarrojo
//////////////////////////////////////////////////////////////Locker 12
byte magnetopin6=8;     //Lectura del sensor magnético
byte infrapin6=17;      //Lectura del sensor infrarrojo
byte ledpin6=22;        //Activación de luces
byte cerradurapin6=37; //Activación de cerradura
byte activepin6=27;     //Activación del infrarrojo
//////////////////////////////////////////////////////////////Locker 15
byte magnetopin7=11;    //Lectura del sensor magnético
byte infrapin7=14;      //Lectura del sensor infrarrojo
byte ledpin7=40;        //Activación de luces
byte cerradurapin7=45; //Activación de cerradura
//byte activepin7=51;  //Activación del infrarrojo
//////////////////////////////////////////////////////////////Locker 16
byte magnetopin8=10;    //Lectura del sensor magnético
byte infrapin8=15;      //Lectura del sensor infrarrojo
byte ledpin8=41;        //Activación de luces
byte cerradurapin8=46; //Activación de cerradura
byte activepin8=50;     //Activación del infrarrojo
//////////////////////////////////////////////////////////////Locker 19
byte magnetopin9=13;    //Lectura del sensor magnético
byte infrapin9=3;       //Lectura del sensor infrarrojo
byte ledpin9=38;        //Activación de luces
byte cerradurapin9=43; //Activación de cerradura
//byte activepin9=53;  //Activación del infrarrojo
//////////////////////////////////////////////////////////////Locker 20
byte magnetopin10=12;   //Lectura del sensor magnético
byte infrapin10=2;      //Lectura del sensor infrarrojo
byte ledpin10=39;       //Activación de luces
byte cerradurapin10=44; //Activación de cerradura
byte activepin10=48;    //Activación del infrarrojo
////////////////////////////////////////////////////////////////////DECLARACION DE PUERTOS DE ENTRADA Y
SALIDA
```

```
void setup() {
////////////////////////////////////////////////////////////Locker 3
pinMode (ledPin1,OUTPUT);
pinMode (cerraduraPin1,OUTPUT);
pinMode (activePin1,OUTPUT);
digitalWrite (ledPin1,HIGH);
digitalWrite (cerraduraPin1,HIGH);
digitalWrite (activePin1,HIGH);
pinMode (infraPin1,INPUT);
pinMode (magnetoPin1,INPUT);
digitalWrite (infraPin1,HIGH);
digitalWrite (magnetoPin1,HIGH);
////////////////////////////////////////////////////////////Locker 4
pinMode (ledPin2,OUTPUT);
pinMode (cerraduraPin2,OUTPUT);
pinMode (activePin2,OUTPUT);
digitalWrite (ledPin2,HIGH);
digitalWrite (cerraduraPin2,HIGH);
digitalWrite (activePin2,HIGH);
pinMode (infraPin2,INPUT);
pinMode (magnetoPin2,INPUT);
digitalWrite (infraPin2,HIGH);
digitalWrite (magnetoPin2,HIGH);
////////////////////////////////////////////////////////////Locker 7
pinMode (ledPin3,OUTPUT);
pinMode (cerraduraPin3,OUTPUT);
pinMode (activePin3,OUTPUT);
digitalWrite (ledPin3,HIGH);
digitalWrite (cerraduraPin3,HIGH);
digitalWrite (activePin3,HIGH);
pinMode (infraPin3,INPUT);
pinMode (magnetoPin3,INPUT);
digitalWrite (infraPin3,HIGH);
digitalWrite (magnetoPin3,HIGH);
////////////////////////////////////////////////////////////Locker 8
pinMode (ledPin4,OUTPUT);
pinMode (cerraduraPin4,OUTPUT);
pinMode (activePin4,OUTPUT);
digitalWrite (ledPin4,HIGH);
digitalWrite (cerraduraPin4,HIGH);
digitalWrite (activePin4,HIGH);
pinMode (infraPin4,INPUT);
pinMode (magnetoPin4,INPUT);
digitalWrite (infraPin4,HIGH);
digitalWrite (magnetoPin4,HIGH);
////////////////////////////////////////////////////////////Locker 11
pinMode (ledPin5,OUTPUT);
pinMode (cerraduraPin5,OUTPUT);
pinMode (activePin5,OUTPUT);
digitalWrite (ledPin5,HIGH);
digitalWrite (cerraduraPin5,HIGH);
digitalWrite (activePin5,HIGH);
pinMode (infraPin5,INPUT);
pinMode (magnetoPin5,INPUT);
digitalWrite (infraPin5,HIGH);
digitalWrite (magnetoPin5,HIGH);
////////////////////////////////////////////////////////////Locker 12
pinMode (ledPin6,OUTPUT);
pinMode (cerraduraPin6,OUTPUT);
//pinMode (activePin6,OUTPUT);
digitalWrite (ledPin6,HIGH);
digitalWrite (cerraduraPin6,HIGH);
pinMode (infraPin6,INPUT);
pinMode (magnetoPin6,INPUT);
digitalWrite (infraPin6,HIGH);
digitalWrite (magnetoPin6,HIGH);
////////////////////////////////////////////////////////////Locker 15
pinMode (ledPin7,OUTPUT);
pinMode (cerraduraPin7,OUTPUT);
//pinMode (activePin7,OUTPUT);
digitalWrite (ledPin7,HIGH);
digitalWrite (cerraduraPin7,HIGH);
digitalWrite (activePin7,HIGH);
pinMode (infraPin7,INPUT);
pinMode (magnetoPin7,INPUT);
digitalWrite (infraPin7,HIGH);
digitalWrite (magnetoPin7,HIGH);
////////////////////////////////////////////////////////////Locker 16
pinMode (ledPin8,OUTPUT);
pinMode (cerraduraPin8,OUTPUT);
pinMode (activePin8,OUTPUT);
digitalWrite (ledPin8,HIGH);
```

```
digitalWrite (cerraduraPin8,HIGH);
pinMode (infraPin8,INPUT);
pinMode (magnetoPin8,INPUT);
digitalWrite (infraPin8,HIGH);
digitalWrite (magnetoPin8,HIGH);
////////////////////////////////////////////////////////Locker 19
pinMode (ledPin9,OUTPUT);
pinMode (cerraduraPin9,OUTPUT);
//pinMode (activePin9,OUTPUT);
digitalWrite (ledPin9,HIGH);
digitalWrite (cerraduraPin9,HIGH);
digitalWrite (activePin9,HIGH);
pinMode (infraPin9,INPUT);
pinMode (magnetoPin9,INPUT);
digitalWrite (infraPin9,HIGH);
digitalWrite (magnetoPin9,HIGH);
////////////////////////////////////////////////////////Locker 20
pinMode (ledPin10,OUTPUT);
pinMode (cerraduraPin10,OUTPUT);
pinMode (activePin10,OUTPUT);
digitalWrite (ledPin10,HIGH);
digitalWrite (cerraduraPin10,HIGH);
digitalWrite (activePin10,HIGH);
pinMode (infraPin10,INPUT);
pinMode (magnetoPin10,INPUT);
digitalWrite (infraPin10,HIGH);
digitalWrite (magnetoPin10,HIGH);

pinMode (52,OUTPUT);          //Led indicador cuando recibe un dato por i2c
////////////////////////////////////////////////////////// UNIR ESTE DISPOSITIVO AL BUS I2C CON DIRECCION
2
Wire.begin(2);
Wire.onReceive(receiveEvent); // Registrar el evento al recibir datos por i2c
Serial.begin(9600);           // Iniciamos el monitor serie para monitorear la comunicación
}
/////////////////////////////////////IDENTIFICACION DE LOCKER MEDIANTE EL DATO RECIBIDO POR PARTE DEL
MAESTRO
void loop(){                  // Funcion loop
  switch (datorx){            // Comparar a que caso pertenece el dato recibido del maestro
  case 3:                     // Número único que identifica el locker 3
    locker3();                // Ir a la función de petición del locker 3
    break;
  case 4:                     // Número único que identifica el locker 4
    locker4();                // Ir a la función de petición del locker 4
    break;
  case 7:                     // Número único que identifica el locker 7
    locker7();                // Ir a la función de petición del locker 7
    break;
  case 8:                     // Número único que identifica el locker 8
    locker8();                // Ir a la función de petición del locker 8
    break;
  case 11:                    // Número único que identifica el locker 11
    locker11();               // Ir a la función de petición del locker 11
    break;
  case 12:                    // Número único que identifica el locker 12
    locker12();               // Ir a la función de petición del locker 12
    break;
  case 15:                    // Número único que identifica el locker 15
    locker15();               // Ir a la función de petición del locker 15
    break;
  case 16:                    // Número único que identifica el locker 16
    locker16();               // Ir a la función de petición del locker 16
    break;
  case 19:                    // Número único que identifica el locker 19
    locker19();               // Ir a la función de petición del locker 19
    break;
  case 20:                    // Número único que identifica el locker 20
    locker20();               // Ir a la función de petición del locker 20
    break;
  case 200:                   // Dato que se recibe para sensar todos los lockers.
    inicio();                 // Ir a la función inicio para sensar todos los puestos
    break;
  }
}
////////////////////////////////////////////////////////////////////////// FUNCION PARA SENSAR TODOS LOS
LOCKERS
void inicio(){
////////////////////////////////////////////////////////////////////////////////////////////////Locker 3
  digitalWrite(activePin1,LOW);            // Activar sensor infrarrojo
  for(a=0;a<10000;a++){                    // Empieza a sensar  el infrarrojo
      respuesta=digitalRead(infraPin1);}   // Guardar el valor en la variable respuesta (0 o 1)
  digitalWrite(activePin1,HIGH);           // Apagar sensor infrarrojo
```

```
  if(respuesta==LOW){                       // Si es 0 detecta proyector
      digitalWrite(ledPin1,LOW);            // Encender luz verde
      datostx=21;                           // Cargar dato que identifica que hay proyector
      envio();}                             // Enviar dato que si hay proyector al maestro
  else{                                     // Si es 1 no hay proyector
      digitalWrite(ledPin1,HIGH);           // Encender luz roja
      datostx=20;                           // Cargar dato que identifica que no hay proyector
      envio();}                              // Enviar dato que no hay proyector al maestro
////////////////////////////////////////////////////////////////////////////////////////////Locker 4
digitalWrite(activePin2,LOW);               // Activar sensor infrarrojo
  for(a=0;a<10000;a++){                     // Empieza a sensar  el infrarrojo
      respuesta=digitalRead(infraPin2);}    // Guardar el valor en la variable respuesta (0 o 1)
digitalWrite(activePin2,HIGH);              // Apagar sensor infrarrojo
  if(respuesta==LOW){                       // Si es 0 detecta proyector
      digitalWrite(ledPin2,LOW);            // Encender luz verde
      datostx=23;                           // Cargar dato que identifica que hay proyector
      envio();}                             // Enviar dato que si hay proyector al maestro
  else{                                     // Si es 1 no hay proyector
      digitalWrite(ledPin2,HIGH);           // Encender luz roja
      datostx=22;                           // Cargar dato que identifica que no hay proyector
      envio();}                             // Enviar dato que no hay proyector al maestro
////////////////////////////////////////////////////////////////////////////////////////////Locker 7
digitalWrite(activePin3,LOW);               // Activar sensor infrarrojo
  for(a=0;a<10000;a++){                     // Empieza a sensar  el infrarrojo
      respuesta=digitalRead(infraPin3);}    // Guardar el valor en la variable respuesta (0 o 1)
digitalWrite(activePin3,HIGH);              // Apagar sensor infrarrojo
  if(respuesta==LOW){                       // Si es 0 detecta proyector
      digitalWrite(ledPin3,LOW);            // Encender luz verde
      datostx=25;                           // Cargar dato que identifica que hay proyector
      envio();}                             // Enviar dato que si hay proyector al maestro
  else{                                     // Si es 1 no hay proyector
      digitalWrite(ledPin3,HIGH);           // Encender luz roja
      datostx=24;                           // Cargar dato que identifica que no hay proyector
      envio();}                             // Enviar dato que no hay proyector al maestro
////////////////////////////////////////////////////////////////////////////////////////////Locker 8
digitalWrite(activePin4,LOW);               // Activar sensor infrarrojo
  for(a=0;a<10000;a++){                     // Empieza a sensar  el infrarrojo
      respuesta=digitalRead(infraPin4);}    // Guardar el valor en la variable respuesta (0 o 1)
digitalWrite(activePin4,HIGH);              // Apagar sensor infrarrojo
  if(respuesta==LOW){                       // Si es 0 detecta proyector
      digitalWrite(ledPin4,LOW);            // Encender luz verde
      datostx=27;                           // Cargar dato que identifica que hay proyector
      envio();}                             // Enviar dato que si hay proyector al maestro
  else{                                     // Si es 1 no hay proyector
      digitalWrite(ledPin4,HIGH);           // Encender luz roja
      datostx=26;                           // Cargar dato que identifica que no hay proyector
      envio();}                             // Enviar dato que no hay proyector al maestro
////////////////////////////////////////////////////////////////////////////////////////////Locker 11
digitalWrite(activePin5,LOW);               // Activar sensor infrarrojo
  for(a=0;a<10000;a++){                     // Empieza a sensar  el infrarrojo
      respuesta=digitalRead(infraPin5);}    // Guardar el valor en la variable respuesta (0 o 1)
digitalWrite(activePin5,HIGH);              // Apagar sensor infrarrojo
  if(respuesta==LOW){                       // Si es 0 detecta proyector
      digitalWrite(ledPin5,LOW);            // Encender luz verde
      datostx=29;                           // Cargar dato que identifica que hay proyector
