* **UTN Facultad Regional Mendoza**

**Informática II**

**Trabajo Final**

**Nombre proyecto**

CAJA FUERTE ELECTRONICA

Profesor Titular: Ing. Marcelo Ledda

Profesora JTP: Ing. Diedrichs, Ana Laura

Nombres Vazquez Anibal Gabriel

Fecha:29-11-2023

Curso: Indique 2R20

## INDICE

[INDICE](#_heading=)

[1. Introducción](#_heading=)

[2. Diagrama funcional](#_heading=)

[3. Código fuente de los Programas](#_heading=)

[4. Salidas del Programa](#_heading=)

[5. Anexos](#_heading=h.uqmwlgtf6z77)

[5. 1 Esquemático](#_heading=h.wzbei8qwbnc9)

[5.2 Listado de componentes](#_heading=h.yasxqoqrwecb)

[5.3 Referencias](#_heading=h.739mrylheo4r)

[5.4 Otros anexos que deba incluir](#_heading=h.hurpfp5kmxbt)

## 1. Introducción

Este informe documenta el desarrollo de un proyecto que combina programación en lenguaje C++ utilizando la plataforma Arduino y programación visual con Processing. Este proyecto tiene como objetivo principal la creación de una caja fuerte electrónica. La interfaz de esta caja fuerte es controlada por un sistema Arduino, y la visualización de datos se realiza a través de una interfaz gráfica desarrollada en Processing.

Descripción Breve del Trabajo:

La caja fuerte electrónica diseñada durante este proyecto tiene la capacidad de registrar aperturas y cierres, brindando una funcionalidad de seguridad avanzada. La comunicación entre Arduino y Processing se establece mediante la comunicación serie, permitiendo la supervisión en tiempo real de las operaciones de la caja fuerte en una interfaz gráfica fácil de entender.

Herramientas Utilizadas:

Se utilizó las siguientes herramientas:

Arduino: Una plataforma que facilita la creación de prototipos electrónicos.

Processing: Un entorno de desarrollo para la creación de aplicaciones visuales y gráficas.

Lenguaje C++: Empleado para programar tanto el Arduino como parte de la lógica del proyecto.

Librerías estándar de Arduino y Processing: Se utilizaron para aprovechar funciones preexistentes y simplificar la implementación.

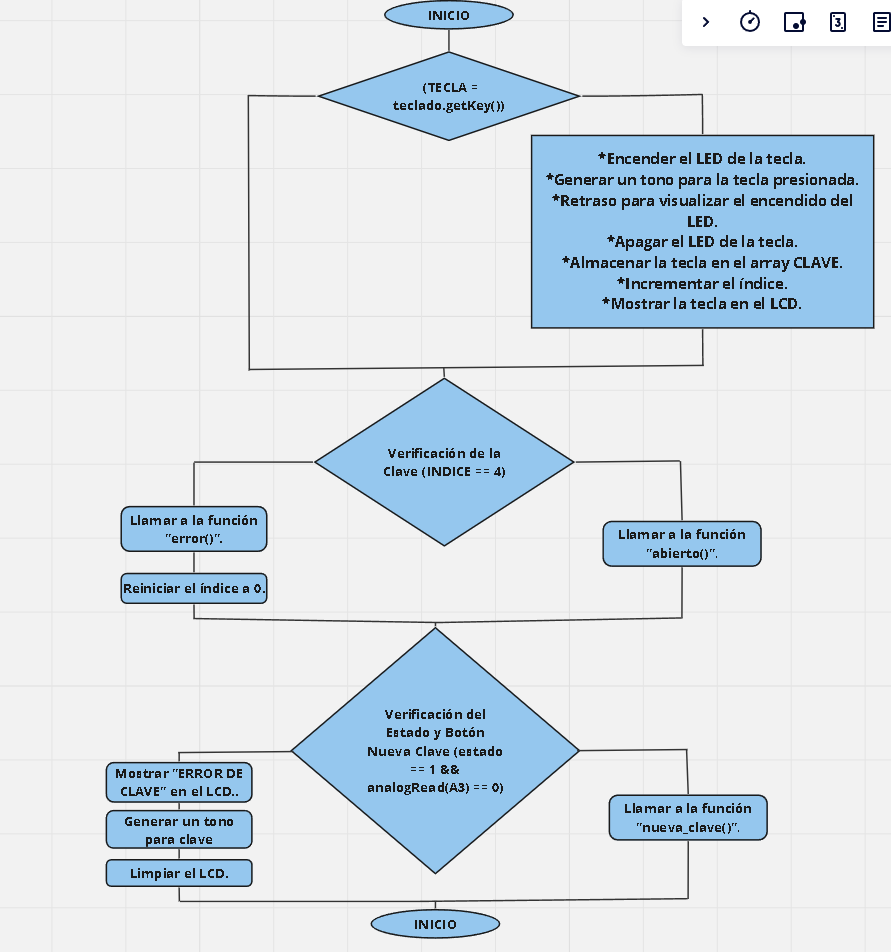
Observaciones y Limitaciones:

Es importante mencionar que este proyecto se basó en los conocimientos adquiridos durante el el año. Se exploraron conceptos como la comunicación serie, el manejo de sensores y la programación orientada a objetos. Aunque se reconocen áreas donde se podría mejorar en futuras versiones, se presenta una implementación inicial que demuestra la aplicación de habilidades y conocimientos adquiridos durante el curso.

La estructura del informe se organiza de manera clara y técnica, enfocándose en aspectos específicos del diseño y funcionamiento del sistema, evitando detalles innecesarios

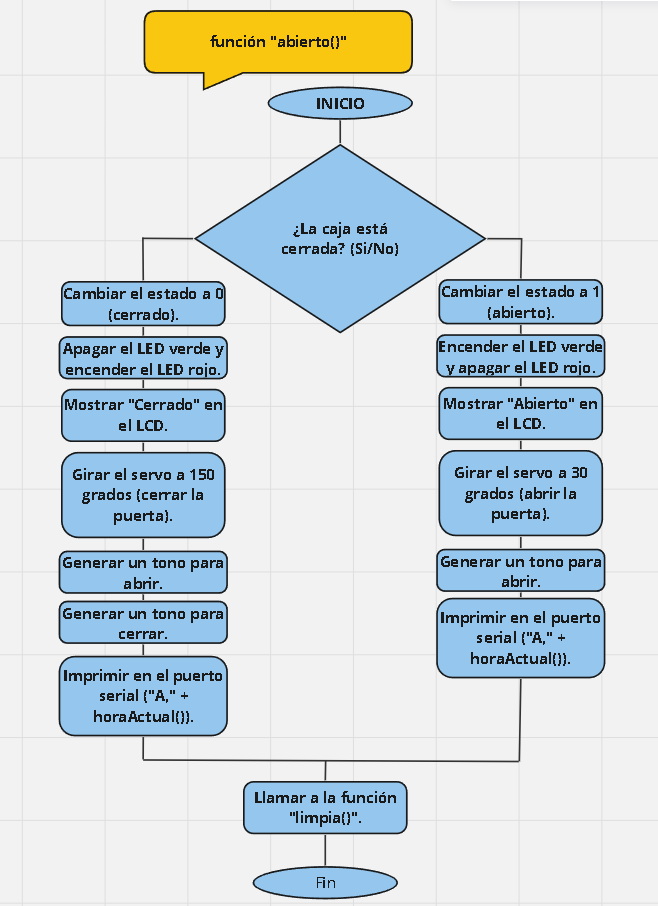
## 2. Diagrama funcional

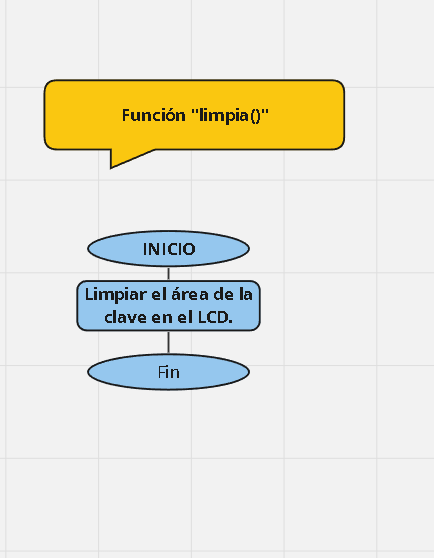
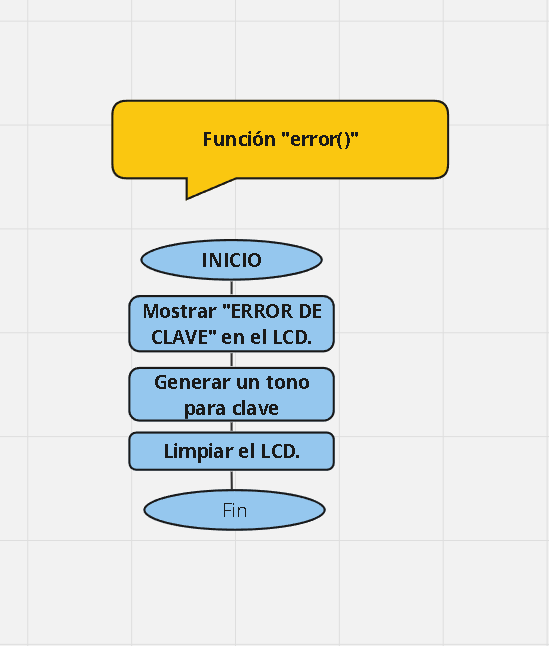
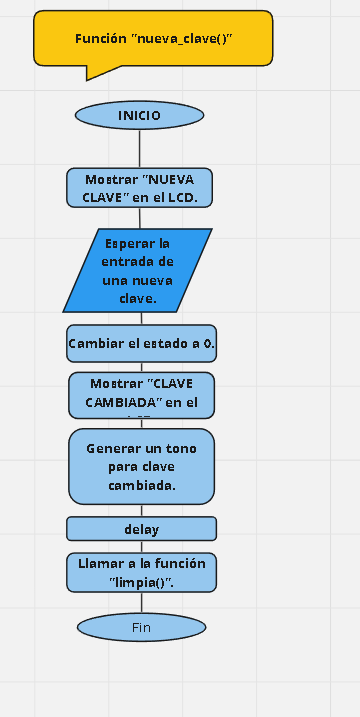
Diagrama de Arduino



