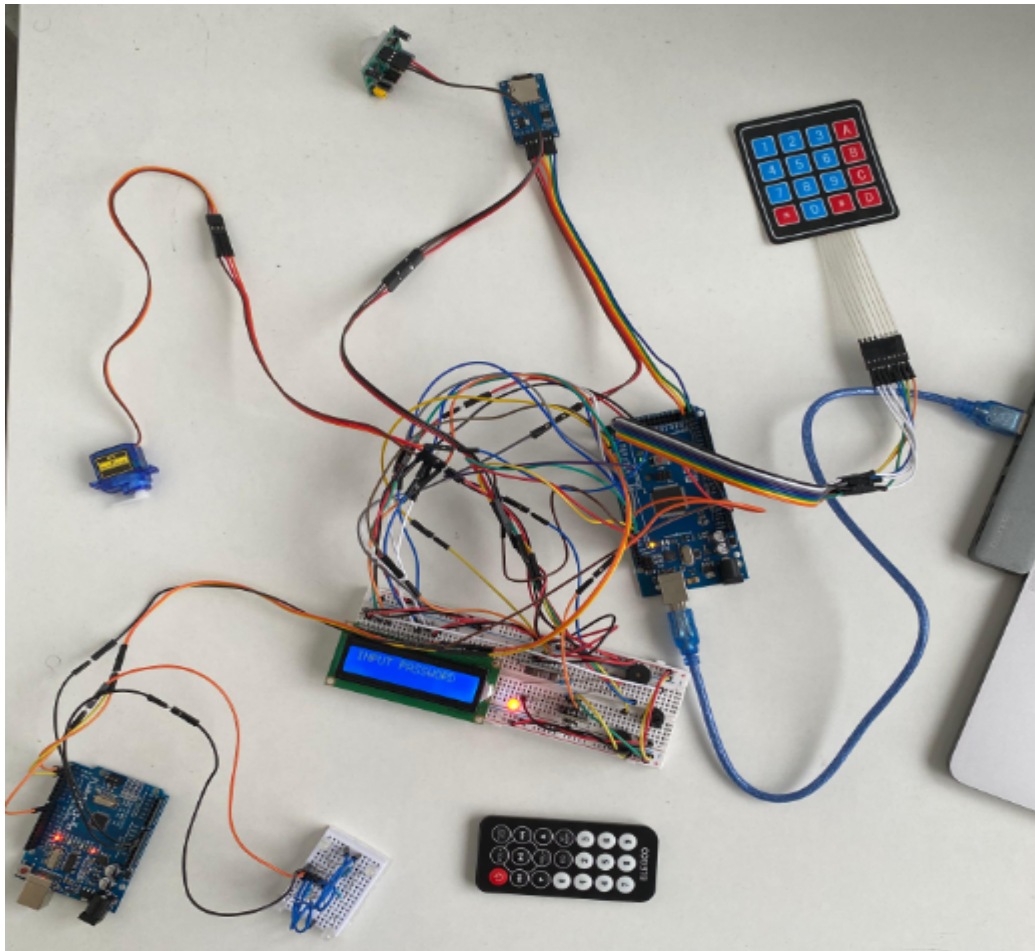


# Sistema Automático de Controlo e Gestão de Pavilhão



**FRANCISCO SOUSA – a22301231**

**ANDRÉ MARQUES - a22207598**

**GONÇALO BARATA - a22205060**

**A.A.C | LEI | 23/11/2023**

# Índice

<i>Introdução .....</i>	<b>3</b>
<i>Abordagem Metodológica .....</i>	<b>3</b>
<i>Objetivo do Projeto .....</i>	<b>4</b>
<i>Componentes Utilizados.....</i>	<b>4</b>
<i>Código Master .....</i>	<b>9</b>
<i>Código Slave .....</i>	<b>19</b>
<i>Problemas Ocorridos.....</i>	<b>20</b>
<i>Conclusão .....</i>	<b>21</b>

# Introdução

Para o nosso projeto decidimos desenvolver um Sistema Automático de Controlo e Gestão de um Pavilhão, explorando os conceitos adquiridos na unidade curricular.