      envio();}                             // Enviar dato que si hay proyector al maestro
  else{                                     // Si es 1 no hay proyector
      digitalWrite(ledPin5,HIGH);           // Encender luz roja
      datostx=28;                           // Cargar dato que identifica que no hay proyector
      envio();}                             // Enviar dato que no hay proyector al maestro
////////////////////////////////////////////////////////////////////////////////////////////Locker 12
digitalWrite(activePin6,LOW);               // Activar sensor infrarrojo
  for(a=0;a<10000;a++){                     // Empieza a sensar  el infrarrojo
      respuesta=digitalRead(infraPin6);}    // Guardar el valor en la variable respuesta (0 o 1)
digitalWrite(activePin6,HIGH);              // Apagar sensor infrarrojo
  if(respuesta==LOW){                       // Si es 0 detecta proyector
      digitalWrite(ledPin6,LOW);            // Encender luz verde
      datostx=31;                           // Cargar dato que identifica que hay proyector
      envio();}                             // Enviar dato que si hay proyector al maestro
  else{                                     // Si es 1 no hay proyector
      digitalWrite(ledPin6,HIGH);           // Encender luz roja
      datostx=30;                           // Cargar dato que identifica que no hay proyector
      envio();}                             // Enviar dato que no hay proyector al maestro
////////////////////////////////////////////////////////////////////////////////////////////Locker 15
//digitalWrite(activePin7,LOW);             // Activar sensor infrarrojo (siempre activo)
  for(a=0;a<10000;a++){                     // Empieza a sensar  el infrarrojo
      respuesta=digitalRead(infraPin7);}    // Guardar el valor en la variable respuesta (0 o 1)
//digitalWrite(activePin7,HIGH);
  if(respuesta==LOW){                       // Si es 0 detecta proyector
      digitalWrite(ledPin7,LOW);            // Encender luz verde
      datostx=33;                           // Cargar dato que identifica que hay proyector
      envio();}                             // Enviar dato que si hay proyector al maestro
```

```
  else{                                    // Si es 1 no hay proyector
      digitalWrite(ledPin7,HIGH);          // Encender luz roja
      datostx=32;                          // Cargar dato que identifica que no hay proyector
      envio();}                            // Enviar dato que no hay proyector al maestro
////////////////////////////////////////////////////////////////////////////////////////////Locker 16
digitalWrite(activePin8,LOW);              // Activar sensor infrarrojo
  for(a=0;a<10000;a++){                      // Empieza a sensar  el infrarrojo
      respuesta=digitalRead(infraPin8);}     // Guardar el valor en la variable respuesta (0 o 1)
digitalWrite(activePin8,HIGH);             // Apagar sensor infrarrojo
  if(respuesta==LOW){                      // Si es 0 detecta proyector
      digitalWrite(ledPin8,LOW);           // Encender luz verde
      datostx=35;                          // Cargar dato que identifica que hay proyector
      envio();}                            // Enviar dato que si hay proyector al maestro
  else{                                    // Si es 1 no hay proyector
      digitalWrite(ledPin8,HIGH);          // Encender luz roja
      datostx=34;                          // Cargar dato que identifica que no hay proyector
      envio();}                            // Enviar dato que no hay proyector al maestro
////////////////////////////////////////////////////////////////////////////////////////////Locker 19
//digitalWrite(activePin9,LOW);            // Activar sensor infrarrojo (siempre activo)
  for(a=0;a<10000;a++){                      // Empieza a sensar  el infrarrojo
      respuesta=digitalRead(infraPin9);}     // Guardar el valor en la variable respuesta (0 o 1)
//digitalWrite(activePin9,HIGH);
  if(respuesta==LOW){                      // Si es 0 detecta proyector
      digitalWrite(ledPin9,LOW);           // Encender luz verde
      datostx=37;                          // Cargar dato que identifica que hay proyector
      envio();}                            // Enviar dato que si hay proyector al maestro
  else{                                    // Si es 1 no hay proyector
      digitalWrite(ledPin9,HIGH);          // Encender luz roja
      datostx=36;                          // Cargar dato que identifica que no hay proyector
      envio();}                            // Enviar dato que no hay proyector al maestro
////////////////////////////////////////////////////////////////////////////////////////////Locker 20
digitalWrite(activePin10,LOW);             // Activar sensor infrarrojo
  for(a=0;a<10000;a++){                      // Empieza a sensar  el infrarrojo
      respuesta=digitalRead(infraPin10);}    // Guardar el valor en la variable respuesta (0 o 1)
digitalWrite(activePin10,HIGH);            // Apagar sensor infrarrojo
  if(respuesta==LOW){                      // Si es 0 detecta proyector
      digitalWrite(ledPin10,LOW);          // Encender luz verde
      datostx=39;                          // Cargar dato que identifica que hay proyector
      envio();}                            // Enviar dato que si hay proyector al maestro
  else{                                    // Si es 1 no hay proyector
      digitalWrite(ledPin10,HIGH);         // Encender luz roja
      datostx=38;                          // Cargar dato que identifica que no hay proyector
      envio();}                            // Enviar dato que no hay proyector al maestro
////////////////////////////////////////////////////////////////////////////////////////////// MAESTRO
datostx=201                  // Cargar  dato para enviar al maestro indicando que se terminó de sensar
envio();                     // Llamar a la función de envio
}
/////////////////////////////////////////////////////////////////////////////////// FUNCIÓN DE PETICION DEL
LOCKER 3
void locker3(){
  digitalWrite(cerraduraPin1,LOW);             // Abrir la cerradura
  while(digitalRead(magnetoPin1)==LOW){}       // Esperar hasta que se separe la puerta
  datostx=120;                                 // Cargar dato que identifica que está abierta la puerta
  envio();                                     // Enviar dato al maestro
  while(digitalRead(magnetoPin1)==HIGH){}      // Espera hasta que se cierre la puerta
  digitalWrite(cerraduraPin1,HIGH);            // Cerrar la cerradura
  datostx=121;                                 // Cargar dato que identifica que está cerrada la puerta
  envio();                                     // Enviar dato al maestro
  digitalWrite(activePin1,LOW);                // Activar sensor infrarrojo
  for(a=0;a<10000;a++){                        // Empieza a sensar  el infrarrojo
      respuesta=digitalRead(infraPin1);}       // Guardar el valor en la variable respuesta (0 o 1)
  digitalWrite(activePin1,HIGH);               // Apagar sensor infrarrojo
  if(respuesta==LOW){                          // Si es 0 detecta proyector
      digitalWrite(ledPin1,LOW);               // Encender luz verde
      datostx=21;                               // Cargar dato que identifica que hay proyector
      envio();}                                // Enviar dato que si hay proyector al maestro
  else{                                        // Si es 1 no hay proyector
      digitalWrite(ledPin1,HIGH);              // Encender luz roja
      datostx=20;                               // Cargar dato que identifica que no hay proyector
      envio();}                                // Enviar dato que no hay proyector al maestro
}
/////////////////////////////////////////////////////////////////////////////////// FUNCIÓN DE PETICION DEL
LOCKER 4
void locker4(){
  digitalWrite(cerraduraPin2,LOW);             // Abrir la cerradura
  while(digitalRead(magnetoPin2)==LOW){}       // Esperar hasta que se separe la puerta
  datostx=120;                                 // Cargar dato que identifica que está abierta la puerta
  envio();                                     // Enviar dato al maestro
  while(digitalRead(magnetoPin2)==HIGH){}      // Espera hasta que se cierre la puerta
  digitalWrite(cerraduraPin2,HIGH);            // Cerrar la cerradura
  datostx=121;                                 // Cargar dato que identifica que está cerrada la puerta
  envio();                                     // Enviar dato al maestro
```

```
  digitalWrite(activePin2,LOW);                 // Activar sensor infrarrojo
  for(a=0;a<10000;a++){                         // Empieza a sensar  el infrarrojo
      respuesta=digitalRead(infraPin2);}        // Guardar el valor en la variable respuesta (0 o 1)
      digitalWrite(activePin2,HIGH);            // Apagar sensor infrarrojo
  if(respuesta==LOW){                           // Si es 0 detecta proyector
      digitalWrite(ledPin2,LOW);                // Encender luz verde
      datostx=23;                                // Cargar dato que identifica que hay proyector
      envio();}                                 // Enviar dato que si hay proyector al maestro
  else{                                         // Si es 1 no hay proyector
      digitalWrite(ledPin2,HIGH);               // Encender luz roja
      datostx=22;                                // Cargar dato que identifica que no hay proyector
      envio();}                                 // Enviar dato que no hay proyector al maestro
}
////////////////////////////////////////////////////////////////////////////// FUNCIÓN DE PETICION DEL
LOCKER 7
void locker7(){
 digitalWrite(cerraduraPin3,LOW);               // Abrir la cerradura
  while(digitalRead(magnetoPin3)==LOW){}        // Esperar hasta que se separe la puerta
  datostx=120;                                  // Cargar dato que identifica que está abierta la puerta
  envio();                                      // Enviar dato al maestro
  while(digitalRead(magnetoPin3)==HIGH){}       // Espera hasta que se cierre la puerta
  digitalWrite(cerraduraPin3,HIGH);             // Cerrar la cerradura
  datostx=121;                                  // Cargar dato que identifica que está cerrada la puerta
  envio();                                      // Enviar dato al maestro
  digitalWrite(activePin3,LOW);                 // Activar sensor infrarrojo
  for(a=0;a<10000;a++){                         // Empieza a sensar  el infrarrojo
      respuesta=digitalRead(infraPin3);}        // Guardar el valor en la variable respuesta (0 o 1)
      digitalWrite(activePin3,HIGH);            // Apagar sensor infrarrojo
  if(respuesta==LOW){                           // Si es 0 detecta proyector
      digitalWrite(ledPin3,LOW);                // Encender luz verde
      datostx=25;                                // Cargar dato que identifica que hay proyector
      envio();}                                 // Enviar dato que si hay proyector al maestro
  else{                                         // Si es 1 no hay proyector
      digitalWrite(ledPin3,HIGH);               // Encender luz roja
      datostx=24;                                // Cargar dato que identifica que no hay proyector
      envio();}                                 // Enviar dato que no hay proyector al maestro
}
////////////////////////////////////////////////////////////////////////////// FUNCIÓN DE PETICION DEL
LOCKER 8
void locker8(){
  digitalWrite(cerraduraPin4,LOW);              // Abrir la cerradura
  while(digitalRead(magnetoPin4)==LOW){}        // Esperar hasta que se separe la puerta
  datostx=120;                                  // Cargar dato que identifica que está abierta la puerta
  envio();                                      // Enviar dato al maestro
  while(digitalRead(magnetoPin4)==HIGH){}       // Espera hasta que se cierre la puerta
  digitalWrite(cerraduraPin4,HIGH);             // Cerrar la cerradura
  datostx=121;                                  // Cargar dato que identifica que está cerrada la puerta
  envio();                                      // Enviar dato al maestro
  digitalWrite(activePin4,LOW);                 // Activar sensor infrarrojo
  for(a=0;a<10000;a++){                         // Empieza a sensar  el infrarrojo
      respuesta=digitalRead(infraPin4);}        // Guardar el valor en la variable respuesta (0 o 1)
      digitalWrite(activePin4,HIGH);            // Apagar sensor infrarrojo
  if(respuesta==LOW){                           // Si es 0 detecta proyector
      digitalWrite(ledPin4,LOW);                // Encender luz verde
      datostx=27;                                // Cargar dato que identifica que hay proyector
      envio();}                                 // Enviar dato que si hay proyector al maestro
