

Simulação de assistente para *Smart Home* projetada em *Tinkercad*

**44549 Jorge Fernando Dias Cardoso
47289 Tiago Luís Nunes Leal Brandão
42906 Miguel Carvalho da Costa Ramalho**

Licenciatura em Engenharia Informática
Internet das Coisas
Eng. Joaquim Moreira da Silva Torres

Data: 22/04/2024



UNIVERSIDADE PORTUGALENSE

IMP.GE.203.0

ÍNDICE

1. INTRODUÇÃO	1
2. IMPLEMENTAÇÃO DO PROJETO.....	2
3. FUNCIONALIDADES.....	4
3.1. Funcionalidades exigidas	4
3.1.1. Visor LCD	4
3.1.2. Detecção e alarme de gás	4
3.1.3. Detecção e controlo de temperatura.....	5
3.1.4. Detecção e alarme de incêndio.....	6
3.1.5. Ajuste de iluminação automático	6
3.2. Funcionalidades extra.....	7
3.2.1. Abertura automática do portão	7
3.2.2. Teclado com código de segurança	8
3.2.3. Botão para abrir/fechar porta de entrada	8
3.2.4. Controlo remoto infravermelhos	9
3.2.4.1. Introdução de código de segurança	9
3.2.4.2. Visualização das temperaturas interior e exterior.....	9
3.2.4.3. Alteração da frequência (Hz) do alarme	10
CONCLUSÃO	11
ANEXOS.....	12
Anexo I - Vista esquemática do circuito.....	12
Anexo II - Código Arduino Interior (U_INT_Arduino)	14
Anexo III - Código Arduino Exterior (U_EXT_Arduino).....	21
Anexo IV - Mapa de teclas de controlo remoto.....	26
Anexo V - Comandos passados para Arduino Interior	28

TABELA DE FIGURAS

Tabela 1 – Lista de componentes.....	2
Figura 1 – Vista do circuito.....	3
Equação 1 – Cálculo da temperatura (em Celsius) capturada pelo sensor de temperatura	5
Equação 2 -Cálculo da distância entre um objeto e o sensor de ultrassom.....	7

1. INTRODUÇÃO

O presente relatório consiste na documentação do projeto a realizar no âmbito da unidade curricular Internet das Coisas, lecionada no 6º semestre do plano curricular do ciclo de estudos da Licenciatura em Engenharia Informática.

Com o crescimento exponencial da Internet, o mercado da tecnologia, tanto a nível de oferta como procura, posicionou-se de forma a explorar a capacidade dos objetos domésticos comunicarem através de redes, seja para uma aplicação de telemóvel, uma interface na web ou registos em bases de dados. Esta comunicação pode ser feita também sem ligação à Internet e permite-nos usufruir das comodidades e “inteligência das coisas”, localmente.

Ao longo do presente relatório, será documentada a simulação de um assistente de casa inteligente com tecnologia Arduino e linguagem C++, no ambiente de simulação *Tinkercad*. Para cumprimento dos requisitos mínimos exigidos pelo docente, são exigidas algumas funcionalidades (deteção e alarme de incêndio; deteção e alarme de gás; deteção e controlo de temperatura, visor LCD, ajuste de iluminação automático) e, ao critério do grupo de trabalho, foram implementadas funcionalidades extra.

De notar que, sendo um projeto elaborado em ambiente de simulação, os tempos de espera, tempos de repouso, entre outros intervalos de tempo utilizados, são estabelecidos no código (Anexos II e III) de forma a serem perceptíveis todas as funcionalidades, durante os testes realizados, em tempo útil, sendo que o tempo real não corresponde ao tempo do simulador *Tinkercad*.

2. IMPLEMENTAÇÃO DO PROJETO

Nesta secção, serão apresentadas as motivações do projeto, os componentes utilizados (Tabela 1), a vista do circuito desenvolvido (Figura 1) e a vista esquemática do circuito desenvolvido (Anexo I).

Tabela 1 – Lista de componentes

Nome	Quantidade	Componente
U_INT_Arduino	2	Arduino Uno R3
U_EXT_Arduino		
U_INT_LCD	1	Baseado em PCF8574, 32 (0x20) LCD 16 x 2 (I2C)
U_INT_TemperatureSensor	2	Sensor de temperatura [TMP36]
U_EXT_TemperatureSensor		
D_INT_GreenLed	1	Verde LED
D_INT_BlueLed	1	Azul LED
D_INT_RedLED	1	Vermelho LED
R1	6	1 kΩ Resistor
R2		
R3		
R7		
R4		
R6		
PIR_INT_MovementSensor	1	Sensor PIR
L_INT_Lights	1	Lâmpada
R_INT_LightSensor	1	Fotorresistor
GAS_INT_GasSensor	1	Sensor de gás
PIEZO_INT_Alarm	1	Piezo
K_INT_Rele	1	Relé SPDT
U_EXT_InfraredSensor	1	Sensor de infravermelho
SERVO_EXT_Gate	2	Posicional Micro servo
SERVO_INT_Frontdoor		
DIST_EXT_GateDistanceSensor	1	Sensor de distância ultrassônico (quatro pinos)
S_INT_DoorButton	1	Botão
KEYPAD_EXT_Keypad	1	Teclado 4x4