Integramos um Arduino Mega e um Arduino Uno R3 com diversos sensores e componentes para criar um sistema inteligente que proporciona uma gestão automatizada e otimizada de um pavilhão. Ao longo deste projeto debatemo-nos com alguns problemas mas todos eles contribuíram para a nossa aprendizagem e para o resultado final deste projeto.

## Abordagem Metodológica

Na execução do nosso Sistema, adotaremos uma abordagem metodológica focada na aplicação prática de vários conhecimentos abordados pelo professor, visando a integração de vários componentes eletrónicos. A implementação será estruturada em etapas lógicas, cada uma destinada a configurar e otimizar as funcionalidades específicas do sistema.

Inicialmente, focamo-nos na configuração do LCD, utilizando o potenciômetro para ajustar a intensidade do display e o LED indicador para sinalizar o estado do sistema. Posteriormente, dedicaremos atenção ao receiver, permitindo o controlo remoto fazer as operações do pavilhão. Temos um buzzer ativo que emite sons distintivos e um buzzer passivo, com uma variedade de sons, que acrescentarão partes auditivas bastante importantes para a comunicação do sistema.

A integração do RTC será crucial para manter uma sincronização precisa da data e hora dos 2 users no pavilhão, o sensor de movimento será responsável por detetar o movimento dos users. O microSD, tem um ficheiro importante para o funcionamento do sistema que será implementado para assegurar o armazenamento dos vários dados.

No Arduino Slave, quatro LEDs estarão interligados que serão as luzes do teto. A implementação de um servo será destinada à abertura e fecho da porta do pavilhão. Além

disso, a matriz de teclado 4x4 será utilizada para inserir códigos específicos e terá mais funcionalidades dependendo das teclas pressionadas, contribuindo para a segurança e controlo de acesso ao pavilhão.

Ao fazermos isto, realizaremos a implementação de um Sistema Automático de Controlo e Gestão de Pavilhão bastante funcional e inteligente, que nos fará consolidar várias técnicas práticas no âmbito da disciplina.

## Objetivo do Projeto

O nosso Sistema Automático de Controlo e Gestão de um Pavilhão foi criado para tornar a gestão de um espaço grande mais fácil, segura e inteligente. Nós imaginamos um local onde a entrada e saída são feitas de forma automática, sem precisar de qualquer tipo de chaves. O sistema utiliza um teclado 4x4 simples para inserção de códigos, um LCD que mostra informações e até mesmo um comando à distância remoto para controlar as luzes.

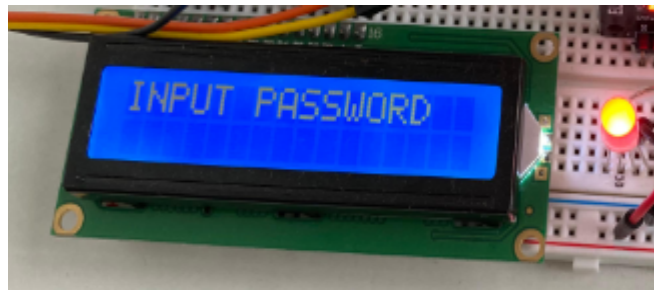
A principal ideia é simplificar o acesso e melhorar a segurança do local. Quando alguém insere o código certo no teclado, a porta abre-se automaticamente, e o sistema mostra uma mensagem indicando que a entrada foi autorizada. Além disso, podemos personalizar algumas coisas, como ligar ou desligar as luzes do teto com o comando remoto, ver o nome da pessoa que entrou e até mesmo controlar outras funcionalidades do espaço.

O sistema não é apenas sobre automação, também guarda informações importantes sobre quando as pessoas entram ou saem, criando um histórico num ficheiro. Isto é como ter um assistente tecnológico que facilita o controle do espaço de maneira inteligente, proporcionando mais segurança.

## Componentes Utilizados

### **LCD:**

Inicialmente solicita introdução de o código de autorização de cada user. Tem um LED vermelho incorporado que indica que o sistema está fechado, e ao inserir a senha correta, o LED muda para verde, indicando que o sistema foi aberto com sucesso.



## Receiver para o controlo remoto:

O receiver com um controlo remoto permite a interação através da seleção de números no comando. Ao pressionar os botões correspondentes no comando remoto, é possível enviar informações para o sistema. O comando remoto, também controla as luzes dos 4 LEDs, com a opção de piscar as luzes se mantermos pressionado qualquer botão do controle remoto e liga/desliga os leds se pressionarmos qualquer botão durante 1 ou 2 segundos.