  else{                                         // Si es 1 no hay proyector
      digitalWrite(ledPin4,HIGH);               // Encender luz roja
      datostx=26;                                // Cargar dato que identifica que no hay proyector
      envio();}                                 // Enviar dato que no hay proyector al maestro
}
////////////////////////////////////////////////////////////////////////////// FUNCIÓN DE PETICION DEL LOCKER
11
void locker11(){
  digitalWrite(cerraduraPin5,LOW);              // Abrir la cerradura
  while(digitalRead(magnetoPin5)==LOW){}        // Esperar hasta que se separe la puerta
  datostx=120;                                  // Cargar dato que identifica que está abierta la puerta
  envio();                                      // Enviar dato al maestro
  while(digitalRead(magnetoPin5)==HIGH){}       // Espera hasta que se cierre la puerta
  digitalWrite(cerraduraPin5,HIGH);             // Cerrar la cerradura
  datostx=121;                                  // Cargar dato que identifica que está cerrada la puerta
  envio();                                      // Enviar dato al maestro
  digitalWrite(activePin5,LOW);                 // Activar sensor infrarrojo
  for(a=0;a<10000;a++){                         // Empieza a sensar  el infrarrojo
      respuesta=digitalRead(infraPin5);}        // Guardar el valor en la variable respuesta (0 o 1)
      digitalWrite(activePin5,HIGH);            // Apagar sensor infrarrojo
  if(respuesta==LOW){                           // Si es 0 detecta proyector
      digitalWrite(ledPin5,LOW);                // Encender luz verde
      datostx=29;                                // Cargar dato que identifica que hay proyector
      envio();}                                 // Enviar dato que si hay proyector al maestro
  else{                                         // Si es 1 no hay proyector
      digitalWrite(ledPin5,HIGH);               // Encender luz roja
```

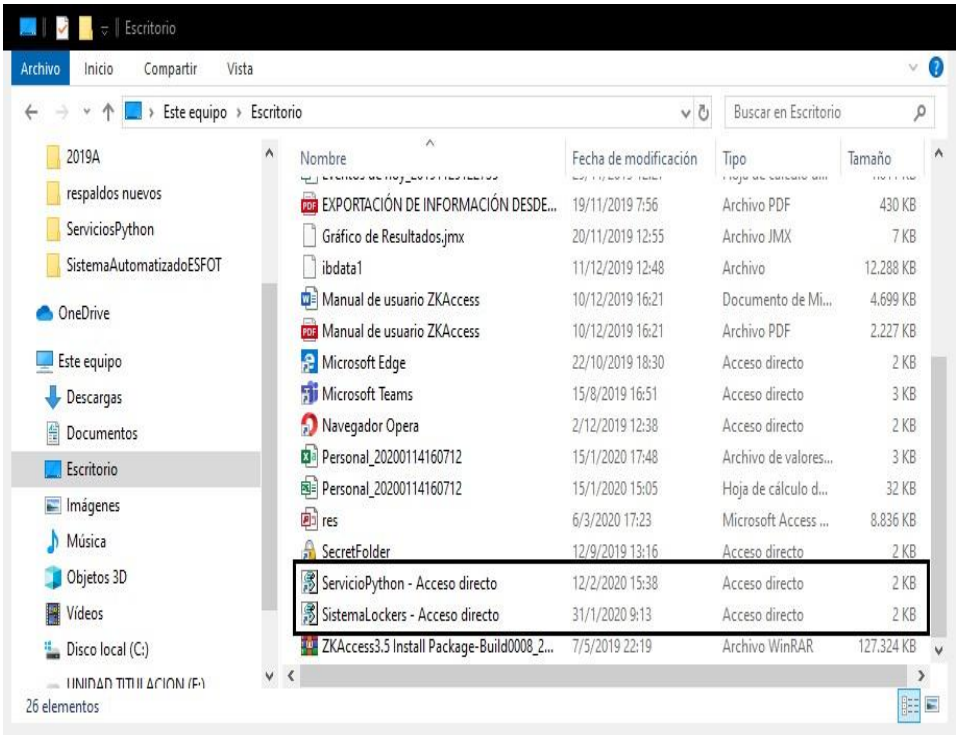
```
      datostx=28;                                  // Cargar dato que identifica que no hay proyector
      envio();}                                    // Enviar dato que no hay proyector al maestro
}
////////////////////////////////////////////////////////////////////////////// FUNCIÓN DE PETICION DEL LOCKER
12
void locker12(){
  digitalWrite(cerraduraPin6,LOW);                // Abrir la cerradura
  while(digitalRead(magnetoPin6)==LOW){}          // Esperar hasta que se separe la puerta
  datostx=120;                                    // Cargar dato que identifica que está abierta la puerta
  envio();                                        // Enviar dato al maestro
  while(digitalRead(magnetoPin6)==HIGH){}         // Espera hasta que se cierre la puerta
  digitalWrite(cerraduraPin6,HIGH);               // Cerrar la cerradura
  datostx=121;                                    // Cargar dato que identifica que está cerrada la puerta
  envio();                                        // Enviar dato al maestro
  digitalWrite(activePin6,LOW);                   // Activar sensor infrarrojo
  for(a=0;a<10000;a++){                           // Empieza a sensar  el infrarrojo
      respuesta=digitalRead(infraPin6);}          // Guardar el valor en la variable respuesta (0 o 1)
  digitalWrite(activePin6,HIGH);                  // Apagar sensor infrarrojo
  if(respuesta==LOW){                             // Si es 0 detecta proyector
      digitalWrite(ledPin6,LOW);                  // Encender luz verde
      datostx=31;                                 // Cargar dato que identifica que hay proyector
      envio();}                                   // Enviar dato que si hay proyector al maestro
  else{                                           // Si es 1 no hay proyector
      digitalWrite(ledPin6,HIGH);                 // Encender luz roja
      datostx=30;                                 // Cargar dato que identifica que no hay proyector
      envio();}                                   // Enviar dato que no hay proyector al maestro
}
////////////////////////////////////////////////////////////////////////////// FUNCIÓN DE PETICION DEL LOCKER
15
void locker15(){
  digitalWrite(cerraduraPin7,LOW);                // Abrir la cerradura
  while(digitalRead(magnetoPin7)==LOW){}          // Esperar hasta que se separe la puerta
  datostx=120;                                    // Cargar dato que identifica que está abierta la puerta
  envio();                                        // Enviar dato al maestro
  while(digitalRead(magnetoPin7)==HIGH){}         // Espera hasta que se cierre la puerta
  digitalWrite(cerraduraPin7,HIGH);               // Cerrar la cerradura
  datostx=121;                                    // Cargar dato que identifica que está cerrada la puerta
  envio();                                        // Enviar dato al maestro
  // digitalWrite(activePin7,LOW);                // Activar sensor infrarrojo(siempre activo)
  for(a=0;a<10000;a++){                           // Empieza a sensar  el infrarrojo
      respuesta=digitalRead(infraPin7);}          // Guardar el valor en la variable respuesta (0 o 1)
      // digitalWrite(activePin7,HIGH);
  if(respuesta==LOW){                             // Si es 0 detecta proyector
      digitalWrite(ledPin7,LOW);                  // Encender luz verde
      datostx=33;                                 // Cargar dato que identifica que hay proyector
      envio();}                                   // Enviar dato que si hay proyector al maestro
  else{                                           // Si es 1 no hay proyector
      digitalWrite(ledPin7,HIGH);                 // Encender luz roja
      datostx=32;                                 // Cargar dato que identifica que no hay proyector
      envio();}                                   // Enviar dato que no hay proyector al maestro
}
////////////////////////////////////////////////////////////////////////////// FUNCIÓN DE PETICION DEL LOCKER
16
void locker16(){
  digitalWrite(cerraduraPin8,LOW);                // Abrir la cerradura
  while(digitalRead(magnetoPin8)==LOW){}          // Esperar hasta que se separe la puerta
  datostx=120;                                    // Cargar dato que identifica que está abierta la puerta
  envio();                                        // Enviar dato al maestro
  while(digitalRead(magnetoPin8)==HIGH){}         // Espera hasta que se cierre la puerta
  digitalWrite(cerraduraPin8,HIGH);               // Cerrar la cerradura
  datostx=121;                                    // Cargar dato que identifica que está cerrada la puerta
  envio();                                        // Enviar dato al maestro
  digitalWrite(activePin8,LOW);                   // Activar sensor infrarrojo
  for(a=0;a<10000;a++){                           // Empieza a sensar  el infrarrojo
      respuesta=digitalRead(infraPin8);}          // Guardar el valor en la variable respuesta (0 o 1)
  digitalWrite(activePin8,HIGH);                  // Apagar sensor infrarrojo
  if(respuesta==LOW){                             // Si es 0 detecta proyector
      digitalWrite(ledPin8,LOW);                  // Encender luz verde
      datostx=35;                                 // Cargar dato que identifica que hay proyector
      envio();}                                   // Enviar dato que si hay proyector al maestro
  else{                                           // Si es 1 no hay proyector
      digitalWrite(ledPin8,HIGH);                 // Encender luz roja
      datostx=34;                                 // Cargar dato que identifica que no hay proyector
      envio();}                                   // Enviar dato que no hay proyector al maestro
}
////////////////////////////////////////////////////////////////////////////// FUNCIÓN DE PETICION DEL LOCKER
19
void locker19(){
  digitalWrite(cerraduraPin9,LOW);                // Abrir la cerradura
  while(digitalRead(magnetoPin9)==LOW){}          // Esperar hasta que se separe la puerta
  datostx=120;                                    // Cargar dato que identifica que está abierta la puerta
  envio();                                        // Enviar dato al maestro
```

```
  while(digitalRead(magnetoPin9)==HIGH){}       // Espera hasta que se cierre la puerta
  digitalWrite(cerraduraPin9,HIGH);             // Cerrar la cerradura
  datostx=121;                                  // Cargar dato que identifica que está cerrada la puerta
  envio();                                      // Enviar dato al maestro
   // digitalWrite(activePin9,LOW);             // Activar sensor infrarrojo(siempre activo)
  for(a=0;a<10000;a++){                         // Empieza a sensar  el infrarrojo
      respuesta=digitalRead(infraPin9);}        // Guardar el valor en la variable respuesta (0 o 1)
      // digitalWrite(activePin9,HIGH);
  if(respuesta==LOW){                           // Si es 0 detecta proyector
      digitalWrite(ledPin9,LOW);                // Encender luz verde
      datostx=37;                               // Cargar dato que identifica que hay proyector
      envio();}                                 // Enviar dato que si hay proyector al maestro
  else{                                         // Si es 1 no hay proyector
      digitalWrite(ledPin9,HIGH);               // Encender luz roja
      datostx=36;                               // Cargar dato que identifica que no hay proyector
      envio();}                                 // Enviar dato que no hay proyector al maestro
}
//////////////////////////////////////////////////////////////////////////////// FUNCIÓN DE PETICION DEL LOCKER
20
void locker20(){
 digitalWrite(cerraduraPin10,LOW);              // Abrir la cerradura
  while(digitalRead(magnetoPin10)==LOW){}       // Esperar hasta que se separe la puerta
  datostx=120;                                  // Cargar dato que identifica que está abierta la puerta
  envio();                                      // Enviar dato al maestro
  while(digitalRead(magnetoPin10)==HIGH){}      // Espera hasta que se cierre la puerta
  digitalWrite(cerraduraPin10,HIGH);            // Cerrar la cerradura
  datostx=121;                                  // Cargar dato que identifica que está cerrada la puerta
  envio();                                      // Enviar dato al maestro
  digitalWrite(activePin10,LOW);                // Activar sensor infrarrojo
  for(a=0;a<10000;a++){                         // Empieza a sensar  el infrarrojo
      respuesta=digitalRead(infraPin10);}       // Guardar el valor en la variable respuesta (0 o 1)
      digitalWrite(activePin10,HIGH);           // Apagar sensor infrarrojo
  if(respuesta==LOW){                           // Si es 0 detecta proyector
      digitalWrite(ledPin10,LOW);               // Encender luz verde
      datostx=39;                               // Cargar dato que identifica que hay proyector
      envio();}                                 // Enviar dato que si hay proyector al maestro
  else{                                         // Si es 1 no hay proyector
      digitalWrite(ledPin10,HIGH);              // Encender luz roja
      datostx=38;                               // Cargar dato que identifica que no hay proyector
      envio();}                                 // Enviar dato que no hay proyector al maestro
}
////////////////////////////////////////////////////////////////////////////////FUNCION CUANDO SE RECIBE UN DATO POR
I2C
//Función que se ejecuta siempre que se reciben datos del master, siempre que en el master ejecute la
//sentencia endTransmission recibirá toda la información que se transmitirá a través de la sentencia
Wire.write
void receiveEvent() {
  digitalWrite(52,HIGH);              // Encender led indicador de que se recibió un dato del maestro
  if (Wire.available() == 1){         // Si hay un byte disponible
   datorx = Wire.read();              // Leer y cargar la variable
    }
  delay(500);                         // Pausa de medio segundo
  digitalWrite(52,LOW);               // Apagar led indicador de que se recibió un dato del maestro
}
////////////////////////////////////////////////////////// FUNCIONES DE ENVIO DE DATOS A LOS DEMÁS
DISPOSITIVOS
void envio(){                        // FUNCION PARA ENVIAR DATOS AL MAESTRO
    Wire.beginTransmission(0);       // Comenzar la transmisión al dispositivo 0
    Wire.write(datostx);             // Enviar un byte
    Wire.endTransmission();          // Parar la transmisión
    datorx=0;                        // Encerar variables
    datostx=0;
    delay(1000);                     // Esperar 1 segundo
}
```

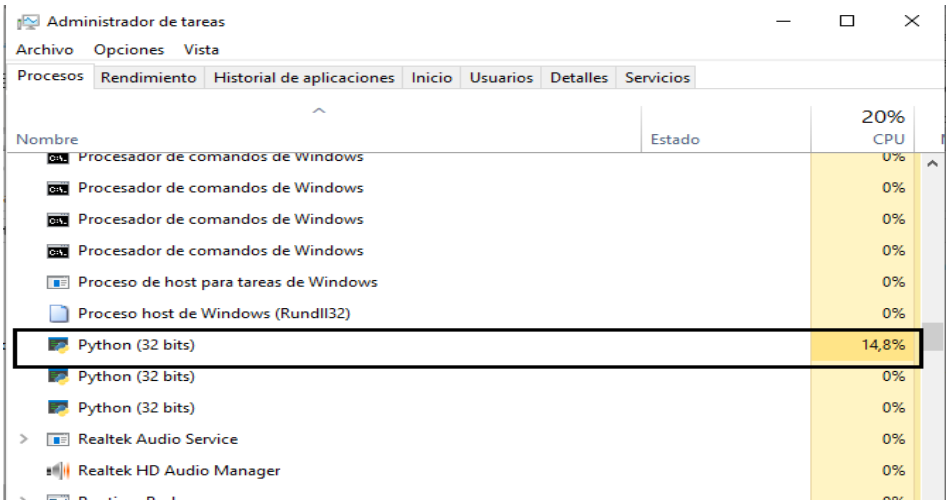
## ANEXO C: Manual de usuario

### Inicio del Sistema

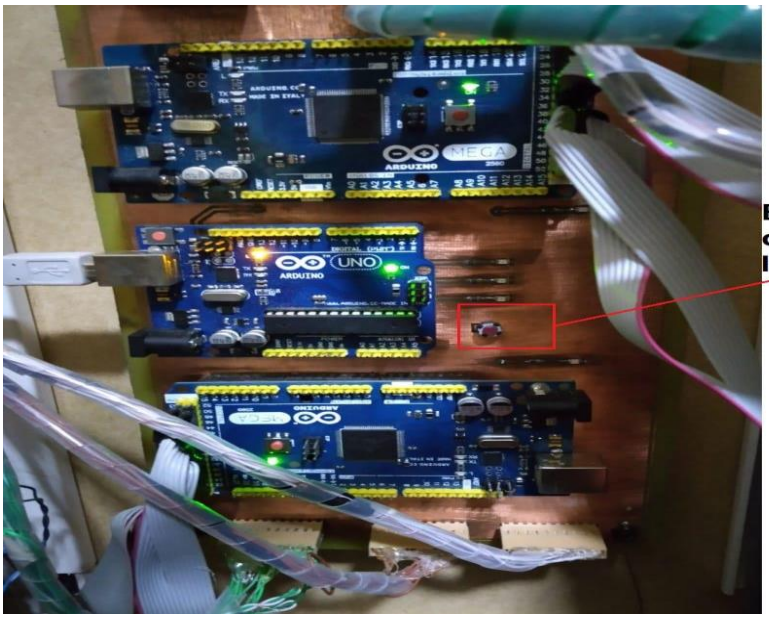
- Para iniciar el sistema se debe realizar la conexión USB del Arduino.
