

# **THESIS**

**Ruth Borges**

**2024**



**HUNGARIAN UNIVERSITY OF AGRICULTURE AND LIFE SCIENCES**

**Faculty Of Mechanical Engineering**

**Institute of Technology**

A Graduation Project submitted in partial fulfillment of the Requirements for the degree of

**Bachelor of Mechanical Engineering**

# **Data Collection and Management with Microcontrollers**

**Student**

**Ruth Zenaida Lopes Vasconcelos Borges**

**Supervisor**

**Attila Lágymányosi**

**Gödöllő, Hungary**

**November 2024**

## Contents

Introduction .....	4
1. LITERATURE REVIEW .....	7
1.1. Microcontrollers .....	7
1.1.1. Concept and history .....	7
1.1.2. Basic Structure.....	8
1.2. Application of microcontrollers in mechanical engineering .....	10
1.2.1. Control of motors .....	10
1.2.2. Automation and control systems .....	10
1.2.3. Sensors and actuators.....	10
1.2.4. Internet of things (IoT).....	10
1.3. Advantages of Microcontrollers .....	10
1.4. Challenges in the implementation of microcontrollers.....	11
1.5. Communication protocol.....	11
1.5.1. UART - Universal Asynchronous Receiver- Transmitter .....	12
1.5.2. I2C – Inter- Integrated Circuit.....	13
1.5.3. SPI - Serial Peripheral Interface.....	14
1.5.4. CAN - Controller Area Network.....	16
1.6. Case Study .....	17
1.7. Future of microcontrollers in engineering.....	19
1.8. Microcontrollers- Specifications .....	20
2. METHODOLOGY .....	21
2.1. Type of research.....	21
2.2. System Description .....	21
2.2.1. Illumination.....	21
2.2.2. Security .....	22
2.2.3. HVAC.....	22
2.2.4. Voice Assistant.....	23
2.2.5. Application for Control .....	23
2.3. Hardware .....	24
2.3.1. Microcontroller – Arduino Uno .....	24
2.3.2. Temperature sensor.....	25
2.3.3. Magnetic sensor .....	27
2.3.4. Light sensor.....	28
2.3.5. Presence sensor .....	30
2.3.6. Buzzer .....	31
2.3.7. Bluetooth Module .....	32

2.3.8.	Relays.....	33
2.3.9.	Hardware system design.....	35
2.4.	Software .....	35
2.5.	Application .....	36
3.	RESULT AND IMPLEMENTATION .....	37
3.1.	3D printing of the Model.....	37
3.2.	Software Codes.....	40
3.3.	App Screens and Codes .....	50
3.4.	Schematic of the connections.....	52
3.4.1.	Schematic Diagram of Illumination Circuit.....	53
3.4.2.	Schematic Diagram of Security Circuit .....	54
3.4.3.	Schematic Diagram of Temperature Circuit .....	55
3.5.	Errors .....	56
3.6.	Model Result .....	59
3.7.	Cost Analysis.....	62
4.	Conclusion .....	64
	<b>SUMMARY</b> .....	66
4.1.	References .....	68
4.2.	List of Figures .....	71
4.3.	List of tables.....	72
4.4.	List of Annexes.....	72

## Nomenclature

HVAC – Heating, Ventilation, and Air Conditioning.

CPU – Central Processing Unit.

ROM – Reading Only Memory.

RAM – Random Access Memory.

EEPROM – Electronic Erasable programmable read-only memory.

PIR – Passive Infra-Red.

UC – Unit of Control.

ALU – Arithmetic-Logic Unit.

CNC – Computer Numeric Control.

IOT – Internet of Things.

TX – Transmitter.

RX – Receiver.

PCB – Printed Circuit Board.

CRC – Cyclic Redundancy Check.

PWM – Pulse- Width Modulation.

DC – Direct Current.

AC- Alternating Current.

AI - Artificial Intelligence.

LDR – Light Dependent Resistor.

USB – Universal Serial Bus.

SRAM – Static Random Access Memory.

LCD – Liquid Crystal Display.

## Introduction

Residential automatization is becoming a promising trend around the world, bringing comfort, security, and energetic efficiency in modern houses. In this subject, the use of microcontrollers brings an affordable and versatile solution to implement the automatization systems. This thesis has the goal of developing an automatized system for residences that includes many aspects such as lightning, security, and climatization.

The first component of the system approaches the illumination. There will be a system of illumination control. This system will be controlled remotely, having the possibility to turn on and off the lights through an application that will be created also in the process. And, with light sensor that will activate when it is dark and deactivate the LED when the ambience is clear.