## Buzzer Ativo:

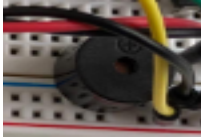
O buzzer ativo faz apenas faz 1 som.

## Buzzer Passivo:

O buzzer passivo faz vários sons.

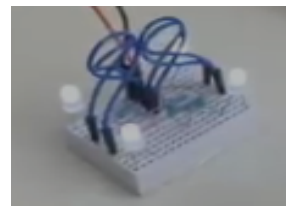
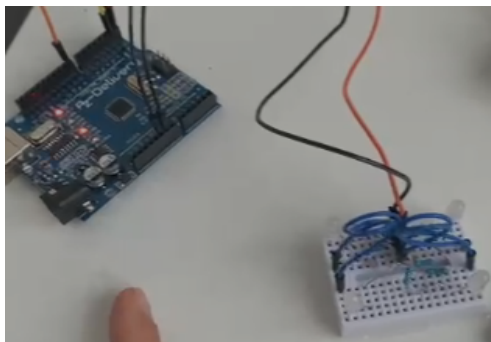
## Potenciômetro:

Muda intensidade do Display.



## Arduino Slave ligado a 4 LEDs:

O Arduino está conectado a quatro LEDs, os quais simbolizam as luzes do teto. O comportamento dos LEDs é controlado pelo Arduino, permitindo que as luzes sejam ligadas ou desligadas conforme necessário.



## Servo:

O servo serve para a abertura da porta.



## Sensor de Movimento:

Este sensor deteta quando houve movimento e regista a data e a hora de quando o user passou pelo sensor. Este sensor é sempre ativados 1 minutos depois de o sistema ser reiniciado.

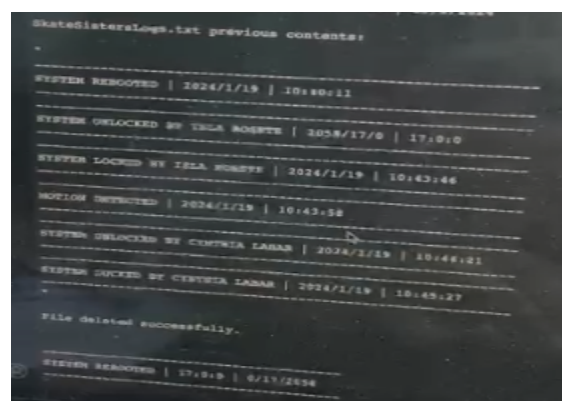


## MicroSD:

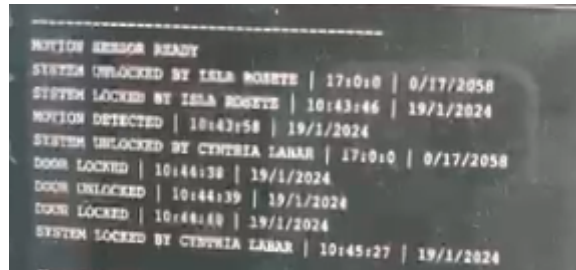


Tem um ficheiro que regista tudo e podemos dar reset ao sistema sempre que for necessário para sabermos as informações que foram registadas. Depois de o sistema ter dado reset temos as informações todas que foram registadas e de seguida apaga as informações que foram recebidas antes do reset. Posteriormente está tudo pronto para serem guardadas mais informações no ficheiro.

Este é o formato do ficheiro após o reset do sistema, onde regista todas as informações:



Este é o formato do ficheiro no momento antes do reset do sistema ser, quando basicamente está a registar todas as informações:



```
NOTION SENSOR READY
SYSTEM UNLOCKED BY ISLA ROBERTS | 17:0:0 | 0/17/2058
SYSTEM LOCKED BY ISLA ROBERTS | 10:43:46 | 19/1/2024
MOTION DETECTED | 10:43:58 | 19/1/2024
SYSTEM UNLOCKED BY CYNTHIA LAMAR | 17:0:0 | 0/17/2058
DOOR LOCKED | 10:44:38 | 19/1/2024
DOOR UNLOCKED | 10:44:39 | 19/1/2024
DOOR LOCKED | 10:44:40 | 19/1/2024
SYSTEM LOCKED BY CYNTHIA LAMAR | 10:45:27 | 19/1/2024
```