- Posteriormente, se accederá a los siguientes archivos dentro del servidor:




- Para comprobar que los archivos se estén ejecutando, se lo puede hacer a través del Administrador de Tareas.



- Una vez verificada la conexión, se procederá a abrir la puerta posterior de la estructura para pulsar el botón de inicio de la placa principal.




## 1. ACCESO AL SISTEMA (ADMINISTRADORES)

- Para ingresar al aplicativo, se debe ingresar al navegador y digitar la dirección IP: 192.168.1.100 y buscar.
- Para realizar la gestión de los proyectores, en la pantalla de la tablet se observará:


- Se procederá a la identificación en el lector biométrico.

- Al acceder al sistema, se podrá observar lo siguiente:


- En la pantalla se puede observar que los únicos proyectores que se encuentran en los casilleros son el número 13 y el número 14.
- Para ingresar el resto de proyectores, se debe presionar el botón “Lista Proyectores”, y se podrá observar la siguiente pantalla:

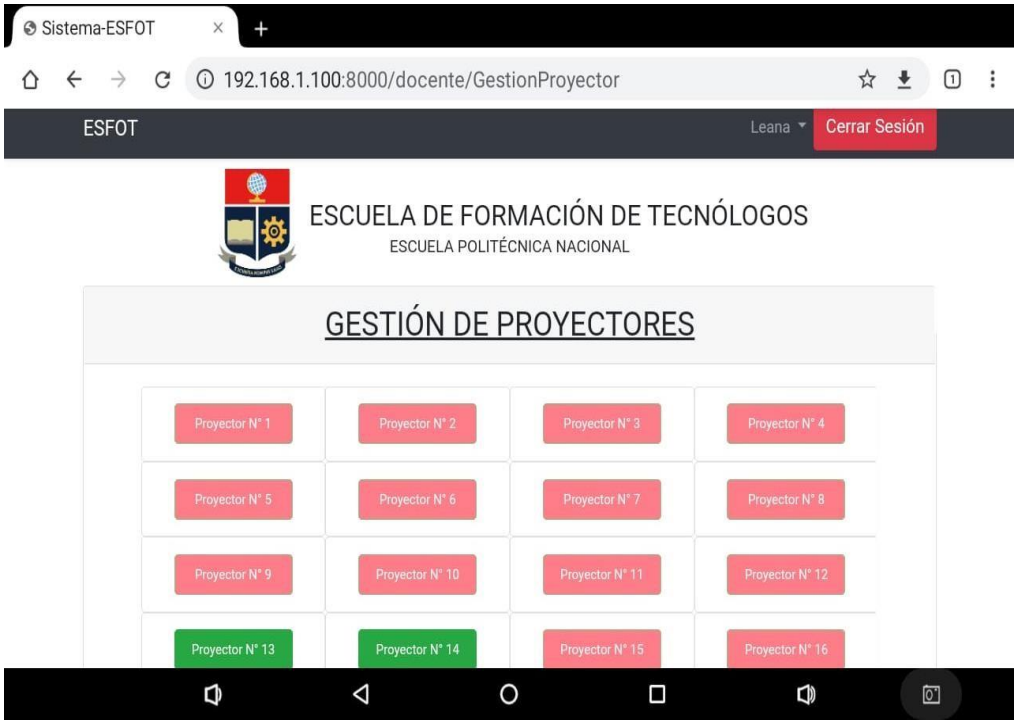
- Aquí se muestra la lista de todos los proyectores con cada una de las características. Para abrir la puerta del casillero correspondiente, se debe pulsar el botón “Abrir” acorde con el número del proyector o *kit;* la puerta se abrirá automáticamente. Este proceso no puede realizarse en conjunto; es decir, no se puede abrir todas las puertas al mismo tiempo. Por lo tanto, se debe ingresar los dispositivos uno por uno.
- No olvide **cerrar sesión.**

## 2. PRÉSTAMO Y DEVOLUCIÓN DE PROYECTORES

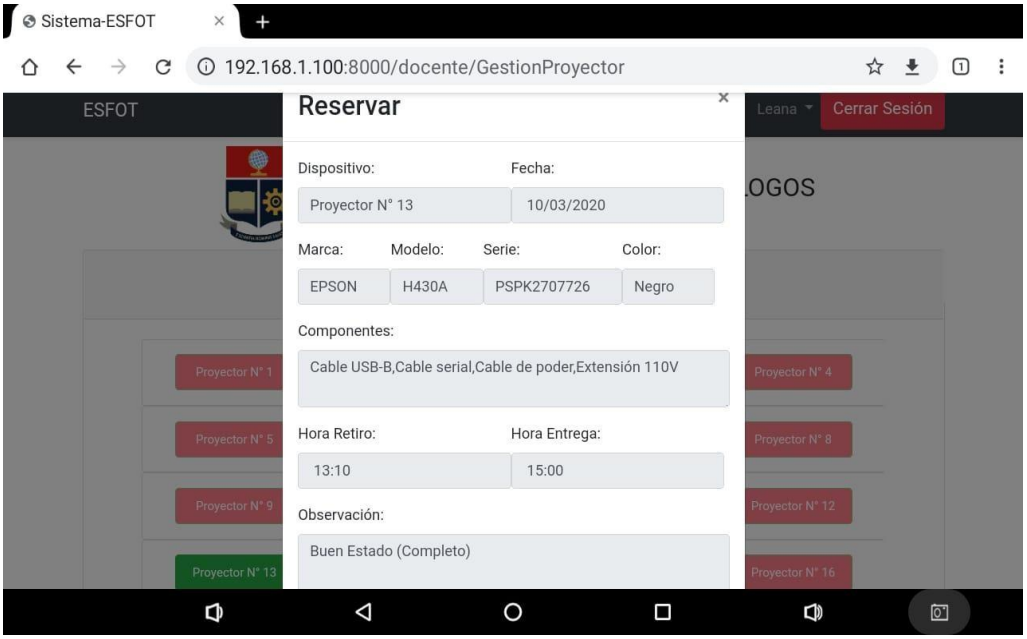
- Para realizar la petición de un proyector, en la pantalla de la tablet se observará:


- Posterior, se procederá a la identificación en el lector biométrico.
- Una vez ingresado al sistema, se podrá observar lo siguiente:

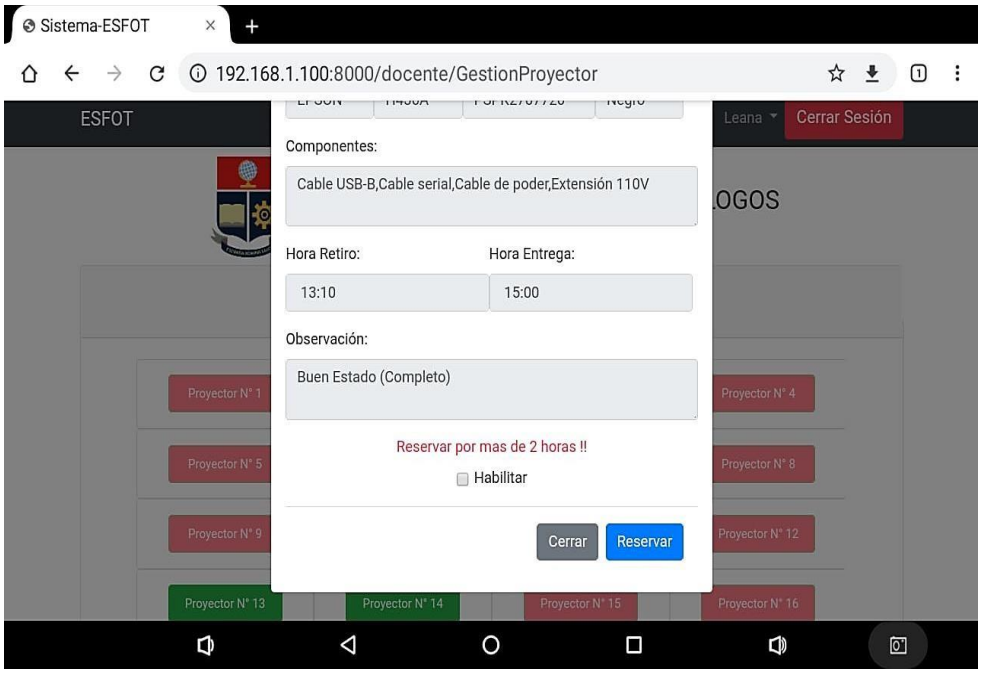
- Se debe acceder a la opción “Gestión Proyectores”, y se mostrará los proyectores que están disponibles con el casillero de color verde, mientras que la ausencia de estos presenta color rojo:



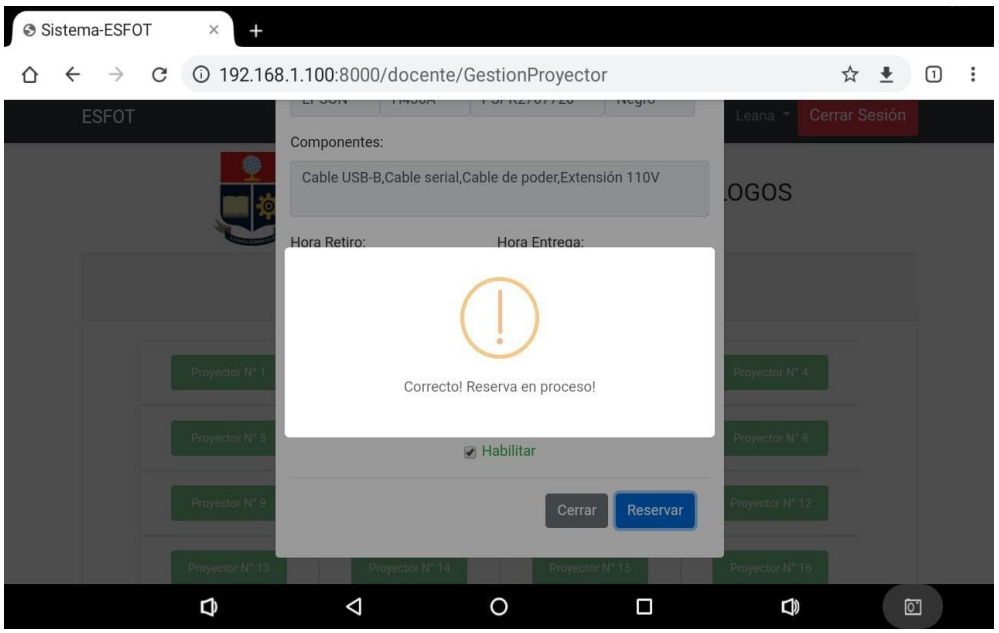

- En la pantalla se puede observar que los únicos proyectores disponibles son el número 13 y 14; es decir, solo se puede gestionar cualquiera de los dos, y se procede a seleccionar:

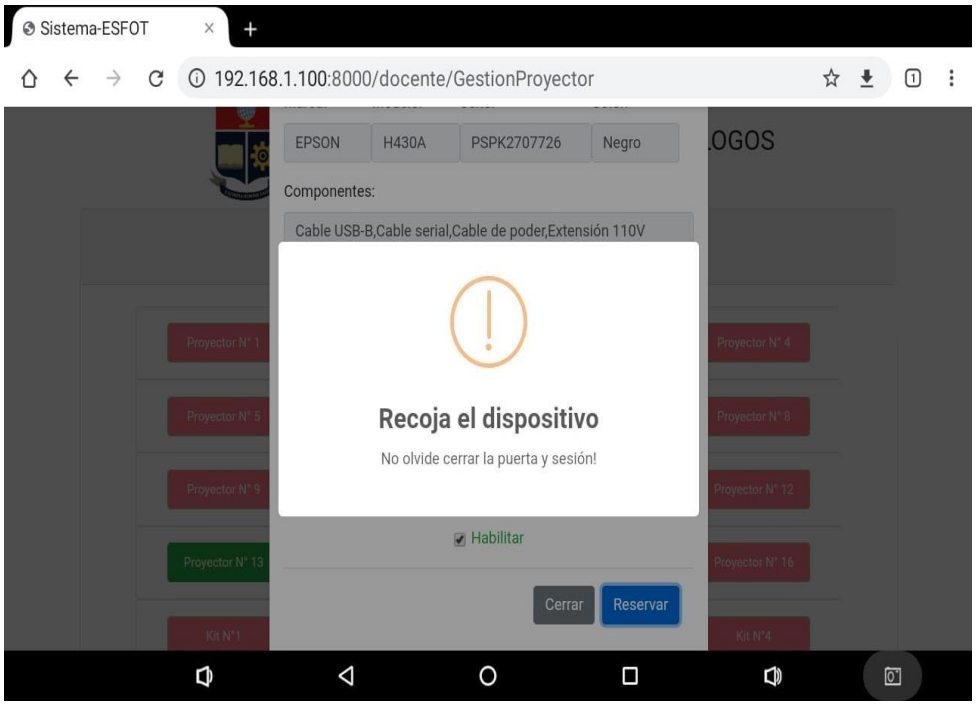


- En la imagen, se seleccionó el proyector número 13, en la cual se puede evidenciar la marca, modelo, serie, color y los componentes que posee el dispositivo, así como la hora de retiro y la hora de entrega ya establecidas, y si presenta alguna observación.