la lectura del teclado es un bloque de código que se ejecuta repetidamente en el bucle principal (loop). La función teclado.getKey() es parte de la biblioteca Keypad y retorna la tecla presionada, o NO\_KEY si ninguna tecla está siendo presionada en ese momento.

En este caso, el código utiliza un if para verificar si TECLA tiene un valor (diferente de NO\_KEY). Si es el caso, se ejecutan una serie de acciones para procesar la tecla presionada. Sin embargo, el código no utiliza un bucle while específicamente para la lectura del teclado, ya que el bucle principal (loop) en Arduino ejecuta continuamente.

DIAGRAMA DE FUNCION “ABIERTO()”



## 3. Código fuente de los Programas

CODIGO ARDUINO

#include <LiquidCrystal\_I2C.h>

#include <Keypad.h>

#include <Servo.h>

#include <TimeLib.h>

int estado = 0; // 0=cerrado 1=abierto

Servo servo11; // Crea el objeto servo11 con las características de Servo

const byte FILAS = 4; // define el número de filas

const byte COLUMNAS = 4; // define el número de columnas

char keys[FILAS][COLUMNAS] = { // define la distribución de teclas

{'1', '2', '3', 'A'},

{'4', '5', '6', 'B'},

{'7', '8', '9', 'C'},

{'\*', '0', '#', 'D'}

};

byte pinesFilas[FILAS] = {9, 8, 7, 6}; // pines correspondientes a las filas

byte pinesColumnas[COLUMNAS] = {5, 4, 3, 2}; // pines correspondientes a las columnas

Keypad teclado = Keypad(makeKeymap(keys), pinesFilas, pinesColumnas, FILAS, COLUMNAS); // crea objeto teclado

char TECLA; // almacena la tecla presionada

char CLAVE[5]; // almacena en un array 4 dígitos ingresados

char CLAVE\_MAESTRA[5] = "0000"; // almacena en un array la contraseña inicial

byte INDICE = 0; // índice del array

LiquidCrystal\_I2C lcd(0x27, 16, 2); // dependiendo del fabricante del I2C, el código 0x27 puede cambiar a 0x3F, 0x20, 0x38

const int LED\_VERDE = 13; // Definir el pin del LED verde

const int LED\_ROJO = 12; // Definir el pin del LED rojo

const int LED\_TECLA = A2; // Definir el pin del LED para la tecla

const int SPEAKER\_PIN = A0; // Definir el pin del parlante

void setup() {

lcd.init(); // inicializa el LCD

lcd.backlight();

servo11.attach(11, 660, 1400); // Asocia el servo1 al pin 11, define el min y max del ancho del pulso

servo11.write(150); // Gira el servo a 150 grados Cierra la puerta

limpia();

pinMode(LED\_VERDE, OUTPUT); // Configurar el pin del LED verde como salida

digitalWrite(LED\_VERDE, LOW); // Apagar el LED verde al inicio

pinMode(LED\_ROJO, OUTPUT); // Configurar el pin del LED rojo como salida

digitalWrite(LED\_ROJO, LOW); // Apagar el LED rojo al inicio

pinMode(LED\_TECLA, OUTPUT); // Configurar el pin del LED de la tecla como salida