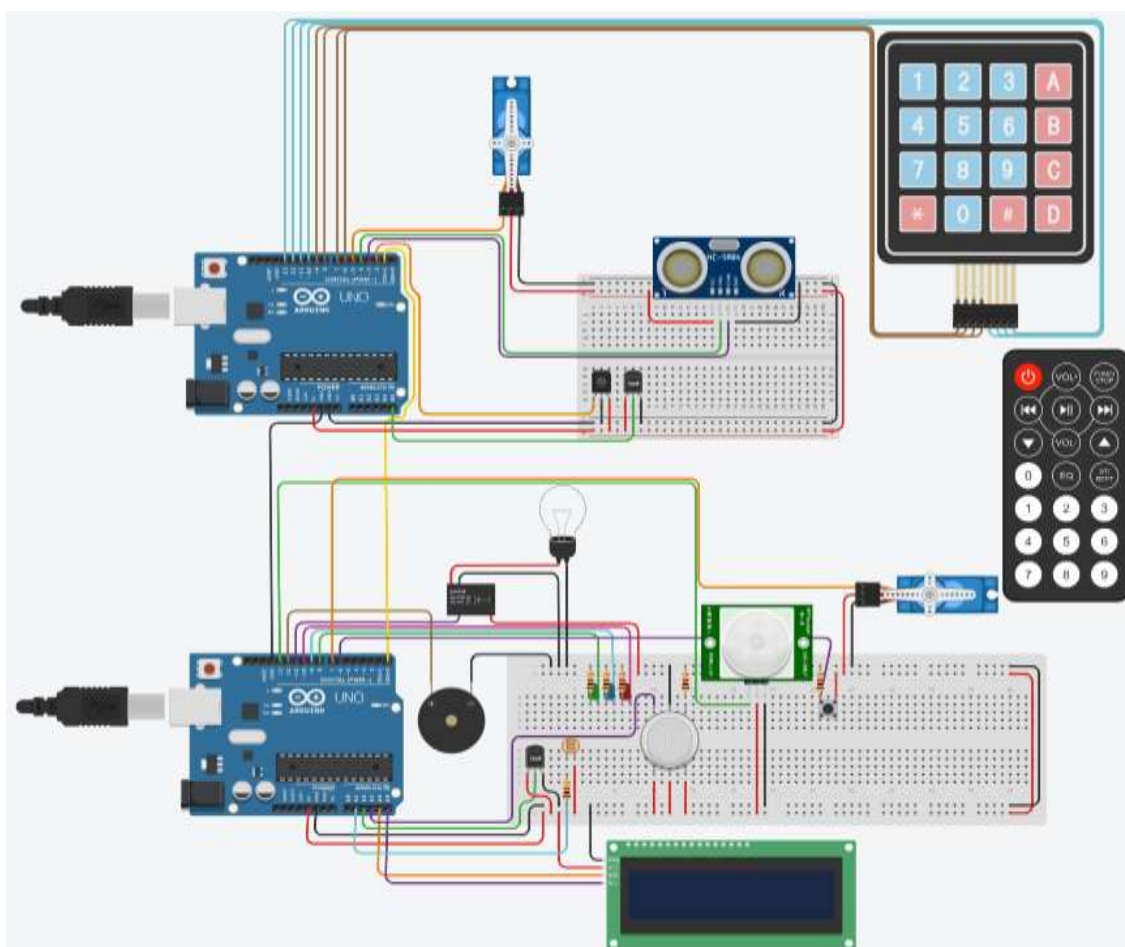
Fonte: Tinkercad (link de projeto)

Link do projeto: <https://www.tinkercad.com/things/6C1RrYInlZC-prod-24-mar-projeto-iot-20232024-?sharecode=ftWysVo9xSlrVLEyymCQkkw1OdeFRVwK8jb7SAFR6sl>

O objetivo do grupo, ao longo do projeto, foi explorar e experimentar os vários componentes que o simulador *Tinkercad* dispõe para desenvolver um assistente que poderia ser implementado numa casa, na realidade. Para além da curiosidade em relação às funcionalidades dos componentes individualmente, também foi explorada a interligação entre os vários componentes e as funcionalidades que poderiam ser extraídas caso houvesse conexões e transferências de informação. Por esse motivo, não foram utilizados componentes em duplicado, a menos que a sua funcionalidade provoque e/ou sofra comportamentos diferentes daqueles já obtidos.

O assistente de casa inteligente desenvolvido (Figura 1) simula 2 ambientes (interior e exterior) de uma casa, conectados pelos Arduinos: ambiente interior representado na parte inferior da figura; ambiente exterior representado na parte superior da figura.

Figura 1 – Vista do circuito



Fonte: Tinkercad ([link do projeto](#))

3. FUNCIONALIDADES

Nesta secção, serão descritas todas as funcionalidades (exigidas e extra) implementadas no assistente de casa inteligente, com a tecnologia Arduino e linguagem de programação C++ (Anexos I e II).

As funcionalidades exigidas são entendidas como as que representavam requisito mínimo para a conclusão do presente projeto. As funcionalidades extra são as desenvolvidas ao critério do grupo de trabalho.

3.1. Funcionalidades exigidas

3.1.1. Visor LCD

Objetivo: Transmitir, ao utilizador, mensagens de informação do sistema

Componentes utilizados: U_INT_LCD

Implementação: As mensagens apresentadas aos utilizadores variam consoante o estado do assistente, cujas alterações serão descritas abaixo. O LCD, entre outras mensagens, apresentará 3 estados:

- OK
- GAS
- FIRE

3.1.2. Detecção e alarme de gás

Objetivo: Sinalizar a deteção de gás

Componentes utilizados:

- GAS_INT_GasSensor
- PIEZO_INT_Alarm
- U_INT_LCD

Implementação: No caso do sensor de gás detetar um valor de 100 ppm de concentração de gás no ambiente, fará o alarme disparar e uma mensagem no LCD irá

aparecer com “GAS”. O valor de 100 p.p.m. (partes por milhão) foi escolhido por ser o valor de concentração de gás que começa a ser prejudicial para a saúde (apesar de ser por longa exposição).