- Para realizar la reserva del dispositivo, se debe ir a la parte inferior de esta pantalla, en la que se podrá observar:

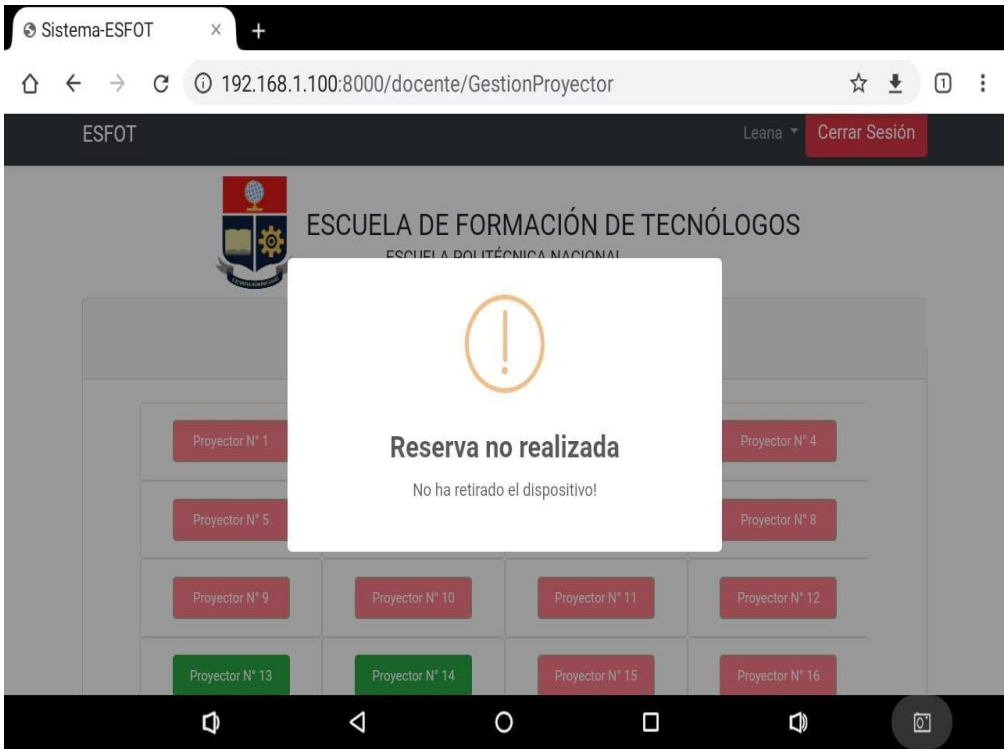



- En este paso se puede hacer la reserva por más de dos horas al seleccionar en el botón “Habilitar” o simplemente realizar la reserva, al tocar en el botón “Reservar”, mismo que abrirá la puerta del proyector seleccionado.

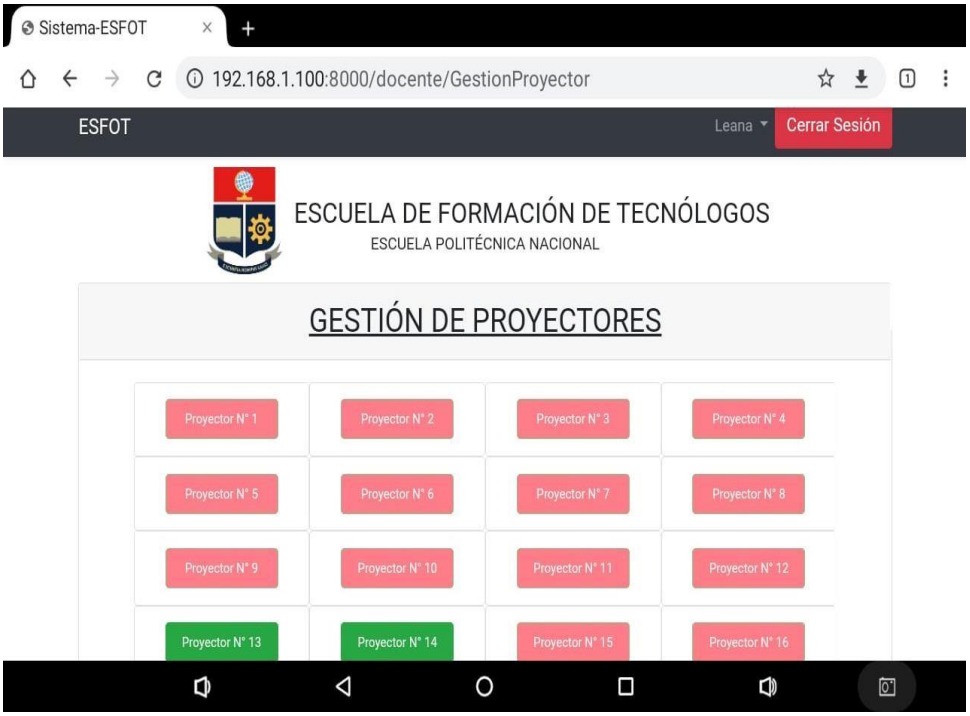




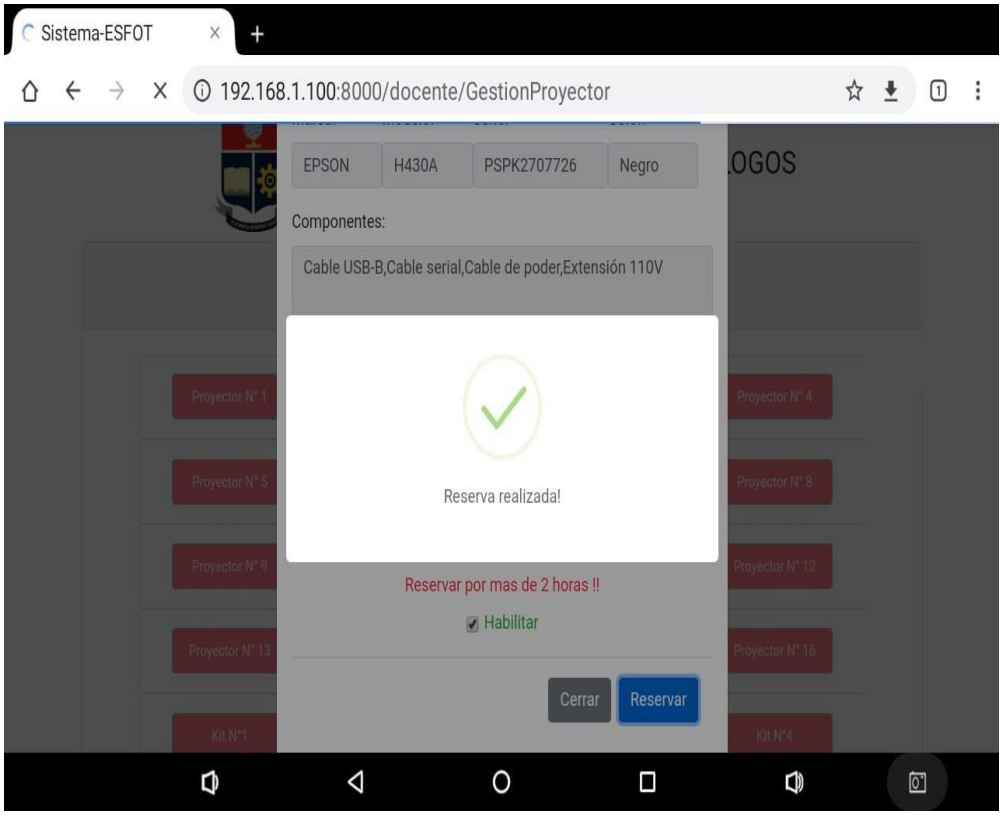

- Una vez, abierta la puerta, en caso de que el docente no requiera dicho dispositivo, y cierre la puerta del casillero, se podrá observar el siguiente mensaje:



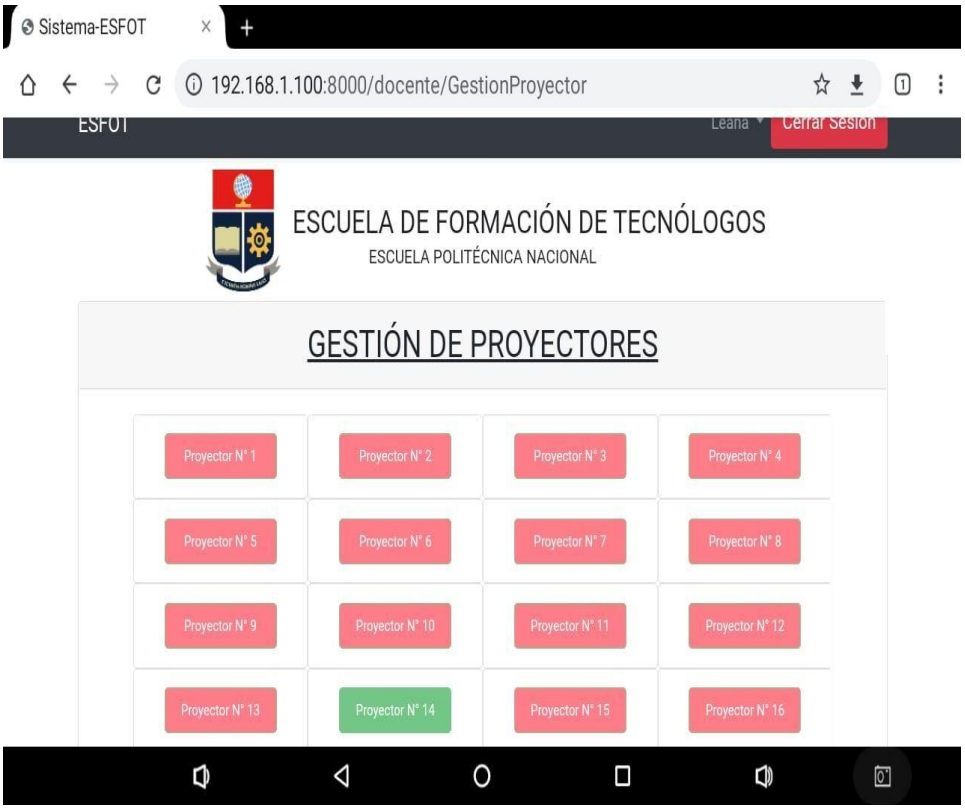
- Y se mostrará nuevamente la pantalla, con los proyectores disponibles:



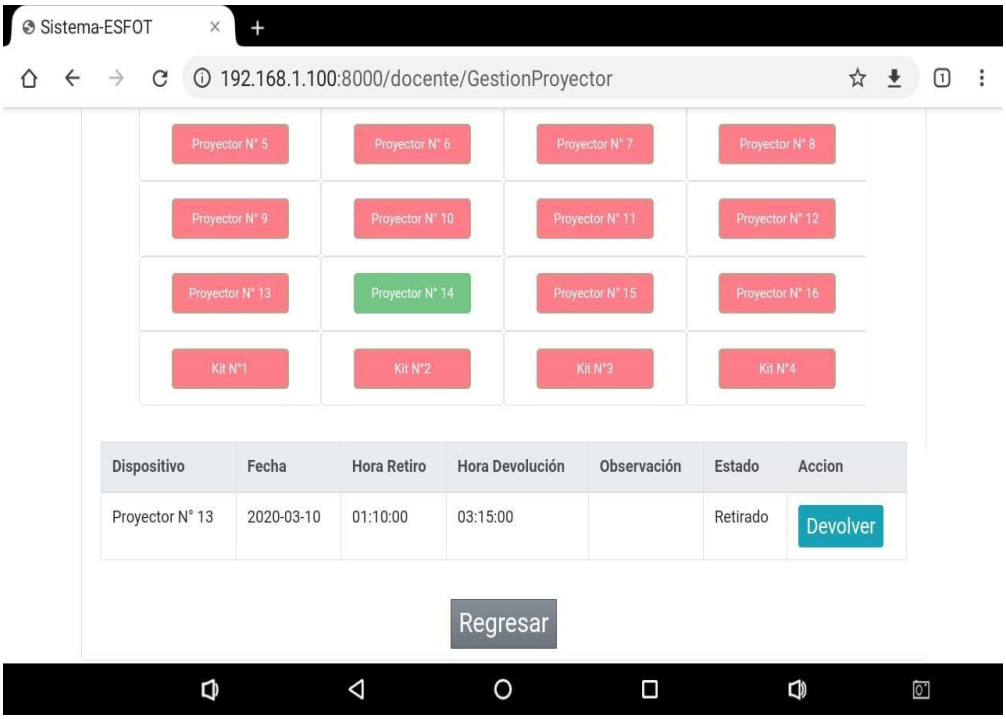

- Caso contrario al retirar el dispositivo del casillero, y cerrar la puerta, se mostrará el siguiente mensaje, y posterior la ventana con los proyectores disponibles, **no olvide cerrar la sesión**.