## Teclado Matriz 4x4:

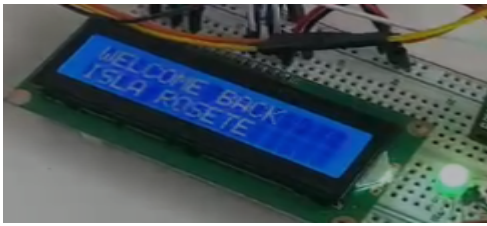
O teclado matriz é integrado no sistema, aceitando dois códigos distintos para os 2 usuários. Este teclado oferece funcionalidades avançadas, como a capacidade de apagar caracteres individualmente ou em série com a tecla "D", permitindo depois escrita no mesmo local. O botão "C" realiza um clear em todo o código. Ao inserir o código correto, uma mensagem é exibida no display LCD, o LED associado ao display muda para verde, indicando a abertura bem-sucedida do sistema, e o servo é ativado para abrir a porta.

Após a aceitação do user, as informações, incluindo data e hora, são registadas no ficheiro do sistema. Os quatro LEDs conectados ao Arduino Slave são ativados, indicando que as luzes do teto estão ligadas. Depois do user ser aceite, o teclado possui funções adicionais, como a exibição do nome do usuário ao pressionar "C", a ativação/desativação dos quatro LEDs que simbolizam as luzes do teto ao pressionar "A", e a abertura/fechamento da porta com a ajuda do servo ao pressionar "B". Se o código estiver incorreto, o acesso é negado.

Estas funcionalidades fornecem um sistema seguro e inovador para o controle e gestão de qualquer pavilhão.







## RTC (Real Time Clock):

Desempenha um papel extremamente importante que regista precisamente a data e a hora sempre que um utilizador é detetado



## Código Master

```
#include <Keypad.h>
#include <LiquidCrystal.h>
#include <Servo.h>
#include <Wire.h>
#include <RTClib.h>
#include <SdFat.h>
#include <SPI.h>
#include <IRremote.h>

#define Password_Lenght 7 // Give enough room for six chars + NULL char
#define MISO_SD 50 // MISO pin = SDI
#define MOSI_SD 51 // MOSI pin = SD0
#define SCK_SD 52 // Serial Clock pin
#define SD_CS 53 // Chip select pin
#define BUZZER 10
#define BUZZER_PASSIVO 30
#define RED 13
#define GREEN 11
```

[illegible]

```

// Open the file for reading
if (file.open(FILE_NAME, O_READ)) {
    // While file.available() returns a value greater than 0, there are still
    characters to read
    while (file.available()) {
        // Read and print one character at a time
        Serial.write(file.read());
    }
    // Close the file once done
    file.close();
} else {
    // If the file failed to open, print an error message
    Serial.println("Error opening the file.");
}
}

// SD CARD

TESTING //
bool deleteFile(const char* fileName) {
    if (sd.remove(fileName)) {
        Serial.println("File deleted successfully.");
        return true;
    } else {
        Serial.println("Error deleting file.");
        return false;
    }
}

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

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

    Wire.begin();
    Serial.begin(9600);
    Serial1.begin(9600);

    rtc.begin();
    //rtc.adjust(DateTime(2024, 2, 19, 20, 40, 55));

    lcd.begin(16, 2);
    pinMode(BUZZER, OUTPUT);
    pinMode(BUZZER_PASSIVO, OUTPUT);
    pinMode(RED, OUTPUT);
    pinMode(GREEN, OUTPUT);
    pinMode(MOTION, INPUT);
    pinMode(SD_CS, OUTPUT);
    digitalWrite(BUZZER, LOW);
    digitalWrite(MOTION, LOW);
    digitalWrite(RED, HIGH);
    myservo.attach(SERV0);