3.1.3. Detecção e controlo de temperatura

Objetivo: Detetar alterações de temperatura e ligar/desligar ar condicionado

Componentes utilizados:

- U_INT_TemperatureSensor
- D_INT_GreenLed
- D_INT_RedLed
- D_INT_BlueLed
- U_INT_LCD

Implementação: Através do sinal capturado pelo sensor de temperatura (U_INT_TemperatureSensor), que é transformado em graus Celsius (Equação 1), é verificada a temperatura e, em caso de:

- Temperatura acima dos 27°C: é transmitida uma mensagem no LCD – “Turning on AC” – e o LED vermelho (D_INT_RedLed) acende;
- Temperatura entre os 27°C e os 21°C: acende o LED verde (D_INT_GreenLed) e mantém o ar condicionado na situação prévia (ligado ou desligado);
- Temperatura abaixo dos 21°C: é transmitida uma mensagem no LCD – “Turning off AC” – e o LED azul (D_INT_BlueLed) acende.

Equação 1 – Cálculo da temperatura (em Celsius) capturada pelo sensor de temperatura

$$input = \frac{analogRead(SENSOR_TEMP) * 5000.0}{1024.0}$$
$$celsius = \frac{input - 500.0}{10.0}$$

analogRead: função embutida no Arduino para ler a frequência recebida

SENSOR_TEMP: identificação do PIN onde o sensor de temperatura está ligado

3.1.4. Detecção e alarme de incêndio

Objetivo: Sinalizar a deteção de fogo

Componentes utilizados:

- GAS_INT_GasSensor
- U_INT_TemperatureSensor
- PIEZO_INT_Alarm
- U_INT_LCD

Implementação: Devido ao simulador *Tinkercad* não ter um detetor de fumo, foi feita uma combinação entre detetor de gás (GAS_INT_GasSensor) e sensor de temperatura (U_INT_TemperatureSensor), ou seja, caso exista uma concentração de gás de 100 p.p.m. (partes por milhão) e uma temperatura acima dos 70°C, o alarme (PIEZO_INT_Alarm) dispara e é transmitida a mensagem “FIRE” no LCD (U_INT_LCD). A temperatura de indicação de incêndio foi escolhida devido à temperatura inicial que um foco de incêndio produz, enquanto não interfere com elevadas temperaturas ambientais.

3.1.5. Ajuste de iluminação automático

Objetivo: Acender a luz através de sensor de movimento em caso de pouca luminosidade

Componentes utilizados:

- L_INT_Lights
- K_INT_Rele
- PIR_INT_MovementSensor
- R_INT_LightSensor

Implementação: Sempre que for detetado movimento pelo sensor (PIR_INT_MovementSensor), é verificado o nível de luz lido, analogicamente, pelo fotorresistor (R_INT_LightSensor). Caso o nível de luz seja considerado baixo, a luz acende. Caso o nível de luz seja considerado alto ou razoável, mesmo que seja detetado

movimento, a luz não se acenderá. Optou-se por utilizar um relé (K_INT_Rele) porque, apesar de a lâmpada (L_INT_Lights) funcionar no simulador, em ambiente real, é necessária a transformação de corrente visto que, a potência transmitida pelo Arduino é de 5V e a lâmpada utilizada funciona a partir dos 120V.

Nota: Apesar de, ser pedido, no enunciado, que fosse transmitida uma mensagem a indicar a mudança do estado da luz, optou-se por não o fazer devido à importância de outro tipo de mensagens, já que o estado das luzes é observado pelo próprio acender das luzes.

3.2. Funcionalidades extra

3.2.1. Abertura automática do portão

Objetivo: Abrir portão de entrada com aproximação do carro.

Componentes utilizados:

- DIST_EXT_GateDistanceSensor
- SERVO_EXT_Gate

Implementação: Através do sensor ultrassom de distância (DIST_EXT_GateDistanceSensor), é possível medir a distância do carro e do sensor instalado no portão através do tempo do eco (Equação 2).

Nota: Como, neste caso, está a ser desenvolvida uma simulação, não foi implementado o RFID (Identificação por Frequência de Rádio, do inglês, *Radio Frequency Identification*) que serviria para a identificação de um carro em específico e, dessa forma, o portão só iria abrir com a aproximação de um carro autorizado. Neste caso, o portão abre com qualquer objeto que se aproxime no raio de 3 metros.

Equação 2 -Cálculo da distância entre um objeto e o sensor de ultrassom

$$distance = \frac{\left(\frac{duration}{2}\right)}{20.7}$$

duration: tempo entre a emissão e a receção da frequência

3.2.2. Teclado com código de segurança

Objetivo: Abrir porta de entrada através de código introduzido no *keypad*

Componentes utilizados:

- KEYPAD_EXT_Keypad
- SERVO_INT_Frontdoor
- PIEZO_INT_Alarm

Implementação: Com a introdução do código PIN (4 dígitos), podemos introduzir o símbolo “#” ou esperar para o código ser verificado. Caso seja introduzido um código menor do que 4 dígitos, o PIN não será verificado. O símbolo “*” permite limpar o código já introduzido e recomeçar a inserção. Caso o código esteja correto, é enviado o código 400 do Arduino Exterior para o Arduino Interior que fará abrir a porta de entrada (SERVO_INT_Frontdoor). Caso seja inserido um código errado por três vezes consecutivas, será enviado o código 401 do Arduino Exterior para o Arduino Interior que fará o alarme (PIEZO_INT_Alarm) tocar até o código ser inserido corretamente.

Nota: Para a leitura linha/coluna do *keypad* ser feita corretamente, nenhuma das linhas poderá estar ligada à entrada 13, visto que, devido a uma função embutida no Arduino, a entrada 13 está ligado ao LED do equipamento e, dessa forma, transmite corrente elétrica que se traduz em erro na leitura de tensão elétrica do *keypad*.