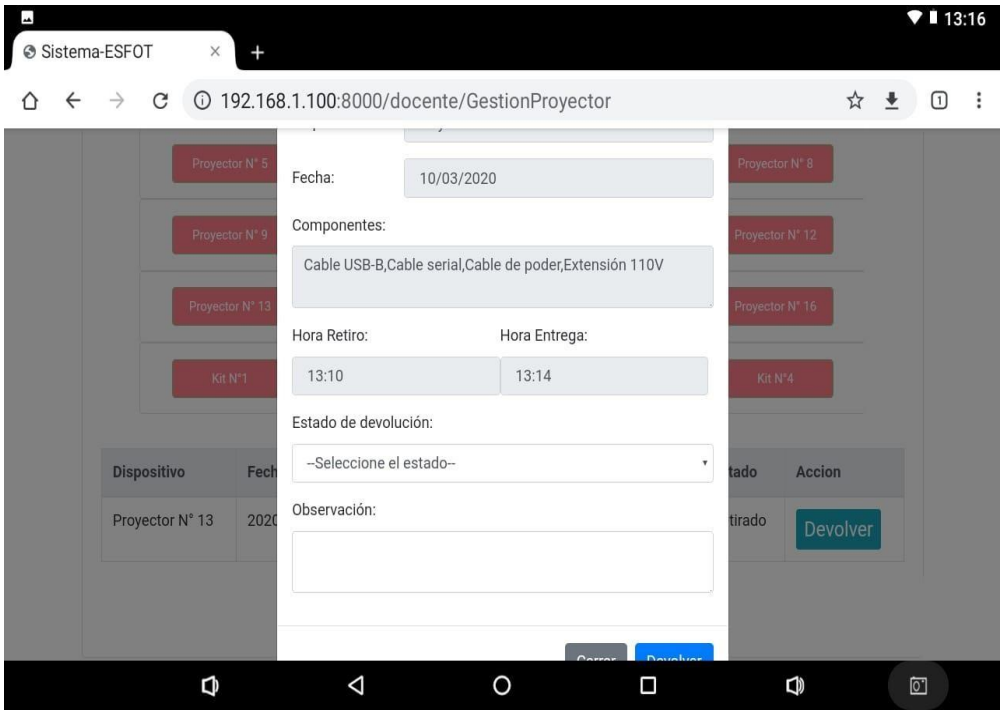

- Para devolver el dispositivo se debe realizar los pasos explicados anteriormente para iniciar sesión nuevamente, hasta observar la pantalla de los dispositivos disponibles:



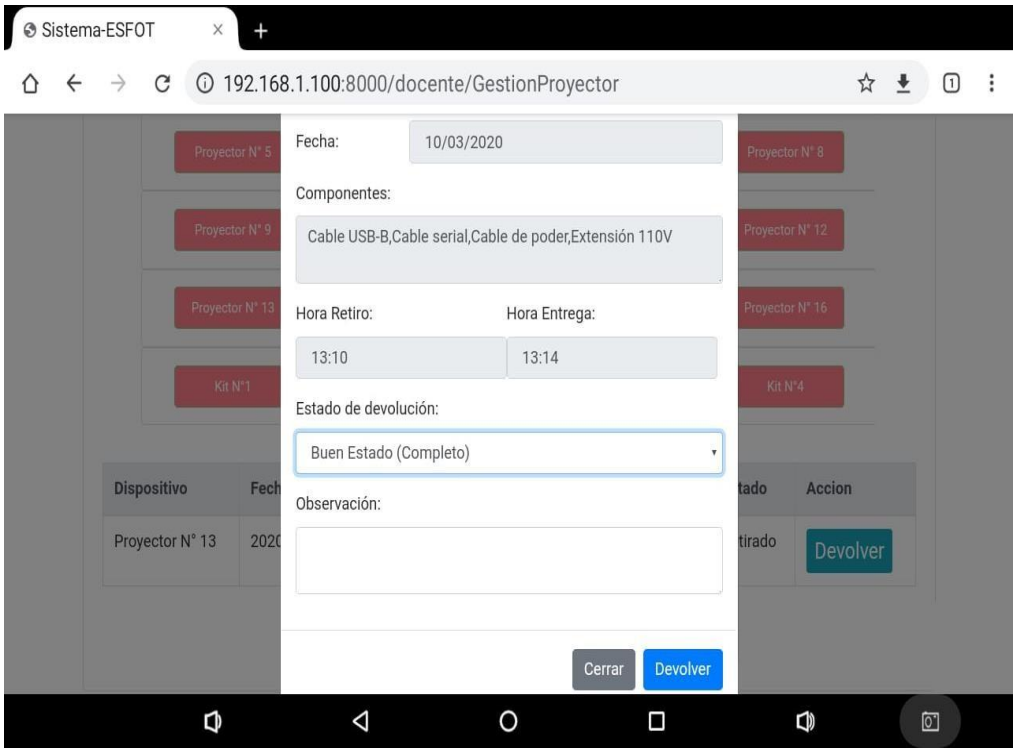
- En este paso, se procede a ir a la parte inferior de la ventana:



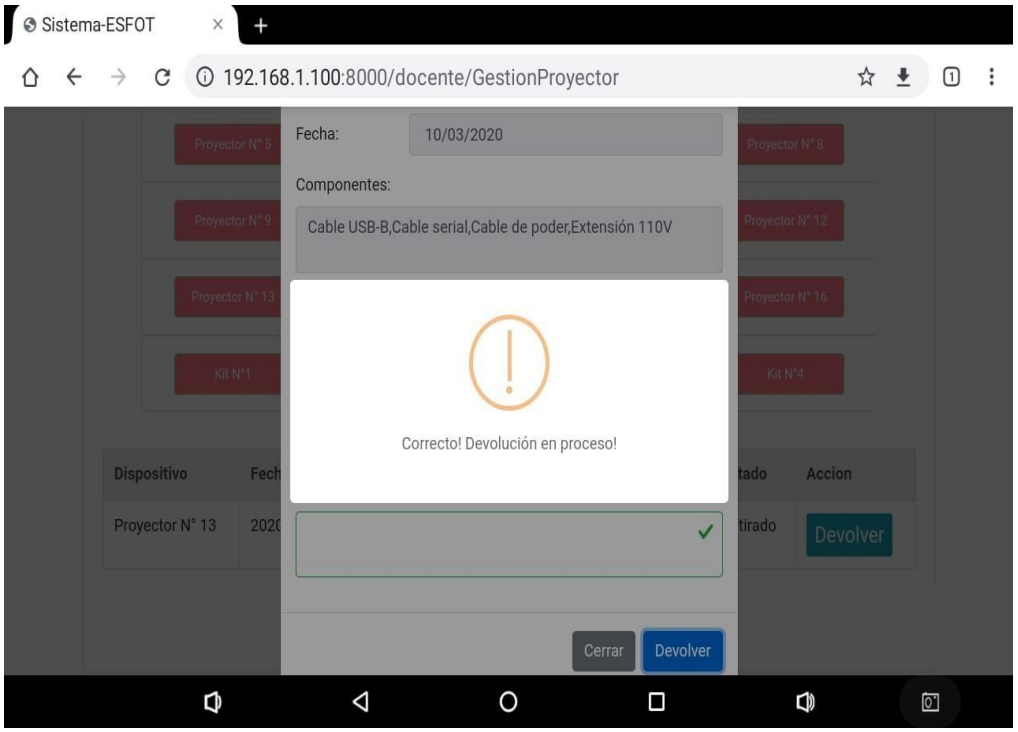

- Aquí se presenta el dispositivo que ha retirado el docente, en conjunto con la opción “Devolver”, en la cual se debe acceder para regresar el dispositivo.



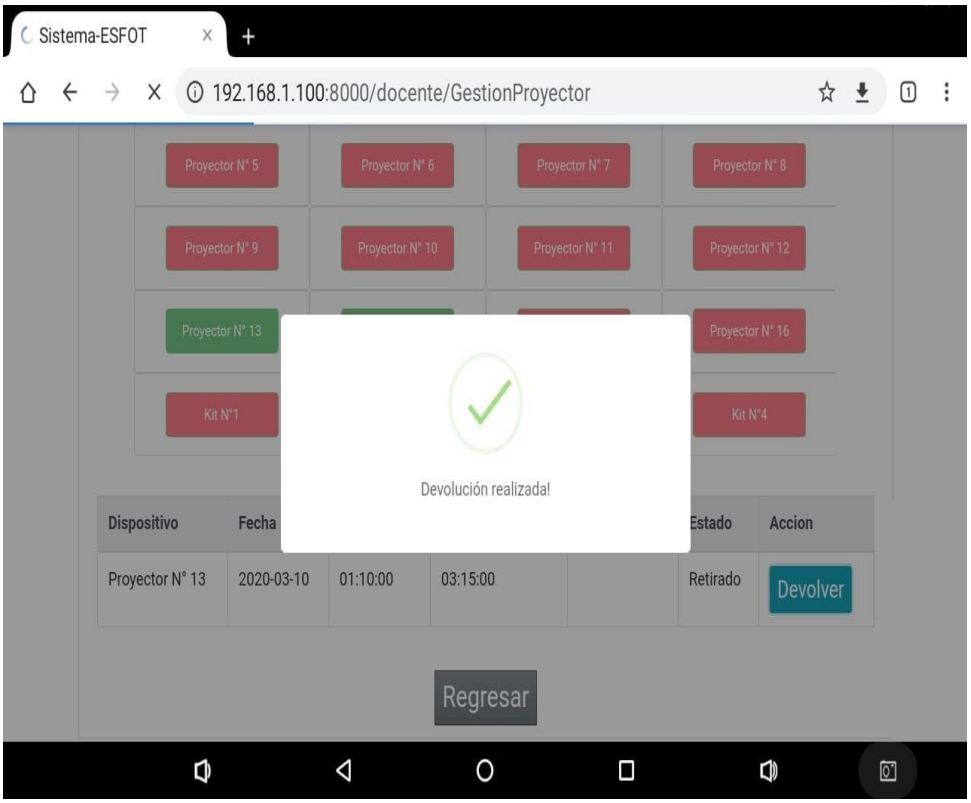

- Se muestra la pantalla en la que se evidenció las características del proyector; en la parte inferior se podrá observar un cuadro específico “Estado de devolución” el cual se debe seleccionar, para dar a conocer el estado en que se regresa el proyector.




- Una vez seleccionado el estado del proyector, se procede a regresar el dispositivo con el botón “Devolver”.