```

```

myservo.write(0); // Move servo to locked position -> 0°
irrecv.enableIRIn();
delay(1000);

if (!sd.begin(SD_CS, SPI_HALF_SPEED)) {
    Serial.println("SD card initialization failed.");
    return;
}

Serial.println();
Serial.print(FILE_NAME);
Serial.println(" previous contents:");
Serial.println();

Serial.println('');
delay(200);
printFileContent();
Serial.println('');
Serial.println();

delay(200);
deleteFile(FILE_NAME);
char messageTemp[] = "SYSTEM REBOOTED";
saveDataSdCard(messageTemp);

Serial.println();
Serial.println("");
Serial.println("-----");
printTime("SYSTEM REBOOTED");
Serial.println("-----");
}

void loop() {
    // put your main code here, to run repeatedly:
    if (!motionSensorIsReady) {
        unsigned long currentMillis = millis();
        if (currentMillis > 60000) {
            Serial.println("MOTION SENSOR READY");
            motionSensorIsReady = true;
        }
    }
    if(motionSensorIsReady)
    {
        checksForMotion(&systemIsLocked, &activeUser , &ledState);
    }
    checksIfSystemOpens(&systemIsLocked, &activeUser, &ledState);
    checksIfSystemCloses(&systemIsLocked, &activeUser, &ledState);
}

char keyInput() // Gets the keypad input

```

```

{
    return customKeypad.getKey();
}

void locksDoor() // Locks the door
{
    myservo.write(0);
}

void unlocksDoor() // Unlocks the door
{
    myservo.write(180);
}

void locksAndUnlocks()
{
    if (myservo.read() == 0) {
        unlocksDoor();
        printsTime("DOOR UNLOCKED");
    } else {
        locksDoor();
        printsTime("DOOR LOCKED");
    }
}

void togglesLights() // Reverses ceiling Lights output
{
    if (ledState == 0) {
        Serial1.println("LED:ON");
    } else {
        Serial1.println("LED:OFF");
    }
}

void activatesBuzzer(int duration) // Turns the buzzer on
{
    digitalWrite(BUZZER,HIGH);
    delay(duration);
    if (duration == 0)
    {
        return;
    }
    deactivatesBuzzer();
}

void deactivatesBuzzer() // Turns the buzzer off
{
    digitalWrite(BUZZER,LOW);
}

void openingMelody() {

```

```

    for (int thisNote = 0; thisNote < 4; thisNote++) {
        playTone(melody[thisNote], 500);
    }
}

void closingMelody() {
    for (int thisNote = sizeof(melody) / sizeof(melody[0]) - 1; thisNote >= 0;
thisNote--) {
        playTone(melody[thisNote], 500);
    }
}

void playTone(int tone, int duration) {
    long tonePeriod = 1000000L / tone;
    long pulseWidth = tonePeriod / 2;

    for (long i = 0; i < duration * 1000L / tonePeriod; i++) {
        digitalWrite(BUZZER_PASSIVO, HIGH);
        delayMicroseconds(pulseWidth);
        digitalWrite(BUZZER_PASSIVO, LOW);
        delayMicroseconds(pulseWidth);
    }
    delay(20);
}

void printsLcd(char message[17], int row, int column) // Prints some message
on the LCD
{
    lcd.setCursor(row, column);
    lcd.print(message);
}

void printsTime(String message)
{
    DateTime now = rtc.now();    // Read the current time from the RTC

    Serial.print(message);
    Serial.print(" | ");
    Serial.print(now.hour(), DEC);
    Serial.print(':');
    Serial.print(now.minute(), DEC);
    Serial.print(':');
    Serial.print(now.second(), DEC);
    Serial.print(" | ");
    Serial.print(now.day(), DEC);
    Serial.print('/');
    Serial.print(now.month(), DEC);
    Serial.print('/');
    Serial.print(now.year(), DEC);
    Serial.println();
}