digitalWrite(LED\_TECLA, LOW); // Apagar el LED de la tecla al inicio

pinMode(SPEAKER\_PIN, OUTPUT); // Configurar el pin del parlante como salida

Serial.begin(9600); // Inicializar la comunicación serial

}

void loop() {

TECLA = teclado.getKey(); // obtiene tecla presionada y asigna a variable

if (TECLA) // comprueba que se haya presionado una tecla

{

digitalWrite(LED\_TECLA, HIGH); // Encender el LED de la tecla

tone(SPEAKER\_PIN, 1000, 100); // Tono para la tecla presionada

delay(100); // Retraso para visualizar el encendido del LED

digitalWrite(LED\_TECLA, LOW); // Apagar el LED de la tecla

CLAVE[INDICE] = TECLA; // almacena en array la tecla presionada

INDICE++; // incrementa índice en uno

lcd.print(TECLA); // envía al LCD la tecla presionada

}

if (INDICE == 4) // si ya se almacenaron los 4 dígitos

{

if (!strcmp(CLAVE, CLAVE\_MAESTRA)) // compara clave ingresada con clave maestra

abierto();

else

error();

INDICE = 0;

}

if (estado == 1 && (analogRead(A3) == 0)) // si está abierta y se pulsa el botón de Nueva Clave

nueva\_clave();

}

void error() {

lcd.setCursor(0, 1);

lcd.print("ERROR DE CLAVE ");

tone(SPEAKER\_PIN, 1500, 300); // Tono para clave incorrecta

limpia();

}

void abierto() {

if (estado == 0) {

estado = 1;

digitalWrite(LED\_VERDE, HIGH); // Encender el LED verde

digitalWrite(LED\_ROJO, LOW); // Apagar el LED rojo

lcd.setCursor(0, 1);

lcd.print("Abierto "); // imprime en el LCD que está abierta

servo11.write(30); // Gira el servo a 30 grados abre la puerta

tone(SPEAKER\_PIN, 2000, 200); // Tono para abrir

Serial.print("A,"); // A para abierto

Serial.println(horaActual()); // horaActual() es una función ficticia que deberías definir

} else {

estado = 0;

digitalWrite(LED\_VERDE, LOW); // Apagar el LED verde

digitalWrite(LED\_ROJO, HIGH); // Encender el LED rojo

lcd.setCursor(0, 1);

lcd.print("Cerrado "); // imprime en el LCD que está cerrada

servo11.write(150); // Gira el servo a 150 grados cierra la puerta

tone(SPEAKER\_PIN, 1000, 200); // Tono para cerrar

Serial.print("C,"); // C para cerrado

Serial.println(horaActual());

}

limpia();

}

void nueva\_clave() {

lcd.setCursor(0, 0);

lcd.print("NUEVA CLAVE: ");

lcd.setCursor(12, 0);

INDICE = 0;

while (INDICE <= 3) {

TECLA = teclado.getKey(); // obtiene tecla presionada y asigna a variable TECLA

if (TECLA) // comprueba que se haya presionado una tecla

{

CLAVE\_MAESTRA[INDICE] = TECLA; // almacena en array la tecla presionada

CLAVE[INDICE] = TECLA;

INDICE++; // incrementa índice en uno

lcd.print(TECLA); // envía a monitor serial la tecla presionada

}

}

estado = 0;

lcd.setCursor(0, 1);

lcd.print("CLAVE CAMBIADA");

tone(SPEAKER\_PIN, 2500, 300); // Tono para clave cambiada

delay(2000);

limpia();

}

void limpia() {

lcd.setCursor(0, 0);

lcd.print("CLAVE : ");

lcd.setCursor(9, 0);

}

String horaActual() {

return String(hour()) + ":" + String(minute()) + ":" + String(second());

}

CODIGO PROCESSING

/\*1. import processing.serial.\*;:

• Importa la biblioteca para manejar la comunicación serial en Processing.

Permite la lectura de datos desde un puerto serial.

2. import java.util.Calendar;:

• Importa la clase Calendar de Java,

que se utiliza para obtener la fecha y la hora actuales.

3. import java.util.ArrayList;:

• Importa la clase ArrayList de Java,

que proporciona una lista dinámica para almacenar datos.

es decir, estoy utilizando un tipo de estructura de datos llamada arrayList

que es parte del paquete java.util en java,utiliza array dinamico

para almacenar elementos

\*/

import processing.serial.\*;

import java.util.Calendar;

import java.util.ArrayList;

/\*la clase serial es proporcionada para manejar la comuncacion serie

y en base a eso, se crea una variable, objeto que es la instancia de esa clase

la llamaremos puertoSerie\*/

Serial puertoSerie;

//clase .objeto

/\*la clase es un molde,plantilla \*/

/\*objeto es un conjunto de atributos(caracteristicas,privado)

y metodos(acciones,que realiza,publico),se le pone nombre, ej cajafuerte1

podemos crear miles de objetos en la misma clase, ej, caballo = c1;caballo = c2;\*/