3.2.3. Botão para abrir/fechar porta de entrada

Objetivo: Abrir/fechar a porta de entrada ao pressionar o botão

Componentes utilizados:

- S_INT_DoorButton
- SERVO_INT_Frontdoor

Implementação: De forma a poder abrir e fechar a porta de entrada (SERVO_INT_Frontdoor) do lado de dentro da casa, foi adicionado um botão (S_INT_DoorButton) que permite abrir a porta quando está fechada e fechar quando está aberta.

3.2.4. Controlo remoto infravermelhos

3.2.4.1. Introdução de código de segurança

Objetivo: Abrir porta de entrada através da inserção de código

Componentes utilizados:

- U_EXT_InfraredSensor
- Controlo remoto
- SERVO_INT_Frontdoor
- PIEZO_INT_Alarm

Implementação: Como o sensor de infravermelhos está constantemente a ler a informação enviada pelo controlo remoto, optou-se por, ao ler o sinal da tecla *Power* (Anexo IV), entrar num ciclo que regista as teclas pressionadas até ao máximo de 4 teclas ou num intervalo temporal de 5 segundos. Este conjunto de teclas pressionadas será tratado como o código PIN do *keypad* (ligar alarme ao fim de 3 tentativas sem sucesso ou abrir a porta de entrada após inserção de código válido).

3.2.4.2. Visualização das temperaturas interior e exterior

Objetivo: Transmitir temperaturas interior e exterior, no LCD

Componentes utilizados:

- U_EXT_InfraredSensor
- Controlo remoto
- U_EXT_TemperatureSensor
- U_INT_TemperatureSensor
- U_INT_LCD

Implementação: Ao carregar no botão 0 do controlo remoto, é registada a temperatura do ambiente exterior pelo sensor de temperatura (U_EXT_TemperatureSensor) ligado ao Arduino Exterior, é enviado o código 12 para o Arduino Interior juntamente com a temperatura capturada. No Arduino Interior, o

comando é separado da informação da temperatura e é também capturada, através do sensor de temperatura (U_INT_TemperatureSensor) ligado ao Arduino Interior, a temperatura do ambiente interior. Os valores são transmitidos no LCD (U_INT_LCD) (Figura 2).

Nota: Os valores capturados pelos sensores de temperatura, não são as temperaturas em graus Celsius e, portanto, têm de ser traduzidas como exemplificado na Equação 1.

3.2.4.3. Alteração da frequência (Hz) do alarme

Objetivo: Alterar o som do alarme para ser mais agudo ou mais grave

Componentes utilizados:

- U_EXT_InfraredSensor
- Controlo remoto
- PIEZO_INT_Alarm

Implementação: Através das teclas *VOL+* e *VOL-* (Anexo IV), pode-se aumentar ou diminuir a frequência (em *hertz*), para que o som do alarme seja mais agudo ou mais grave.

Nota: No código (Anexo II), é necessário verificar os valores limiares da frequência do alarme, visto que a variação é feita entre 31 e 60500 (exclusivo).

CONCLUSÃO

Com o reconhecimento, por parte do grupo de trabalho, da importância do desenvolvimento de competências relativas à área de Internet das Coisas, foi aplicada uma exploração, quanto possível, às ferramentas e interações disponibilizadas pela tecnologia Arduino no ambiente de simulação *Tinkercad*.

A consecutiva resolução de problemas não relacionados com a linguagem utilizada ou com a lógica ou compilação do código escrito, levaram à compreensão de campos de conhecimento a que o grupo não estava acostumado, como a implicação da corrente elétrica nos vários componentes e na própria execução do código.

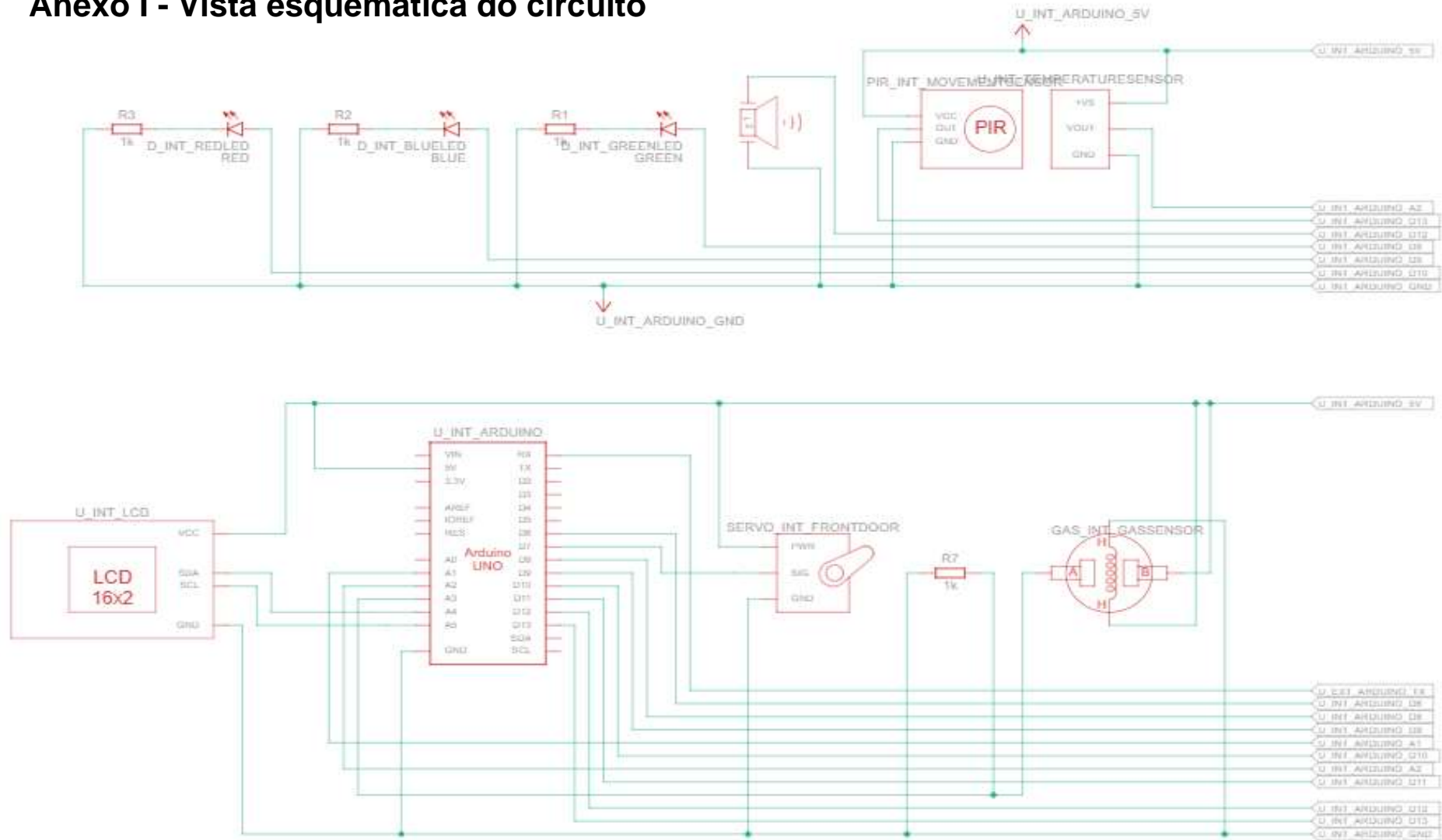
Apesar de, a área da domótica não ser um foco do grupo, foi reconhecida a importância do conhecimento apreendido ao longo do projeto.