The security of the residence will be reinforced with the integration of movement sensors. When any suspicious movement is being detected, an immediate warning is sent to the property owner or anyone with authorization by the smartphone and a sound alert will be emitted. This resource adds protection to the residence, allowing the tenants to monitor and gives them an opportunity to answer in a fast way possible threats.

The goal is to develop a system that includes continuous monitoring of energetic efficiency in the different parts of the residence. It identifies places with the biggest energy consumption and gives the possibility of some actions to optimize the energy use. The data collection will be shown in a clear form and accessible, helping with the decision making to reduce the costs and waste.

The residence will have an integrated voice assistant to execute specific commands which will be automated in the system such as turning off the light, which will give the tenants fast and intuitive control in many aspects of the residence.

The system will have a heating, Ventilation and Air Conditioning (HVAC) control system. The automatization of these systems will be able to monitor and regulate the temperature, ensuring always a comfortable ambient atmosphere and always following the path of having efficient energy consumption.

This project will show how the integration of different technologies and sensors with microcontrollers can change a normal residential in a smart residential increasing the energetic efficiency, improving the security and giving a bigger level of comfort and convenience in the tenants.

I have always had a personal interest in this area, since always, but mainly during high school as I studied Electricity and Electronics in a technical school. I spent the last two years of high school developing projects as a smart watch using logic gates, a functional prototype of an Eolic tower that generates energy through the wind, as well as creating electrical projects of houses and doing electric installations in practices and many other interesting projects. These experiences increased my interest in the area and motivated me to go deeper into my knowledge and maybe explore new possibilities in this research field.

At the end of this thesis, it is expected not only to present a functional system, but also to provide a solid foundation for future innovations in home automation, highlighting the potential of microcontrollers to create smarter, safer and more efficient ambience.

### ***Aim of this Project***

- Develop and implement a system of residential automation using microcontrollers with focusing on a smart, accessible, efficient and adaptable home. With this, it's expected to explore and integrate sensors, communication protocol as Bluetooth modules to create a residential environment to promote comfort, security and energetic efficiency.
- Develop a functioning model of a home that will be printed with a 3D printer and to make the circuit with the electronic hardware and an application to control remotely the house so it will be possible to see in practice the idea of the project and the testing will be possible.
- Check the viability and scalability of the proposed solution, considering aspects such as costs, energy consumption and ease of installation and maintenance.
- By means of practical experiments and tests, this project seeks to prove the efficiency and the applicability of the system offering a relevant contribution for the residential automation and promoting the use of sustainable and intelligent technological resources available.

### ***Questions this thesis aims to answer:***

- How to implement residential automatization efficiently and accessible using microcontrollers.
- In which way the proposed system improves the energy economy in a smart home.
- Which security solutions can be integrated in a smart home to improve the tenant's protection.

- What are the challenges and limitations of the integration of a complete system with only one microcontroller.
- What is the possibility of creating and installing a functional model of an automatized home on a small scale.



## 1. LITERATURE REVIEW

### 1.1. Microcontrollers

#### 1.1.1. Concept and history

The microcontroller is a compact device that has an integrated circuit, processor core, memory and a variety of input and output peripherals. Essentially, it works as a small computer that can execute specific tasks in an efficient way and in a small space. It is applied in embedded systems that require a sequency of tasks predetermined with precision and with a small space this way it meets the demand of specific and different industries.<sup>1</sup>

So, in a direct way, we can say that the microcontroller is a device that mixes the hardware with the software. Because the program which can be C, C++, phyton, assembly, can control the hardware to do specific functions.

To better understand microcontrollers, it is important to compare them with microprocessors as sometimes people confuse them for their similarities and complementary functions. Both are semiconductors able to make operations such as searching for instructions in the memory and execute logic calculations or arithmetic's and they both send the results in an output device.<sup>2</sup>

The main difference between them is in the architecture and in the integration of the components. The microprocessor is essentially a CPU (Central Processor Unit) in an integrated circuit, and it doesn't have an internal memory or input and output interfaces and for this reason, it needs additional circuits to make a complete and functional system. But the microcontroller is an integrated unit and more complete that includes the processor, memory ROM and RAM and peripherals for input and output of data.<sup>2</sup>

The first microcontroller was made by two engineers from Texas Instruments company named Gary Boone and Michael Cochram in 1971 while Intel launched their first microprocessors. The microcontroller named TMS 1000 had 4 bits, and it had ROM and RAM memory integrated in one single chip. At the time, the microcontrollers were essentially microprocessors with built-in memory.<sup>3</sup>