/\*constructor, nos ayuda a inicializar nuestros atributos,

tiene el mismo nombre de la clase\*/

/\*en la clase CajaFuerte, la creamos nosotros, y ademas creamos la variable objeto de esa

clase que le asignaremos por nombre cajaFuerte\*/

CajaFuerte cajaFuerte;

//clase . objeto

void setup() {

size(1280, 750);

String puerto = "COM4"; // Cambia esto al puerto correcto

// Inicializar la instancia de la clase puertoSerie

puertoSerie = new Serial(this, puerto, 9600);

// Inicializar la instancia de la clase CajaFuerte

cajaFuerte = new CajaFuerte(width / 2, height / 2);

//objeto

}

void draw() {

background(200); // Color gris

// Dibujar la caja fuerte y la puerta

cajaFuerte.mostrar();

//objeto.metodo mostrar

// Leer información del puerto serie

/\*bucle while se ejecutará mientras haya datos disponibles en el puerto serie

para ser leídos puertoSerie.available() > 0).\*/

while (puertoSerie.available() > 0) {

String[] datos = split(puertoSerie.readStringUntil('\n'), ',');

/\*Lee una línea completa desde el puerto serie hasta encontrar un carácter de

nueva línea ('\n').

divide la línea leída en un array de strings

utilizando la coma como delimitador.

El resultado es un array llamado datos.ej: ecibes "A,123" datos[1] sería "123". \*/

if (datos.length == 2) {

/\*Verifica si el array datos tiene exactamente dos elementos.

Esto es importante porque estás esperando que los datos tengan un formato específico

(por ejemplo, "A,123").\*/

String estadoCaja = datos[0];

/\*Asigna el primer elemento del array datos a la variable estadoCaja.

En nuestro ejemplo anterior, si datos[0] es "A",

entonces estadoCaja sería igual a "A".\*/

if (estadoCaja.equals("A")) {

/\*Compara el valor de estadoCaja con "A".

Si son iguales, significa que la caja debe abrirse,

por lo que se llama al método abrirCaja() de la instancia cajaFuerte.\*/

cajaFuerte.abrirCaja();

} else if (estadoCaja.equals("C")) {

/\*Si no es igual a "A", verifica si es igual a "C".

Si sí, significa que la caja debe cerrarse,

así que se llama al método cerrarCaja() de la instancia cajaFuerte.

Esta parte del código interpreta el estado de la caja

(si está abierta o cerrada) basándose en los datos recibidos desde Arduino

y toma acciones en consecuencia.\*/

cajaFuerte.cerrarCaja();

}

}

}

// Muestra la cantidad de veces que la caja fue abierta y cerrada

fill(80);

/\* se utiliza para establecer el color de relleno

para las formas y texto que se dibujarán a continuación.\*/

textSize(18);

/\*se utiliza para establecer el tamaño del texto que se dibujará a continuación.\*/

text("Aperturas: " + cajaFuerte.getContadorAperturas(), 100, 100);

/\*Esto obtiene la cantidad de aperturas de la instancia cajaFuerte.\*/

text("Cierres: " + cajaFuerte.getContadorCierres(), 100, 150);

/\*Esto obtiene la cantidad de cierre de la instancia cajaFuerte.\*/

// Muestra el registro completo de aperturas y cierres

/\*

Esta parte del código se encarga de mostrar

en la ventana de dibujo de Processing el registro completo de

aperturas y cierres de la caja fuerte.

Vamos a analizar cada línea:\*/

fill(80);

textSize(14);

text("Registro:", 100, 200);

ArrayList<String> registro = cajaFuerte.getRegistro();

/\*cajaFuerte.getRegistro() llama al método getRegistro()

de la instancia cajaFuerte, que devuelve un ArrayList de strings

que contiene el registro completo de aperturas y cierres de la caja fuerte.

Este ArrayList se almacena en la variable registro.\*/

for (int i = 0; i < registro.size(); i++) {

/\*inicia un bucle for que recorre cada elemento del ArrayList registro.