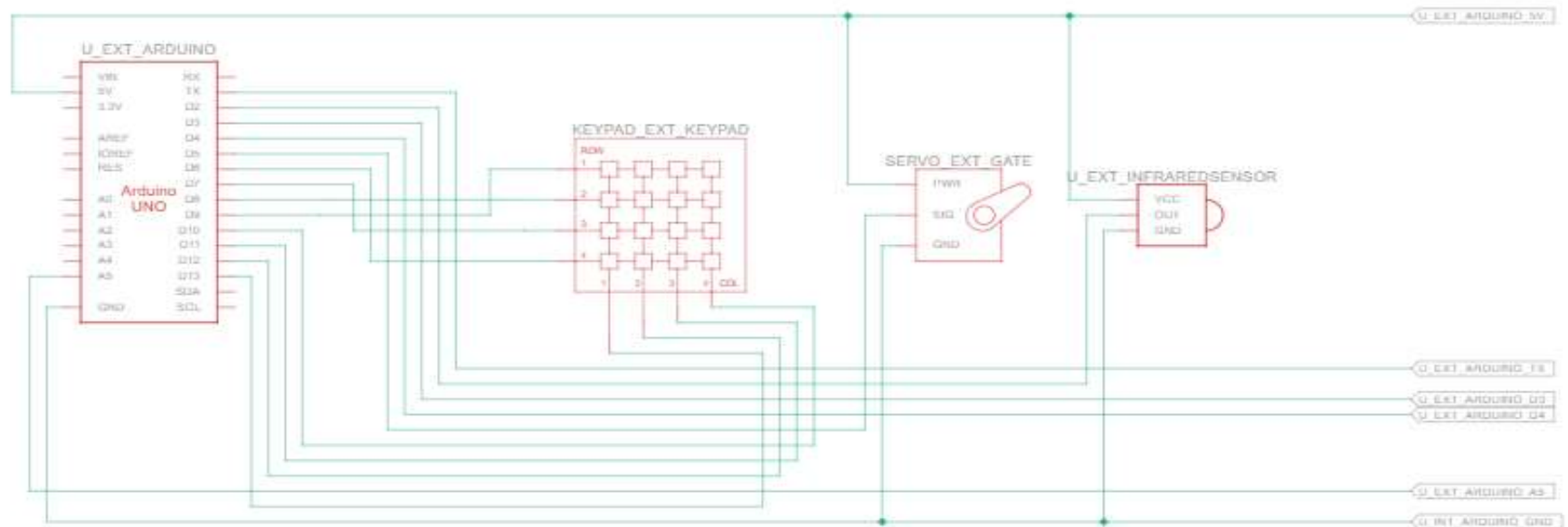
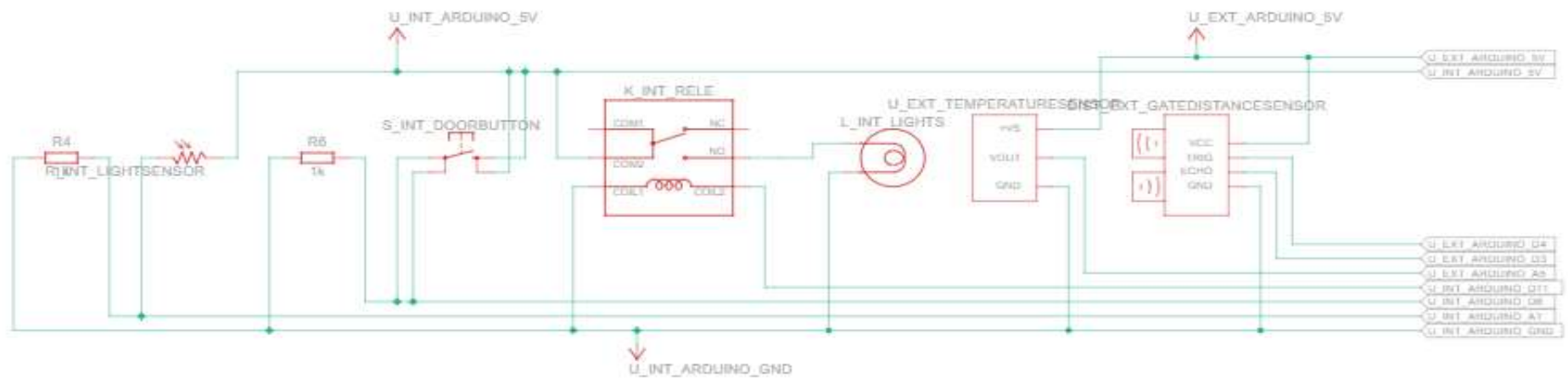
Ao longo do projeto, algumas funcionalidades foram ignoradas por falta de compatibilidade com o simulador utilizado e outras adaptadas para ir de encontro ao ambiente de desenvolvimento, visto que, impossibilita a precisão do mundo real.