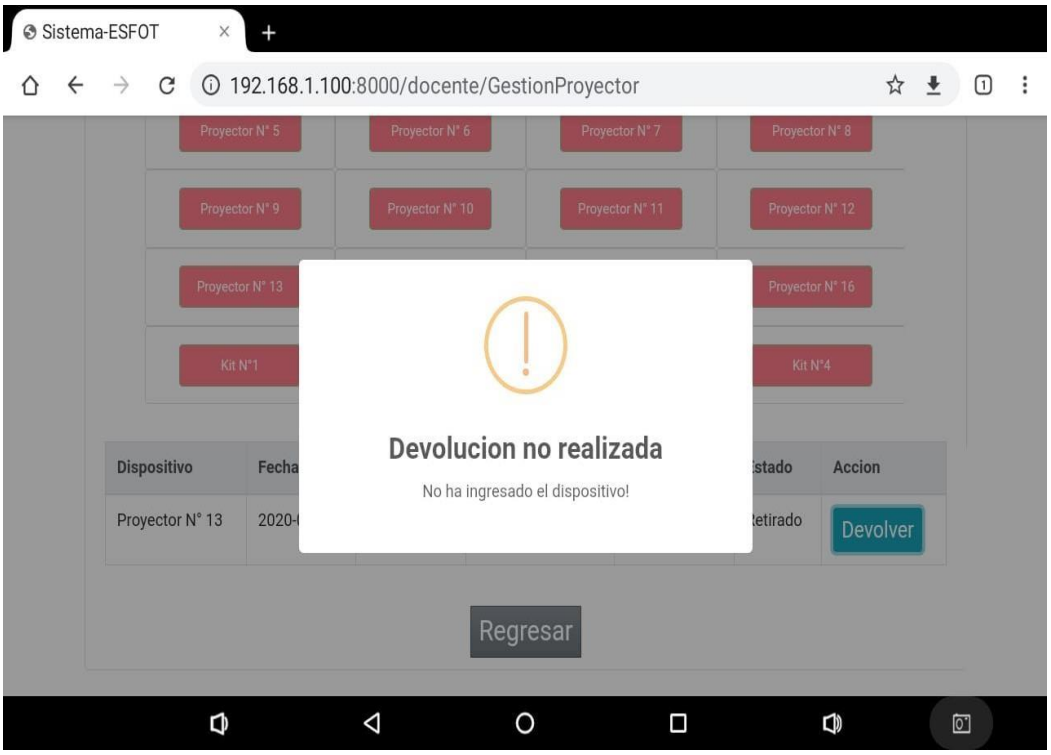


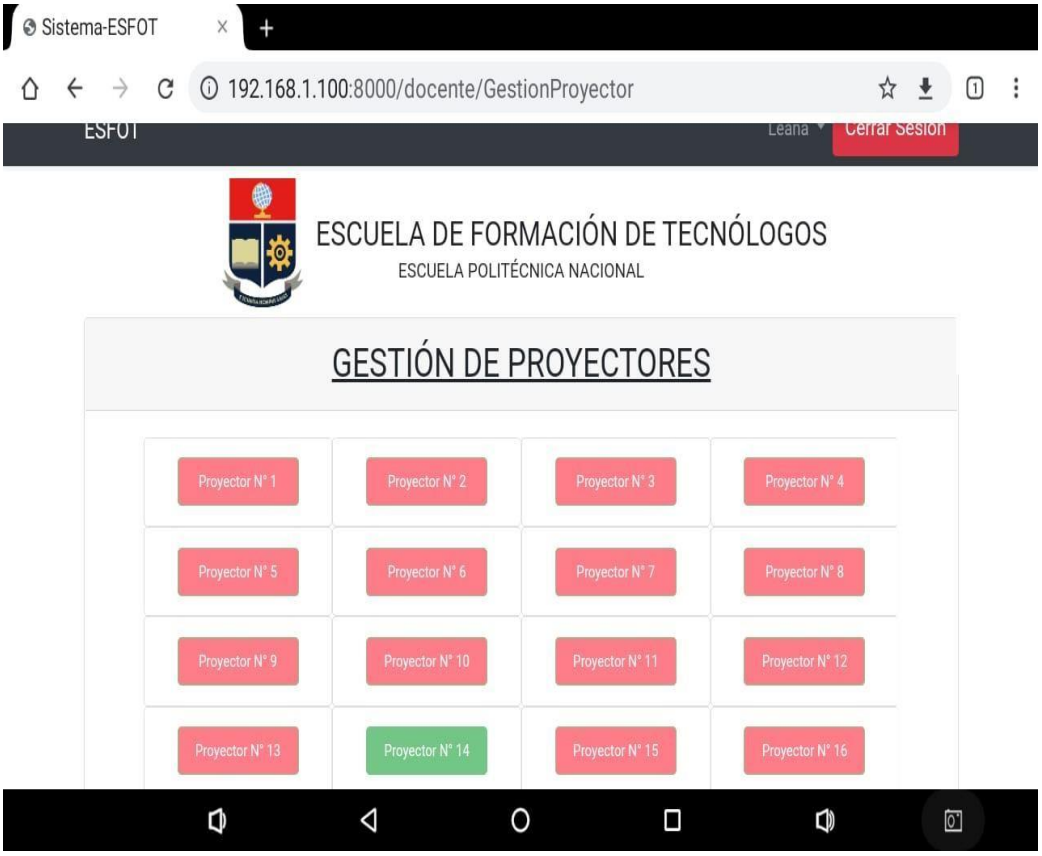

- La puerta del dispositivo a devolver se abrirá, y posterior se procede a ingresar el proyector en el casillero, y cerrar la puerta. Realizado el proceso, se podrá observar en la pantalla:




- Y se mostrará nuevamente la pantalla con los dispositivos disponibles; **no olvide cerrar sesión.**

- Caso contrario, al no ingresar el dispositivo, se mostrará el siguiente mensaje, en conjunto con la ventana de los dispositivos disponibles, y se debe realizar los procesos vistos anteriormente.

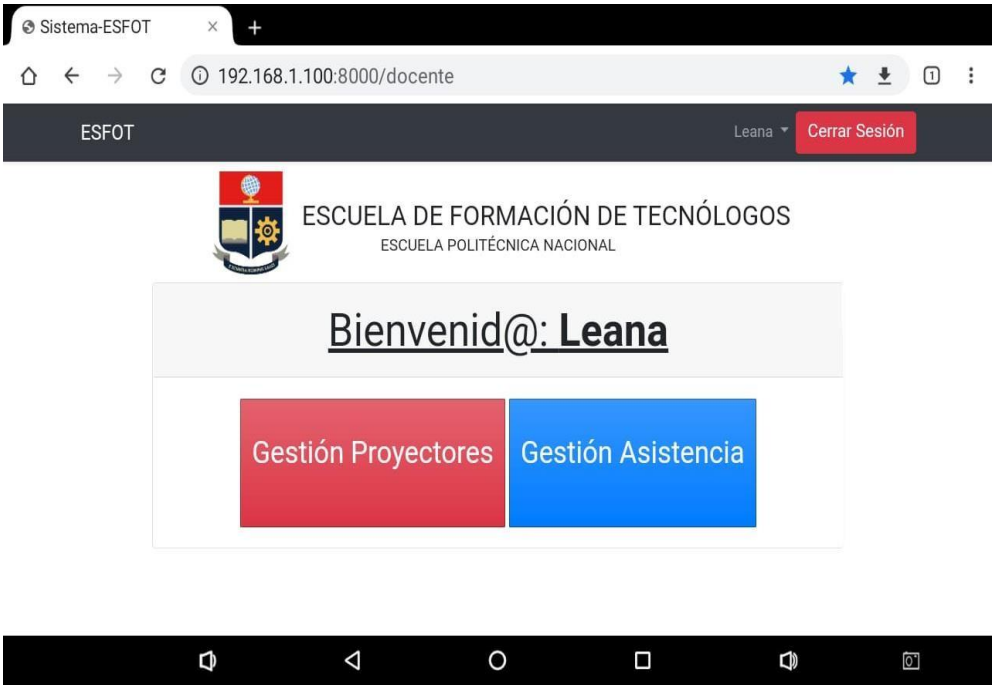


## 3. GESTIÓN DE ASISTENCIA

- Para el registro de asistencia, en la pantalla de la tablet se observará:

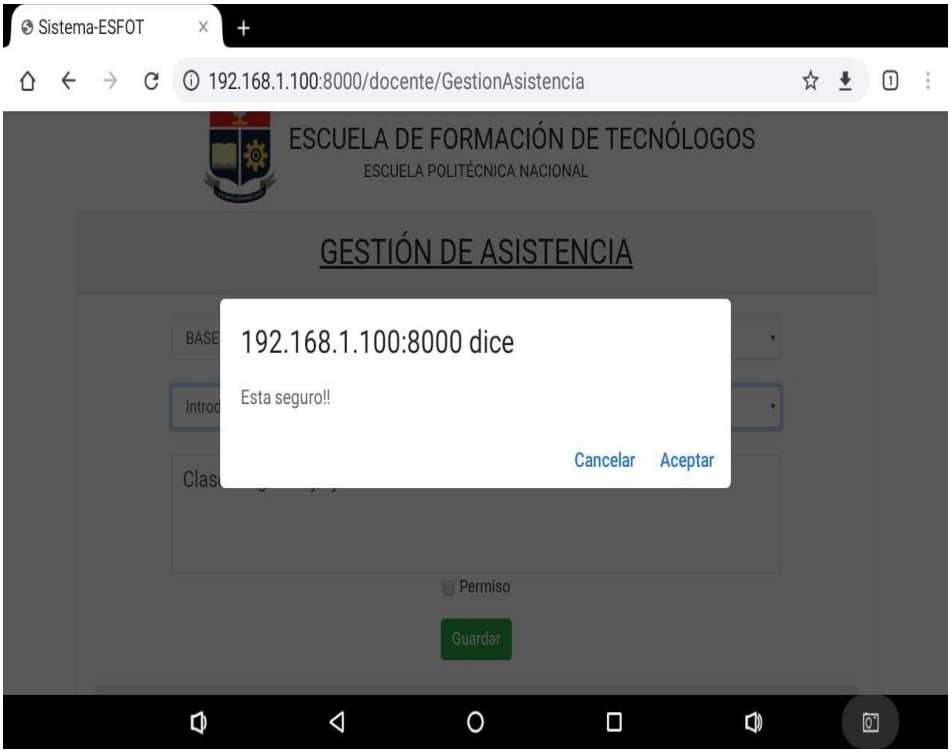

- Posterior, se procederá a la identificación en el lector biométrico.
- Una vez ingresado al sistema, se podrá observar lo siguiente:



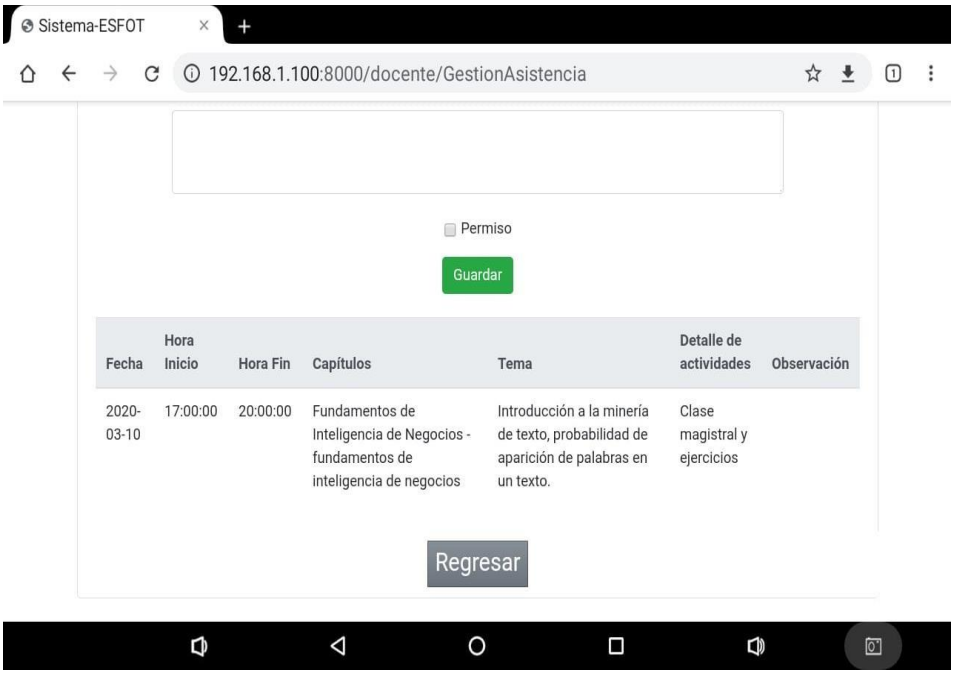
- Se debe acceder a la opción “Gestión Asistencia”, y a continuación se mostrará la siguiente pantalla:


- Las materias, y los temas se presentarán en los cuadros respectivos, acorde con los sílabos previamente registrados. Por ejemplo:

- Una vez seleccionada la información de la clase correspondiente, se presiona el botón guardar, y se podrá observar la siguiente ventana.

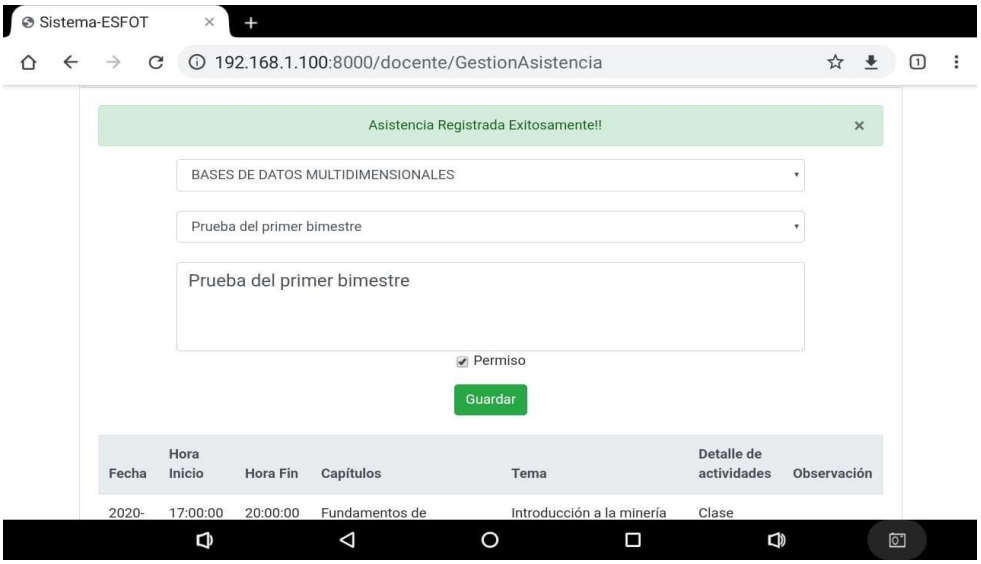


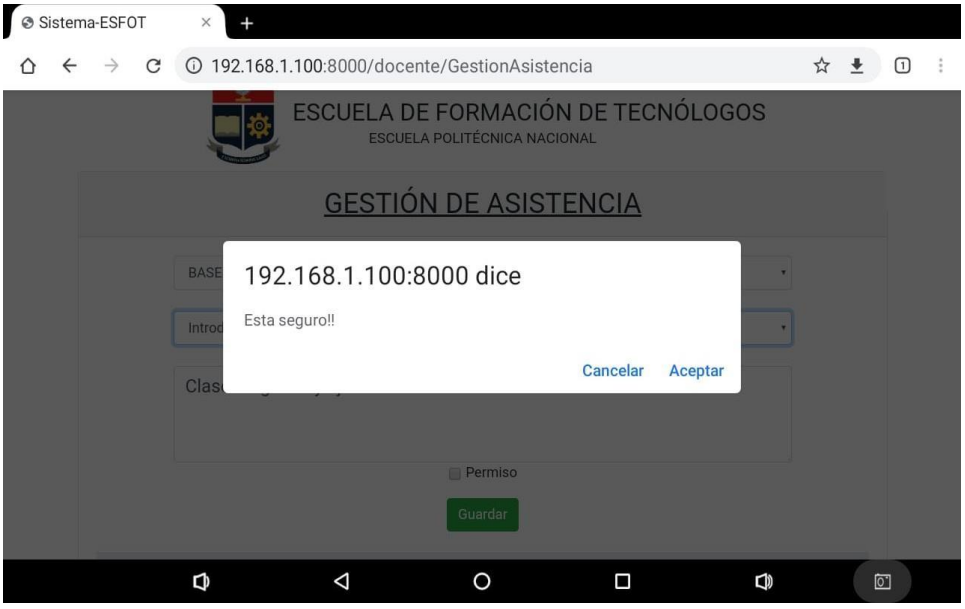

- Se presiona en el botón aceptar y así quedará registrada la respectiva asistencia.