La variable i es el índice del elemento actual en el bucle.\*/

text(registro.get(i), 100, 230 + i \* 20);

/\*registro.get(i) obtiene el elemento actual del ArrayList registro

en la posición i.

text() se utiliza para mostrar este elemento en la ventana de dibujo.\*/

}

}

// Clase CajaFuerte

class CajaFuerte {

private float x, y;

private boolean abierta;

private int contadorAperturas;

private int contadorCierres;

private ArrayList<String> registro;

// Elementos para el cartel

private String estadoCartel;

private int colorCartel;

// Constructor

CajaFuerte(float x, float y) {

this.x = x;

this.y = y;

this.abierta = false;

this.contadorAperturas = 0;

this.contadorCierres = 0;

this.registro = new ArrayList<String>();

this.estadoCartel = "Cerrado";

this.colorCartel = color(255, 0, 0); // Rojo por defecto

}

// Función para abrir la caja fuerte

void abrirCaja() {

if (!abierta) {

abierta = true;

contadorAperturas++;

estadoCartel = "Abierto";

colorCartel = color(0, 255, 0); // Cambiar a verde cuando está abierto

String registroApertura = "Apertura #" + contadorAperturas + ": " + obtenerHoraFechaActual();

registro.add(registroApertura);

guardarRegistro();

}

}

// Función para cerrar la caja fuerte

void cerrarCaja() {

if (abierta) {

abierta = false;

contadorCierres++;

estadoCartel = "Cerrado";

colorCartel = color(255, 0, 0); // Cambiar a rojo cuando está cerrado

String registroCierre = "Cierre #" + contadorCierres + ": " + obtenerHoraFechaActual();

registro.add(registroCierre);

guardarRegistro();

}

}

// Función para mostrar la caja fuerte y la puerta

void mostrar() {

// Dibujar la caja fuerte

fill(255, 220, 185); // Color beige

rectMode(CENTER);

rect(x, y, 200, 150); // Caja fuerte

// Dibujar la puerta

fill(abierta ? 0 : 255, 0, 0); // Color verde si está abierta, rojo si está cerrada

rect(x + 90, y, 10, 90); // Puerta

// Dibujar el cartel en el centro de la caja

fill(colorCartel);

textSize(16);

textAlign(CENTER, CENTER);

text(estadoCartel, x, y);

}

// Función para obtener la hora y fecha actual

String obtenerHoraFechaActual() {

Calendar calendario = Calendar.getInstance();

int hora = calendario.get(Calendar.HOUR\_OF\_DAY);

int minuto = calendario.get(Calendar.MINUTE);

int segundo = calendario.get(Calendar.SECOND);

int dia = calendario.get(Calendar.DAY\_OF\_MONTH);

int mes = calendario.get(Calendar.MONTH) + 1; // Nota: en Java, los meses comienzan desde 0

int anio = calendario.get(Calendar.YEAR);

// Formatear la hora y fecha

return nf(hora, 2) + ":" + nf(minuto, 2) + ":" + nf(segundo, 2) + " " + nf(dia, 2) + "/" + nf(mes, 2) + "/" + anio;

}

// Función para guardar el registro completo en un archivo en el escritorio

void guardarRegistro() {

String[] lineas = new String[registro.size()];

/\*registro es una instancia de la clase ArrayList.

En Java, ArrayList es una estructura de datos dinámica

que puede cambiar de tamaño dinámicamente según sea necesario.

El método size() de ArrayList devuelve la cantidad de elementos

almacenados en el ArrayList.

registro.size() se utiliza aquí para determinar el tamaño del array\*/

// Crea un array de strings para almacenar cada línea del registro

for (int i = 0; i < registro.size(); i++) {

// Llena el array con cada elemento del registro

lineas[i] = registro.get(i);

}

// Obtiene la ruta del escritorio del usuario

String rutaEscritorio = System.getProperty("user.home") + "/Desktop/";

// Define el nombre del archivo de texto a guardar

String nombreArchivo = "registro\_caja.txt";

saveStrings(rutaEscritorio + nombreArchivo, lineas);

// Guarda el array de strings (registro) como un archivo de texto en el escritorio

}

// Funciones para obtener la cantidad de aperturas, cierres y el registro

int getContadorAperturas() {

return contadorAperturas;

}

int getContadorCierres() {

return contadorCierres;

}

ArrayList<String> getRegistro() {

return registro;