Em suma, os objetivos iniciais foram concluídos com sucesso e conseguiram ser desenvolvidas mais funcionalidades do que as que estavam planeadas inicialmente, num espaço de tempo mais curto do que o previsto, permitindo ao grupo ter mais atenção aos detalhes e maior consciência do que poderia ser desenvolvido e como poderia ser feito.

ANEXOS

Anexo I - Vista esquemática do circuito





Anexo II - Código Arduino Interior (U_INT_Arduino)

```
#include <LiquidCrystal_I2C.h>
#include <Servo.h>

//SENSORES
LiquidCrystal_I2C lcd(32, 16, 2); //sensor LCD
const int SENSOR_LIGHT = A1;
const int SENSOR_TEMP = A2; //sensor temperatura
const int SENSOR_GAS = A3;

//PINS
const int PIN_BUTTON = 6;
const int PIN_SERVO = 7;
const int PIN_LED_GREEN = 8;
const int PIN_LED_BLUE = 9;
const int PIN_LED_RED = 10;
const int PIN_LIGHT_BULB = 11;
const int PIN_PIEZO = 12;
const int PIN_PIR = 13;

//standard levels
const int GAS_SAFETY = 100; //ppm
const int TEMP_SAFETY = 70; //degrees - Celsius
const int TEMP_AC_ON = 21; //degrees - Celsius
const int TEMP_AC_OFF = 27; //degrees - Celsius
const int LIGHT_ALIVE = 5000; //milliseconds
const int LUMINOSITY = 500;
const int MESSAGE_TIMEOUT = 10000;
const int MESSAGE_AC_TIMEOUT = 3000;

Servo servo;

String SYSTEM_STATUS; //system status message
bool AC_ON; //air conditioning state
bool ALARM_STATUS; //system is detecting gas
bool DOOR_STATUS;

//light states
int pirState;
int lightSensorValue;
bool lightOn;

//timestamps
float alarm_timestamp;
float ac_message_timestamp;
```



```

float message_timestamp;

unsigned long previousMillisecs = 0;
unsigned int ALARM_FREQUENCY;

int gasLevel;
float celsius;

void setup()
{
    Serial.begin(9600);

    //inputs
    pinMode(PIN_BUTTON, INPUT);
    pinMode(PIN_PIR, INPUT);
    pinMode(SENSOR_LIGHT, INPUT);
    pinMode(SENSOR_GAS, INPUT);

    //outputs
    pinMode(PIN_SERVO, OUTPUT);
    pinMode(PIN_LED_GREEN, OUTPUT);
    pinMode(PIN_LED_BLUE, OUTPUT);
    pinMode(PIN_LED_RED, OUTPUT);
    pinMode(PIN_LIGHT_BULB, OUTPUT);
    pinMode(PIN_PIEZO, OUTPUT);

    servo.attach(PIN_SERVO);
    servo.write(0);
    DOOR_STATUS = false;

    lcd.begin(16, 2);
    lcd.init();
    lcd.backlight();

    pirState = LOW;
    lightSensorValue = 0;
    lightOn = false;

    ALARM_FREQUENCY = 523;

    system_ok(); //system default
}

void loop()
{
    gasLevel = analogRead(SENSOR_GAS);
    //Serial.print("Gas level:");

```

```

//Serial.println(gasLevel);
pirState = digitalRead(PIN_PIR);
lightSensorValue = analogRead(SENSOR_LIGHT);

//check illumination
//checks light level and movement
if (pirState == HIGH && lightSensorValue < LUMINOSITY) {
    digitalWrite(PIN_LIGHT_BULB, HIGH);
    lightOn = true;
    previousMillis = millis();
}

//checks time to shut down the light after last movement detected
if (lightOn && millis() - previousMillis > LIGHT_ALIVE) {
    digitalWrite(PIN_LIGHT_BULB, LOW);
    lightOn = false;
}

float input = analogRead(SENSOR_TEMP) * 5000.0 / 1024.0;
// temperature read from tension received
celsius = (input - 500.0) / 10.0;

//check temperature
if(celsius < TEMP_AC_ON){
    //turn on blue led
    digitalWrite(PIN_LED_BLUE,HIGH);
    digitalWrite(PIN_LED_RED, LOW);
    digitalWrite(PIN_LED_GREEN, LOW);
    if (!AC_ON){
        turn_ac_on(celsius);
    }
}else if (celsius > TEMP_AC_OFF){
    //turn on red led
    digitalWrite(PIN_LED_BLUE, LOW);
    digitalWrite(PIN_LED_RED, HIGH);
    digitalWrite(PIN_LED_GREEN, LOW);
    if (AC_ON){
        turn_ac_off(celsius);
    }
}else{
    //turn on green led
    digitalWrite(PIN_LED_BLUE, LOW);
    digitalWrite(PIN_LED_RED, LOW);
    digitalWrite(PIN_LED_GREEN, HIGH);
}

//check smoke concentration (GAS and FIRE)