Between 1972 and 1974, Texas Instruments use the TMS 1000 internally in their calculators. And in 1974, it starts to be commercialized for electronic sectors with different configurations and with the RAM and ROM memory in different sizes.<sup>3</sup>

In the 90's microcontrollers with EEPROM (Electrically Erasable Programmable Read-Only Memory) started to be created and this is a type of ROM memory but with the ability to erase electrically and programmable.<sup>3</sup>

This innovation allowed the microcontrollers to be programmed, erased and reprogrammed without the need to take it out of the circuit. Before this, it needed specialized equipment to change the device program, and the microcontroller needed to be removed from the system which was making the system slow and expensive. With EEPROM technology it was possible to reprogram the microcontroller in the circuit making the actualization of the software directly on the device with no need to give it back to the manufacturer.<sup>3</sup>

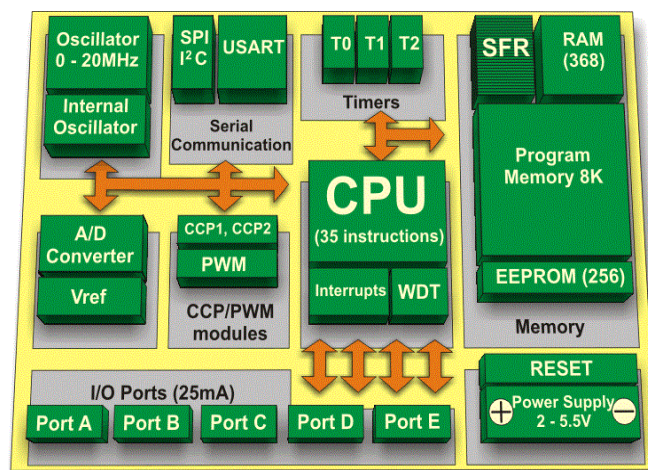
Nowadays, microcontrollers are indispensable in many electronics fields such as devices with low energy consumption, sensors, residential automatization and other technologies. The evolution of microcontrollers made the possibility of creating devices every time more potent and smaller. For example, in 2010, the Atmel announced the launching of flash microcontroller that had only 2mm per 2 mm which is a very significant advance relating to the cost and size. These compact microcontrollers are ideal for every day's products like electronic toys or electric toothbrushes.<sup>3</sup>

#### 1.1.2. Basic Structure

Internally, the microcontroller has many modules. As it is on a computer, each module has a specific function. The following figure shows the architecture of PIC16F887 microcontroller, and I will use it as an illustration of a generic microcontroller.<sup>4</sup>

**Figure 1** :Architecture of PIC16F887 Microcontroller

(Source: <https://projectiot123.com/> )



The elements are:

Power supply - gives energy to the internal component of the microcontroller making them work correctly. <sup>4</sup>

Reset – the microcontroller normally has a pin that restarts based on a voltage signal that is applied to the pin. Basically, it makes the internal program of the microcontroller go back to the beginning. <sup>4</sup>

Memory – it contains different types of memory. The ROM stores the firmware of the microcontroller meaning that it stores the instructions that cannot be modified during normal operation. RAM is used to store temporary data and changeable during the execution of the program. And EEPROM is a non-volatile memory which means that it can retain the stored data even when the electric energy is disconnected. And it can be programmed, erased, and reprogrammed electrically. <sup>4</sup>

Input and Output Ports – they allow the microcontroller to have contact with the external ambient, gathering data from sensors and sending commands to the actuators. <sup>5</sup>

Central Processing Unit (CPU) – it is the brain of the microcontroller, and it is responsible for the execution of the instruction and data processing. It is made of Unit of control (UC) which is responsible to search for instructions in the memory, decode and execute it in a sequence and arithmetic-logic unit (ALU) which makes logical operations (AND, OR, NOT) and arithmetic operations (addition, multiplication, subtraction).<sup>5</sup>

Timers – it is used to measure time, generate clock pulses, and control timed events. The internal clock gives the synchronization signal for all the operations of the microcontroller. It defines the speed at which the CPU will execute the instructions, and it coordinates the function of the I/O ports. <sup>6</sup>

Interrupts – units of interrupts allow the microcontroller to answer external events in an asynchronous way, interrupting the normal execution of the program to manage the events of more priority as entering data, timers, or communication signal. <sup>5</sup>

## 1.2. Application of microcontrollers in mechanical engineering

### 1.2.1. Control of motors

Microcontrollers are widely used to control motors, essentially in any mechanical systems. It allows the implementation of algorithms of velocity control, torque, and position, improving efficiency and movement precision.<sup>7</sup>