```

```

void clearData() // Function to clear the key String
{
    while (data_count != 0)
    { // This can be used for any array size,
        Data[data_count--] = 0; //clear array for new data
    }
    return;
}

void checksForMotion(bool * systemIsLocked, int * activeUser , int * ledState)
// Checks motion for the motion sensor
{
    int alarmFlag = 0;
    while (digitalRead(MOTION) == 1 && *systemIsLocked == true)
    {
        Serial1.println("LED:ON");
        if(alarmFlag == 0)
        {
            printsTime("MOTION DETECTED");
            alarmFlag = 1;
            savesDataSdCard("MOTION DETECTED");
        }
        checksClosedCode( systemIsLocked, activeUser , ledState, -1);
    }
    if (*systemIsLocked)
    {
        Serial1.println("LED:OFF");
    }
}

void checksOpenedCode(bool * systemIsLocked, int * activeUser , int *
ledState, int functionUser)
{
    customKey = keyInput();
    functionUser = *activeUser;

    if (customKey == 'A')
    {

        togglesLights();
        *ledState = !(*ledState);

    }
    if (irrecv.decode(&results)) {
        //Serial.println(results.value, HEX);
        if (results.value == 0xFFFFFFFF) {
            togglesLights();
            *ledState = !(*ledState);
            delay(1000);
        }
    }
}

```

```

    }
    irrecv.resume(); // Receive the next value.
}
if (customKey == 'B')
{
    delay(500);
    locksAndUnlocks();
    delay(500);
}
if (customKey == 'C')
{

    lcd.clear();
    printsLcd("GUEST USER:",0,0);
    printsLcd(users[functionUser],0,1);
    delay(3000);
    lcd.clear();
    printsLcd("INPUT PASSWORD",0,0);
}
if (customKey == '#')
{
    *systemIsLocked = true;
    Serial.print("SYSTEM LOCKED BY ");
    printsTime(users[functionUser]);

    strcpy(messageTemp,"SYSTEM LOCKED BY ");
    strcat(messageTemp,users[functionUser]);
    savesDataSdCard(messageTemp);

    lcd.clear();
    printsLcd("SEE YOU SOON",0,0);
    printsLcd(users[functionUser],0,1);
    digitalWrite(MOTION,HIGH);
    digitalWrite (RED, HIGH);
    digitalWrite (GREEN, LOW);
    togglesLights();
    *ledState = !(*ledState);
    locksDoor();
    lcd.clear();
    clearData();
    printsLcd("INPUT PASSWORD",0, 0);
    closingMelody();
    lcd.clear();
}
}

void checksClosedCode(bool * systemIsLocked, int * activeUser , int *
ledState, int functionUser) // When the system is closed
{

    customKey = keyInput();

```



```

    if (customKey == 'D')
    {
        data_count--;
        lcd.setCursor(data_count, 1); // move cursor to show each new char
        lcd.print(" "); // print char at said cursor
        activatesBuzzer(500);
        lcd.setCursor(data_count, 1);
        lcd.print("*");
    }
    else if(customKey == 'C')
    {
        printsLcd("      ",0,1);
        activatesBuzzer(500);
        printsLcd("*****",0,1);    //To hide your PASSWORD, make sure its the
same lenght as your password
        clearData();
        printsLcd("*",0,1);
        data_count = 0; // decrease data array by 1 to store new char, also keep
track of the number of chars entered

    }
    else if (customKey) // makes sure a key is actually pressed, equal to
(customKey != NO_KEY)
    {
        printsLcd("*****",0,1);    //To hide your PASSWORD, make sure its the
same lenght as your password
        Data[data_count] = customKey; // store char into data array
        lcd.setCursor(data_count, 1); // move cursor to show each new char
        lcd.print(Data[data_count]); // print char at said cursor
        data_count++; // increment data array by 1 to store new char, also keep
track of the number of chars entered
        activatesBuzzer(200);
    }

    if (data_count == Password_Lenght - 1) // if the array index is equal to the
number of expected chars, compare data to master
    {
        if (!strcmp(Data, master0) || !strcmp(Data, master1)) // equal to
(strcmp(Data, Master) == 0)
        {
            Serial.print("SYSTEM UNLOCKED BY ");

            if (!strcmp(Data, master0))
            {
                *activeUser = 0;
            }
            if (!strcmp(Data, master1))
            {
                *activeUser = 1;
            }
        }
    }