- No olvide cerrar sesión.

- **Para el registro de clases de recuperación, se debe seguir los pasos anteriores vistos en este ítem,** pero se debe pulsar previamente en la opción “Permiso” y se procede a guardar, es decir:



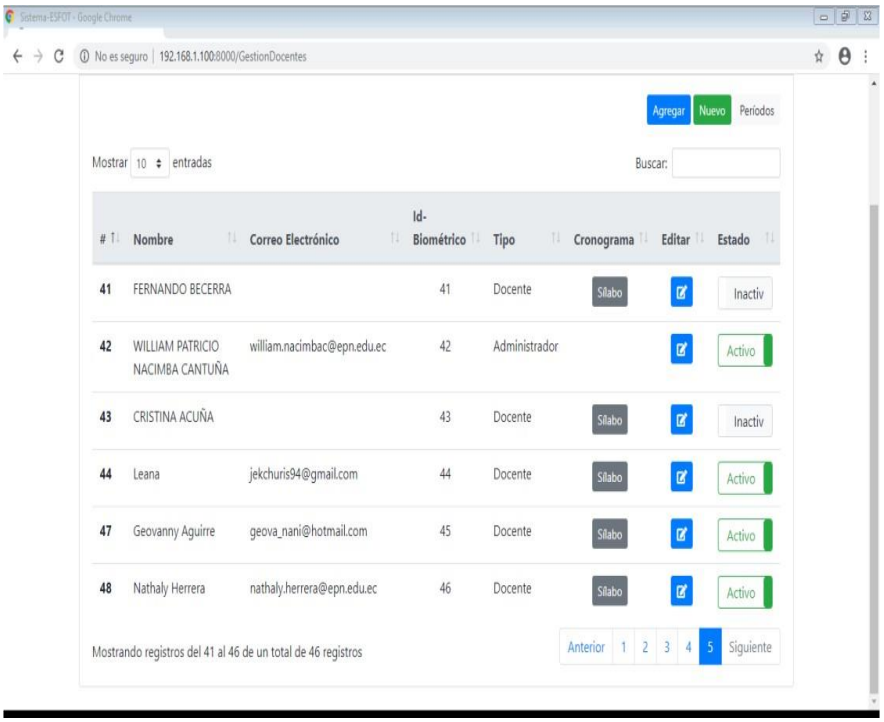




- No olvide cerrar sesión.

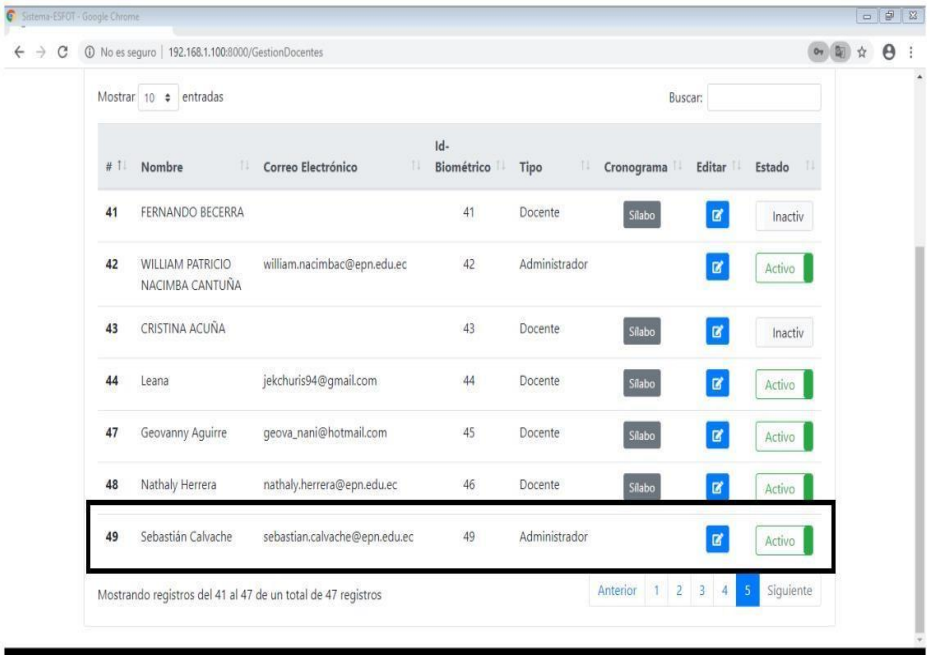
## 4. AÑADIR USUARIOS NUEVOS AL SISTEMA

- Para añadir nuevos usuarios, el administrador deberá ingresar al sistema desde el servidor:

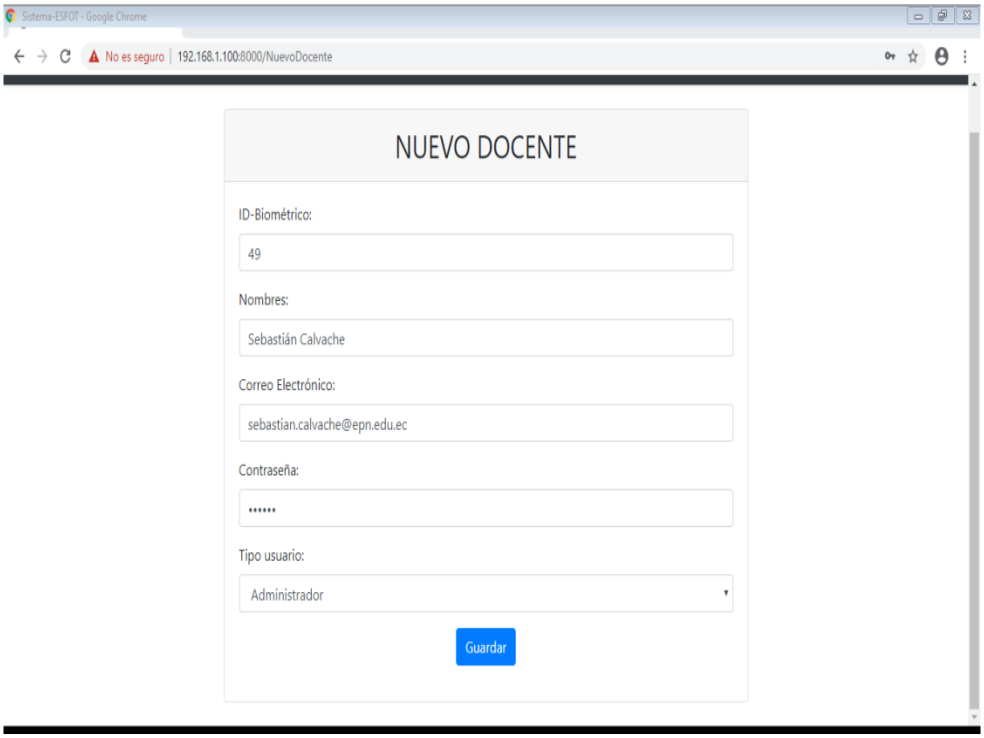

- En el botón “LOGIN”, deberá ingresar la información respectiva, posteriormente se podrá observar la siguiente pantalla:




- En esta ventana, se podrá activar o mantener inhabilitados a los docentes o administradores, respectivamente.




- Para ingresar a un administrador o docente en el sistema, se deberá presionar en el botón “Nuevo” y llenar los datos respectivos. Por ejemplo:

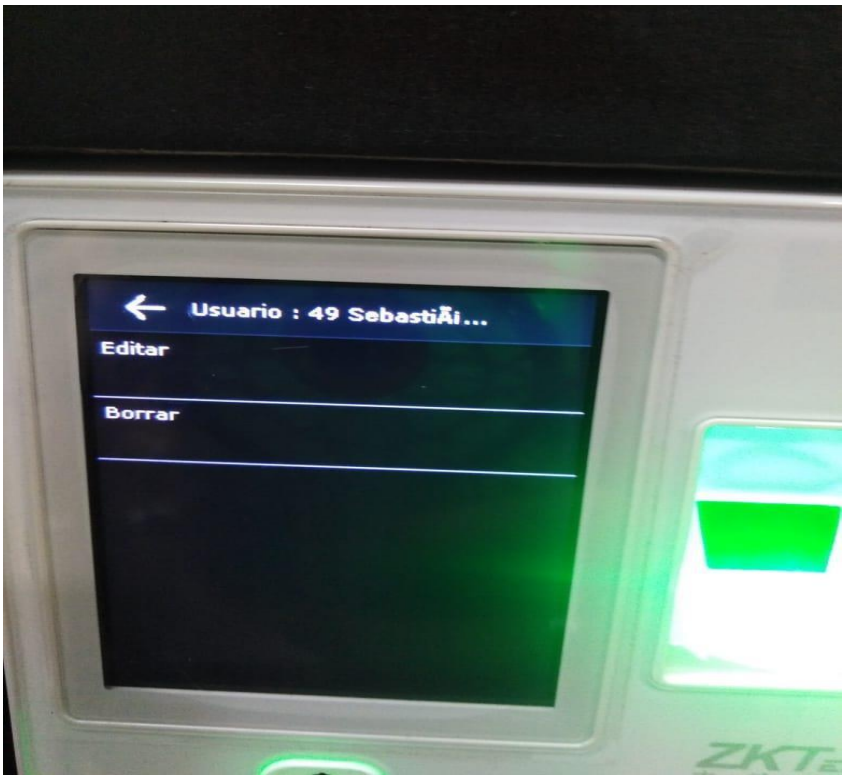



- Una vez terminado el proceso, se pulsa el botón guardar. Cabe señalar que, en este paso se selecciona el “tipo de usuario” que será ingresado (docente / administrador).
- Posteriormente, se podrá evidenciar al usuario nuevo ya añadido.

- Acto seguido, se deberá llevar al usuario a registrar su huella dactilar en el lector biométrico.
- Se debe acceder al menú del biométrico.


- Presione en el botón “User Mgt.”, posteriormente se mostrará la lista de todos los usuarios incluidos en el sistema.

- Pulsar en el nombre del usuario añadido recientemente, y se podrá identificar lo siguiente.




- Presione en el botón “Editar” y por consiguiente aparecerá esta ventana:

- Se puede visualizar que en el cuadro “Huella” se presenta el número 0, esto quiere decir que el usuario aún no está registrado en el biométrico, por lo tanto, seleccionar este botón para configurar el registro del usuario.


- Se podrá observar la siguiente pantalla, con los diferentes dedos, donde se seleccionará el dedo que va a grabar en el lector, el proceso se realiza con un solo dedo.

- El usuario deberá mantener su dedo a grabar en el lector biométrico, hasta que el porcentaje de su huella esté al 100%.
- El usuario quedará registrado, y se podrá evidenciar en la pantalla del biométrico lo siguiente:


- Finalmente, en el cuadro “Huella” se puede observar que el valor cambió a 1, siendo este el número de huellas registradas por usuario. Si desea grabar más huellas del mismo usuario, lo puede realizar siguiendo los pasos señalados anteriormente.

**ADVERTENCIA**

- En caso de reinicio del servidor, se esperará a que el sistema operativo se ejecute correctamente, acto seguido se procederá a reiniciar los Arduinos y posteriormente se seguirá con los pasos vistos en el punto 1.
- Cuando el sistema no se encienda, en primer lugar, se deberá revisar la fuente principal (regulador de voltaje); si los LED’s indicadores están apagados, se tiene que reemplazar el fusible correspondiente, su valor está etiquetado en la fuente.

Una vez verificado el correcto funcionamiento de la fuente principal, y aun así el sistema presenta alguna falla, se procederá a apagar el regulador de voltaje y a revisar los fusibles, para lo cual se va a identificar las etiquetas que se encuentran ubicadas en cada cable de las diferentes fuentes:

- 110v → Fusible 3A
- 12v → Fusible 3A
- 5v → Fusible 1A

En caso de encontrar algún fusible quemado, sustituir con su correspondiente