```

```

if(gasLevel > GAS_SAFETY){
    if(celsius > TEMP_SAFETY){
        system_fire();
    }else{
        system_gas();
    }
}else{
    digitalWrite(PIN_PIEZO, LOW);
    system_ok();
}

if(millis() - ac_message_timestamp > MESSAGE_AC_TIMEOUT){
    ac_message_timestamp = millis();
    refresh_lcd();
}

/*if(millis() - message_timestamp > MESSAGE_TIMEOUT){
    //update LCD
    message_timestamp = millis();
    refresh_lcd();
}*/

//button to close/open door
if(digitalRead(PIN_BUTTON) == HIGH){
    if(DOOR_STATUS){
        servo.write(0);
    }else{
        servo.write(90);
    }
    DOOR_STATUS = !DOOR_STATUS;
}

//receive data from exterior arduino
if (Serial.available() > 0) {
    String receivedData = Serial.readString();
    receivedData.trim();
    String receivedCommand = extract_command(receivedData);
    Serial.print("Command received: ");
    Serial.println(receivedCommand);
    //switch case para funções do comando
    if(receivedCommand.equals("401")){ //insucess security code
        ring_alarm();
    }else if(receivedCommand.equals("400")){//success security code
        servo.write(90);
        DOOR_STATUS = true;
    }else if(receivedCommand.equals("1")){ //Volume up remote
        if(ALARM_FREQUENCY < 60000){

```

```

        ALARM_FREQUENCY = ALARM_FREQUENCY + 500;
    }
} else if (receivedCommand.equals("9")) { //volume down remote
    if (ALARM_FREQUENCY > 531) {
        ALARM_FREQUENCY = ALARM_FREQUENCY - 500;
    }
} else if (receivedCommand.equals("12")) { //0 remote
    String temperature = extract_text(receivedData);
    lcd.clear();
    lcd.print("TEMP INT: ");
    lcd.print(celsius);
    lcd.setCursor(0,1);
    lcd.print("TEMP EXT: " + extract_text(receivedData));
}
}

} //end loop

//system is ok
void system_ok() {
    if (SYSTEM_STATUS != "OK") {
        SYSTEM_STATUS = "OK";
        refresh_lcd();
    }
}

//fire warning
void system_fire() {
    if (SYSTEM_STATUS != "FIRE") {
        SYSTEM_STATUS = "FIRE";
        ring_alarm();
        refresh_lcd();
    } else {
        if (millis() - alarm_timestamp > 3000) {
            ring_alarm();
        }
    }
}

//gas warning
void system_gas() {
    if (SYSTEM_STATUS != "GAS") {
        SYSTEM_STATUS = "GAS";
        ring_alarm();
        refresh_lcd();
    }
}

```

```

    }else{
        if(millis() - alarm_timestamp > 3000){
            ring_alarm();
        }
    }
}

//turn on AC
void turn_ac_on(float celsius){
    AC_ON = !AC_ON;
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("Turning ON AC");
    ac_message_timestamp = millis();
}

//turn off AC
void turn_ac_off(float celsius){
    AC_ON = !AC_ON;
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("Turning OFF AC");
    ac_message_timestamp = millis();
}

//alarm ring
void ring_alarm(){
    alarm_timestamp = millis();
    tone(PIN_PIEZO, ALARM_FREQUENCY, 5000); // 523 Hz, 5 seconds
}

void refresh_lcd(){
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print(SYSTEM_STATUS);
}

String extract_command(String text){
    int text_size = text.length() - 1;
    int index = text.indexOf("-");

    if(index >= 0){
        return text.substring(0, index);
    }

    return text;
}

```

```
String extract_text(String text){  
    int text_size = text.length() - 1;  
    int index = text.indexOf("-");  
  
    if(index >= 0){  
        return text.substring(index + 1, text_size);  
    }  
  
    return text;  
}
```