### 1.2.2. Automation and control systems

In industrial automatization, microcontrollers play an important role in process control from simple monitoring tasks until complex system controls of closed loops. It is used in industrial robots, CNC machines and automatic transport systems.<sup>8</sup>

### 1.2.3. Sensors and actuators

The microcontrollers are frequently integrated into sensors and actuators to monitor and control physics parameters such as temperature, pressure, humidity, and force. This integration makes a system of feedback which is essential for automatization and process control.<sup>9</sup>

### 1.2.4. Internet of things (IoT)

After the internet of things, microcontrollers became even more relevant. It allows communication between devices and systems, making data collection in real time easier and making decisions based on data of industrial ambient.<sup>10</sup>

## 1.3. Advantages of Microcontrollers

Reduction of cost – it allows the reduction of the cost for integrating multi functions in one chip, such as CPU, memory, and others. This integration makes the need for individual components in the system less, making the microcontroller cheaper.<sup>11</sup>

Increases flexibility – the program of microcontrollers offers big flexibility, allowing updates and adjustments in the software as needed without making significative changes to the hardware.<sup>11</sup>

Increases precision and efficiency – the use of microcontrollers in the control system improves the precision and efficiency in the process because of the capacity of processing data in real time and execute complex control algorithms.<sup>11</sup>

Less time to do the operation - it can execute the operations very quickly because of the optimized architecture for specific tasks.<sup>11</sup>

Simple use – because it is integrated and because it is projected for specific tasks, it is generally easy to use compared to more complex systems. <sup>11</sup>

Compact size and adaptability – microcontrollers are small which make them suitable for tasks where the space is limited and even more, they can be adapted for different needs. <sup>11</sup>

Ease of interface with other devices – microcontrollers can be easily connected to input and output devices as sensors, actuators, displays, and others making easier the interaction with the external. <sup>11</sup>

Direct programming – when the microcontroller is programmed, it executes the instructions as programmed, offering stability and predictability in the functioning of the system. <sup>11</sup>

#### 1.4. Challenges in the implementation of microcontrollers

Complexity of the design - the design of a system with microcontrollers can be complex for people without any electronics bases, making it fundamental to have knowledge in electronic, programming, and system control. <sup>11</sup>

Limitation of hardware – even though it is very powerful, the microcontrollers have some limitations related to the capacity of processing, memory and communication interface which can restrict their use in applications with more demands. <sup>11</sup>

Security and reliability - security is a critical aspect, especially in industrial fields and IoT. Microcontrollers can be vulnerable to cyber-attacks and ensuring reliability is a continuous challenge. <sup>11</sup>

Used in micro equipment – it is often used in small devices or systems as consumer electronics, home automation and others. This may limit its applicability to large or complex systems. <sup>11</sup>

Execution number is limited – some microcontrollers have a limited number of cycles of execution or lifetime specified by the manufacturer. This can be a concern in critical or long-term applications where it is necessary to ensure system reliability and durability. Additionally, in battery-powered systems, the number of runs can affect battery life and therefore must be considered during design. <sup>11</sup>

#### 1.5. Communication protocol

With the increasing of the electronics devices functions, integrating many circuits into a system requires an efficient communication approach. The use of a parallel bus with many data lines makes it inviable because of the cost and the complexity in the printed circuit board. To

overcome these challenges, the protocol of serial communication became stronger being divided into two categories, the synchronous and the asynchronous communication where the synchronous uses a watch for the synchronization between the devices and the asynchronous doesn't require an external signal. The communication protocol has an important role in the functioning of the microcontrollers as they allow the microcontrollers to exchange information with other components such as sensors and actuators. The most used are UART, I2C SPI and CAN and each of them has their own particularity and advantages.<sup>12</sup>

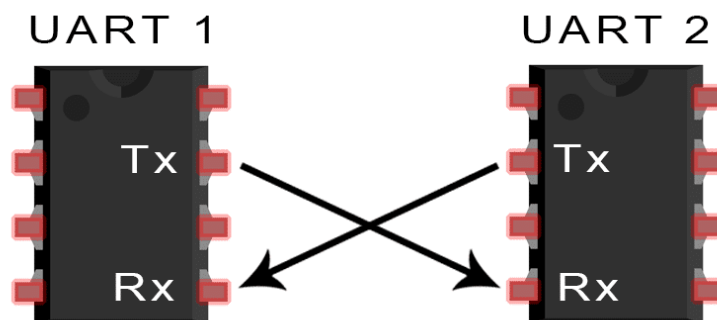
#### 1.5.1. UART - Universal Asynchronous Receiver- Transmitter