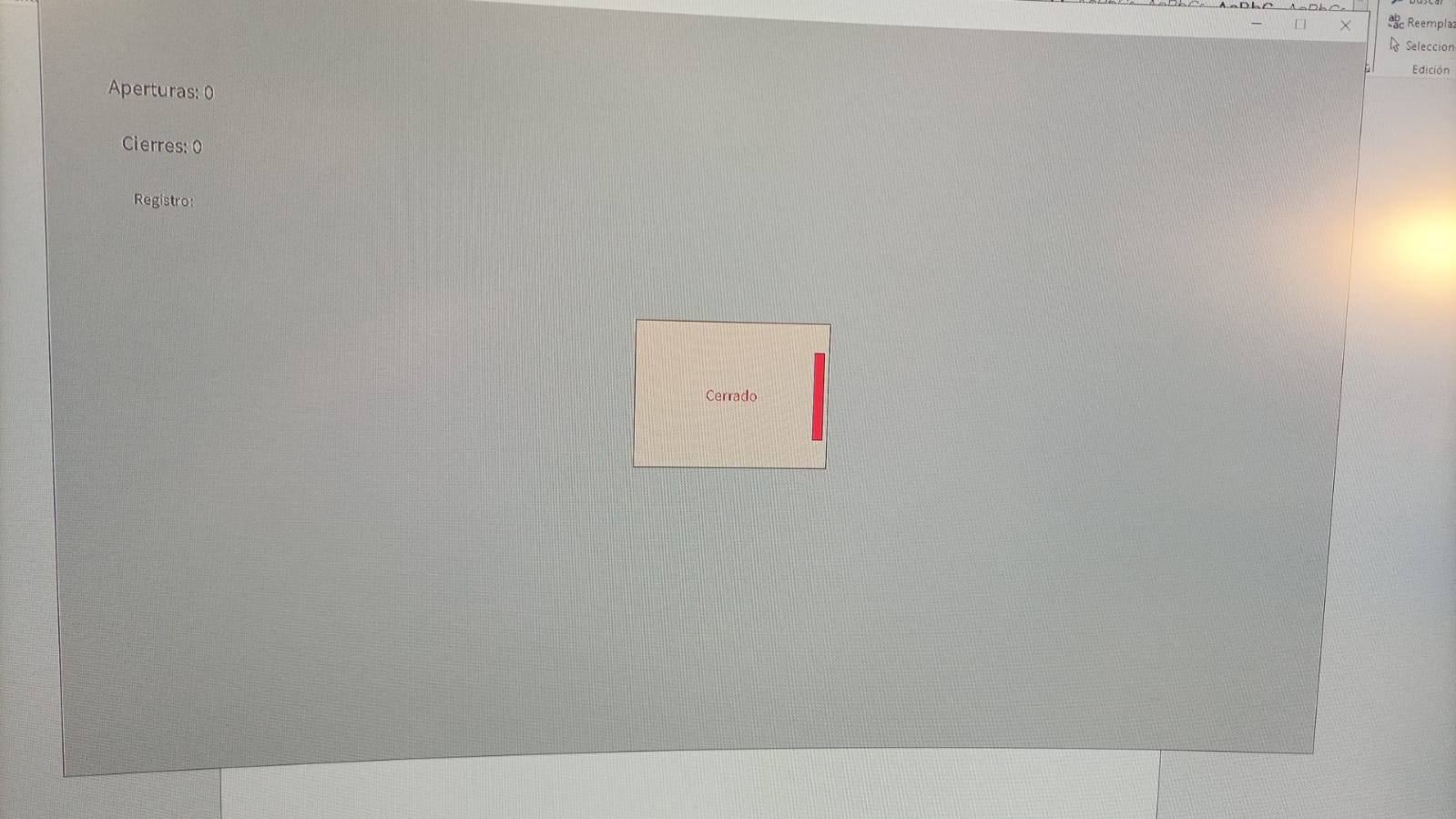
}

}

## 4. Salidas del Programa



En esta sección de la imagen, podemos visualizar que la caja fuerte a iniciado, es el momento en el que podemos ingresar la clave, por default la clave es “0000”.



Y ejecutamos el processing, vemos la interfaz de la misma, viendo la ventana tenemos en el centro una caja fuerte dibujada por processing, que dice cerrado, del lado izquuierdo tenemos información de cuantas veces se abrió, y abajo tenemos un registro de la caja de cuantas veces se cerro. Incluido la fecha y hora en que se abrió y cerro.



Vemos que la puerta esta cerrada por el servo motor



Ahora al ingresar la clave, se prenden los leds amarillo y verde indicando que la caja esta abierta junto con el display que también lo dice

Si visualizamos en processing vemos que en lugar de decir cerrado, ahora dice abierto, y además tenemos un registro de que se abrió 1 vez, y la fecha y hora de apertura de la caja que posteriormente se guardara en un archivo.txt