Anexo III - Código Arduino Exterior (U_EXT_Arduino)

```
#include <Keypad.h>
#include <IRremote.hpp>
#include <Servo.h>

const int PIN_IR_RECEIVER = 2;
const int PIN_TRIGGER = 4;
const int PIN_ECHO = 3;
const int PIN_SERVO = 5;
const int SENSOR_TEMP = A5;

const int SENSOR_DISTANCE = 300;
const int ECHO_TIMEOUT = 2000;
const int SERVO_TIMEOUT = 5000;
const int PASSWORD_SIZE = 4;
const int KEYPAD_WAIT = 1000;

Servo servo;
float duration;
float distance = 0;
float trigger_timestamp = 0;
float servo_timestamp = 0;
float keypad_timestamp = 0;

//4x4 keypad
const byte NUMBER_OF_ROWS = 4;
const byte NUMBER_OF_COLS = 4;
const char PASSWORD[PASSWORD_SIZE] = {'1', '2', '3', '4'};
char ATTEMPT[PASSWORD_SIZE] = {'0', '0', '0', '0'}; //input storage
int z;
char key_received;
int attempts_failed;

//keypad inputs
const char KEYMAP[NUMBER_OF_ROWS][NUMBER_OF_COLS] = {
    {'1', '2', '3', 'A'},
    {'4', '5', '6', 'B'},
    {'7', '8', '9', 'C'},
    {'*', '0', '#', 'D'}
};

byte ROW_PINS[NUMBER_OF_ROWS] = {9, 8, 7, 6}; // Linhas 1, 2, 3 e 4 do teclado
```

```
byte COLUMN_PINS[NUMBER_OF_COLS] = {13, 12, 11, 10}; // Colunas 1, 2, 3
e 4 do teclado
```

```
Keypad keypad = Keypad( makeKeymap(KEYMAP), ROW_PINS, COLUMN_PINS,
NUMBER_OF_ROWS, NUMBER_OF_COLS );
```

```
void setup()
{
    IrReceiver.begin(PIN_IR_RECEIVER, ENABLE_LED_FEEDBACK);
    pinMode(PIN_TRIGGER, OUTPUT);
    pinMode(PIN_ECHO, INPUT);
    servo.attach(PIN_SERVO);
    servo.write(0);
    Serial.begin(9600);
    z = 0;
    distance = 0;
    trigger_timestamp = 0;
    servo_timestamp = 0;
    keypad_timestamp = millis();
    attempts_failed = 0;
}

void loop()
{
    //ultrasonic echo
    if(millis() - trigger_timestamp > ECHO_TIMEOUT){
        trigger_timestamp = millis();
        digitalWrite(PIN_TRIGGER, HIGH);
        delayMicroseconds(100);
        digitalWrite(PIN_TRIGGER, LOW);
        duration = pulseIn(PIN_ECHO, HIGH);
        distance = (duration/2)/20.7;
        //Serial.print("distance: ");
        //Serial.print(distance);
        //Serial.println(" cm");
    }

    //servo work
    if(distance < SENSOR_DISTANCE && millis() - servo_timestamp >
SERVO_TIMEOUT){
        servo_timestamp = millis();
        servo.write(90);
    }
    else if (millis() - servo_timestamp > SERVO_TIMEOUT){
        servo_timestamp = millis();
        servo.write(0);
    }
}
```



```

//remote control onput
if (IrReceiver.decode()){
    auto command = IrReceiver.decodedIRData.command;
    Serial.print(command);
    if(command == 0){
        float command_pressed = millis();
        IrReceiver.resume();
        while(millis() - command_pressed < 1000 && z < 4){
            if (IrReceiver.decode()){
                auto key_command = IrReceiver.decodedIRData.command;
                if(key_command == 16){
                    ATTEMPT[z] = 1;
                }else if(key_command == 17){
                    ATTEMPT[z] = 2;
                }else if(key_command == 18){
                    ATTEMPT[z] = 3;
                }else if(key_command == 20){
                    ATTEMPT[z] = 4;
                }else{
                    ATTEMPT[z] = -1;
                }
                ATTEMPT[z] = key_command;
                z++;
                command_pressed = millis();
            }
        }
        checkPassword();
    }else if(command == 12){ // key 0 to check temperature
        float signal = analogRead(SENSOR_TEMP) * 5000.0 / 1024.0;
        // temperature from tension received
        float celsius = (signal - 500.0) / 10.0;
        Serial.print("-");
        Serial.print(celsius);
        Serial.print(" C");
    }

    IrReceiver.resume();
}

//reading keypad
float timestamp = millis();
//Serial.print("START reading");
while(millis() - timestamp < 1000){
    readKeypad();
}

```

```

} //end loop

//method to read keypad input
void readKeypad() {
    //keypad input
    keypad_timestamp = millis();
    while(millis() - keypad_timestamp < KEYPAD_WAIT) {

        key_received = keypad.getKey();

        if(key_received != NO_KEY) {
            if(key_received == '*') {
                z = 0;
            } else if(key_received == '#') { // confirm password
                checkPassword();
            } else {
                //Serial.println(key_received);
                ATTEMPT[z] = key_received;
                z++;
                keypad_timestamp = millis();
            }
        }
        if(z > PASSWORD_SIZE) {
            checkPassword();
        }
    }
}
















//method to check key
void checkPassword() {
    int correct_digits = 0;
    int i = 0;
    while(i < PASSWORD_SIZE && ATTEMPT[i] == PASSWORD[i]) {
        correct_digits++;
        //Serial.println(ATTEMPT[i]);
        i++;
    }







    if (correct_digits == PASSWORD_SIZE) {
        Serial.print("400");
        attempts_failed = 0;
    } else {
        attempts_failed++;
    }
}

```

```
if(attempts_failed >= 3){  
    Serial.print("401");  
    attempts_failed = 0;  
}  
  
// clear previous attempt  
clearPassword();  
}  
  
void clearPassword(){  
    for (int k=0; k < PASSWORD_SIZE; k++){  
        ATTEMPT[k]=0;  
    }  
    z=0;  
}
```

Anexo IV - Mapa de teclas de controlo remoto

Tecla	Imagem	Comando	Função
Power		0	É iniciada uma sessão de escuta para inserir o código de segurança para abrir a porta. Caso o alarme de segurança esteja a tocar, é desligado.
Vol +		1	
FUNC / STOP		2	
Previous		4	
Play / Pause		5	
Next		6	
DOWN		8	
Vol -		9	
UP		10	
0		12	Regista a temperatura capturada através do sensor de temperatura ligado ao Arduino Exterior.
EQ		13	
ST/REPT		14	
1		16	
2		17	
3		18	

4		20	
5		21	Durante a sessão de inserção de código PIN: -1
6		22	Durante a sessão de inserção de código PIN: -1
7		24	Durante a sessão de inserção de código PIN: -1
8		25	Durante a sessão de inserção de código PIN: -1
9		26	Durante a sessão de inserção de código PIN: -1

Anexo V - Comandos passados para Arduino Interior

Comando	Função
400	Código PIN inserido corretamente. Abrir porta.
401	Código PIN inserido incorreto ao fim de 3 tentativas. Alarme dispara.
1	Aumenta frequência (em <i>hertz</i>) do alarme
9	Diminui frequência (em <i>hertz</i>) do alarme
12	Mostra temperatura exterior e temperatura interior, naquele momento, no LCD.