```

```

    }
    functionUser = *activeUser;

    strcpy(messageTemp,"SYSTEM UNLOCKED BY ");
    strcat(messageTemp,users[functionUser]);
    savesDataSdCard(messageTemp);
    printsTime(users[functionUser]);
    lcd.clear();
    printsLcd("WELCOME BACK",0,0);
    printsLcd(users[functionUser],0,1);
    digitalWrite(MOTION,LOW);
    digitalWrite (RED, LOW);
    digitalWrite (GREEN, HIGH);
    Serial1.println("LED:ON");
    *ledState = 1;
    unlocksDoor();
    openingMelody();
    delay(3000);

    lcd.clear();
    printsLcd("SYSTEM OPENED",0,0);
    printsLcd("press # to close",0,1);
    *systemIsLocked = false;
    }
    else
    {
        lcd.clear();
        printsLcd("ACCESS DENIED!",1,0);
        delay(1500);
        locksDoor();
        *systemIsLocked = true;
    }
    clearData();
}
}

void checksIfSystemOpens(bool * systemIsLocked, int * activeUser , int *
ledState)
{
    if (*systemIsLocked)
    {
        printsLcd("INPUT PASSWORD",0, 0);
        checksClosedCode(systemIsLocked, activeUser , ledState, -1);
    }
}

void checksIfSystemCloses(bool * systemIsLocked, int * activeUser , int *
ledState)
{
    if (*systemIsLocked == false)
    {

```

```

        printsLcd(*users[*activeUser],0,0);
        printsLcd("press # to close",0,1);
        checksOpenedCode( systemIsLocked, activeUser , ledState, -1);
    }
}

void savesDataSdCard(char message[]) {
    // Assume rtc.now() and DateTime are correctly defined elsewhere in your
    code
    DateTime now = rtc.now();

    String dateTimeStr = String(now.year()) + "/" + String(now.month()) + "/" +
    String(now.day()) +
        " | " + String(now.hour()) + ":" + String(now.minute())
    + ":" + String(now.second());

    String messageStr = String(message) + " | " + dateTimeStr;

    if (file.open(FILE_NAME, O_WRITE | O_CREAT | O_AT_END)) {
        file.println("-----
");
        file.println(messageStr);
        file.println("-----
");
        file.close();
    } else {
        Serial.println("Error opening the file.");
    }
}

```

## Código Slave

```

#define LEDS 8
String inputString;

void setup() {
    pinMode(LEDS,OUTPUT);
    Serial.begin(9600);
    inputString.reserve(200);
}

void loop() {
    if (Serial.available()) {
        char inChar = (char)Serial.read();
        inputString += inChar;
    }
}

```

```
// If the incoming character is a newline, that signifies the end of a
command
if (inChar == '\n') {
    if (inputString.startsWith("LED:ON")) {
        digitalWrite(LEDs, HIGH);
    } else if (inputString.startsWith("LED:OFF")) {
        digitalWrite(LEDs, LOW);
    }
    // Clear the string for new input:
    inputString = "";
}
}
```

## Problemas Ocorridos

### **Falhas no Sistema de Registo:**

Inicialmente tivemos problemas na gravação e leitura de informações no ficheiro do sistema que poderiam comprometer o registo preciso de alguns eventos, incluindo a data, a hora e detalhes de acesso dos utilizadores.

### **Desgaste do Servo:**

Ao longo do tempo, o servo apresentou algum desgaste mecânico, afetando a eficácia na abertura e fecho da porta.

### **LEDs:**

Ao longo do desenvolvimento do sistema, foram queimados alguns LEDs.

### **Problemas com o Sensor de Movimento:**

Inicialmente estávamos a ter falhas no sensor, o que levou a não detetar os movimentos.

# Conclusão

Em conclusão, o nosso Sistema Automático de Controlo e Gestão de Pavilhão representa uma abordagem inovadora na automatização e segurança de pavilhões. Ao integrarmos componentes como o teclado o matriz 4x4, o display LCD, o servo motor, o RTC e os restantes dispositivos eletrónicos, conseguimos criar um sistema que não apenas simplifica o acesso, mas também regista e gerencia os eventos de forma inteligente.

A capacidade de personalização do nosso sistema, com funcionalidades como o controlo remoto, exibição de nomes de utilizadores e o registo detalhado do RTC, confere ao sistema uma versatilidade.

No entanto, é importante reconhecer algumas dificuldades, como a má implementação de cada componente do sistema.

Concluindo, o nosso Sistema Automático de Controlo e Gestão de Pavilhão não apenas oferece um controlo de acesso avançado e inteligente, como também representa modernização e eficiência na gestão de pavilhões.