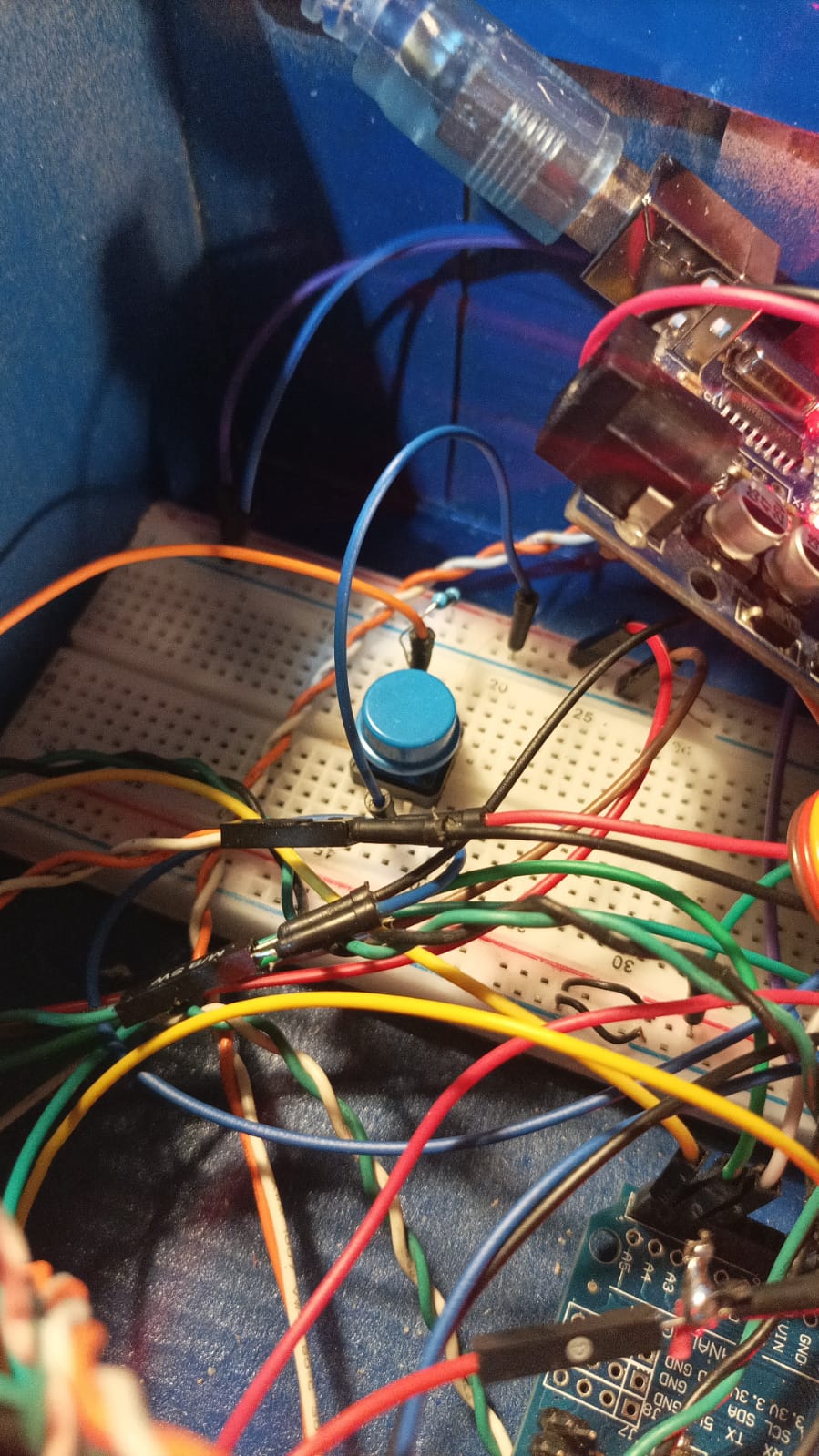


Vemos en esta parte, que el servo motor se ah desplazado unos 150 grados permitiendo abrir la puerta de la caja fuerte

## 5. Anexos

## 

Esto guarda un registro que posteriormente se guarda en un archivo.txt

Además al abrir la caja tenemos un botón que nos permite cambiar la clave 

Al presionar ese botón nos aparece en el display NUEVA CLAVE:



Aclaración, solo funciona cuando la puerta de la caja fuerte esta abierta



Ahora cambiamos la clave que antes era “0000” por la siguiente “1234”

Aclaración, la clave siempre funcionara mientras el Arduino permanesca prendido, una vez que se apaga, la clave se restea, por la default que es la “0000”

### 5. 1 Esquemático



### 5.2 Listado de componentes

### \*un Arduino

\*5 leds

\*varios cables para conectar los componentes

\*un parlante indicador

\*un teclado keypad

\*un botón

\*un display

\*un servo motor

### 5.3 Referencias

Este es el video principal en el que me inspire

<https://www.youtube.com/watch?v=j1ThOHUC8aw>

este es el video secundario que me ayudo a ver y entender mas del proyecto

<https://www.youtube.com/watch?v=Ah-vMwLTVmI>

Cite cualquier bibliografía utilizada.

### 5.4 Otros anexos que deba incluir



Ese es el circuito principal del proyecto, lo único que se le añadió es el parlante indicador, y los leds

El proyecto tiene como propósito hacer una caja fuerte electrónica, funcional por medio del microcontrolador Arduino, en este caso es el uno.