It is a hardware device common in microcontrollers and computers used to facilitate direct serial communication between the devices converting the data into a sequential stream of bits. Different from other protocols, the UART operates without a synchronization signal. It uses asynchronous communication where each device configures its own baud rate (the speed at which the data is transferred within a communication channel). It has only two wires, one to send (TX) and another to receive (Rx) the data. UART is a simple solution and economically widely used for point-to-point transmissions.<sup>13</sup>

##### 1.5.1.1. How does UART communication work?

**Figure 2:** Communication UART

(Source: <https://www.circuitbasics.com/basics-uart-communication/>)



The serial communications transfer data sequentially, bit per bit by only one channel. Different than parallel communication, the serial is less vulnerable to noises and losses of signal increasing its trustability. In UART communication, each data frame includes an initial bit, data bits from 5 to 8, a parity bit (optional), and a stop bit helping the receptor to identify and understand the data with precision.<sup>13</sup>

##### 1.5.1.2. Limitations of the UART communication

The UART transmission is efficient for short distances and simple data rates but is not ideal for connections of high speed or long distances where the signal can be destroyed.<sup>13</sup>

The devices need to be very synchronized in the baud rate and any difference can provoke mistakes.<sup>13</sup>

In ambience with multiple devices, the UART needs prioritization mechanisms and collision detection, and this limits its applications in complexes networks.<sup>13</sup>

Even though it includes a parity bit to detect mistakes, UART doesn't have resources to deal with complex mistakes needing additional approaches for ambience that requires high trustability.<sup>13</sup>

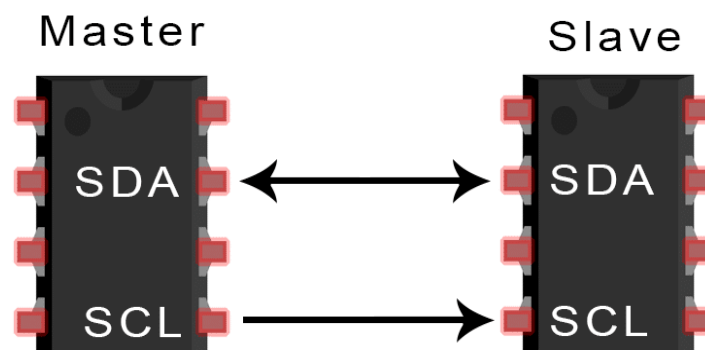
### 1.5.2. I2C – Inter- Integrated Circuit

The I2C protocol, made by Philips Semiconductors Inc in 1982, allows the devices to share in a single communication bus using only two wires, one from the clock signal (SCL) and another one for the data (SDA). This simplified configuration has many advantages for embedded systems where spaces and pins are limited. The I2C allows communication in different components in a single microcontroller where each device receives only an address allowing the master device to connect with a specific peripheral in a network.<sup>14</sup>

#### 1.5.2.1. How does it work

**Figure 3:** Communication in I2C Protocol Communication

(Source: <https://www.circuitbasics.com/basics-of-the-i2c-communication-protocol/>)



The master starts a communication generating a start signal in the SDA line while the SCL is at a high level. This indicates to the devices in the bus that a new data transmission is beginning and after the start signal, the master sends the address of the slave device in which it wants to

communicate, with a bit that will indicate if it is reading or writing (0 for writing, 1 for reading) and the slave monitors the SDA line and answers the address. <sup>14</sup>

After that, the slave that is correspondent of the address answers with an acknowledging signal pulling the SDA line low during the next clock cycle. <sup>14</sup>

Depending on the operation, the master sends or receives data through the SDA line, making the transference with the SCL line synchronous. When communication is concluded, the master generates a stop signal, releasing the SDA line while the SCL is in high level meaning that the transmission is ended, and the devices can prepare for the next communication. <sup>14</sup>

#### 1.5.2.2. Limitations of the protocol

The quantity of devices connected is limited by the bus load and this affects the trustability of the signals, especially over longer distances. <sup>14</sup>

Even though the support by many speeds, the I2C is not ideal for communication that requires high transmission rate. <sup>14</sup>

In larger systems or with a lot of services, the I2C can have interferences compromising the data integrity. <sup>14</sup>

#### 1.5.3. SPI - Serial Peripheral Interface

The SPI is a technology of serial data communication protocol that is made with peripheral devices in a fast way and real time. It is applied in communication as sensors, cards, screens and others. The first company to use this technology in their equipment was Motorola at the end of 1970 and it was connected a microcontroller with peripheral functions and later the SPI was applied by other companies. <sup>15</sup>

It is an interface of 4 wire simple communication and another characteristic of SPI is that there is no concept of transferring property of data bus meaning that there is no specific address for master and/or slave. <sup>15</sup>

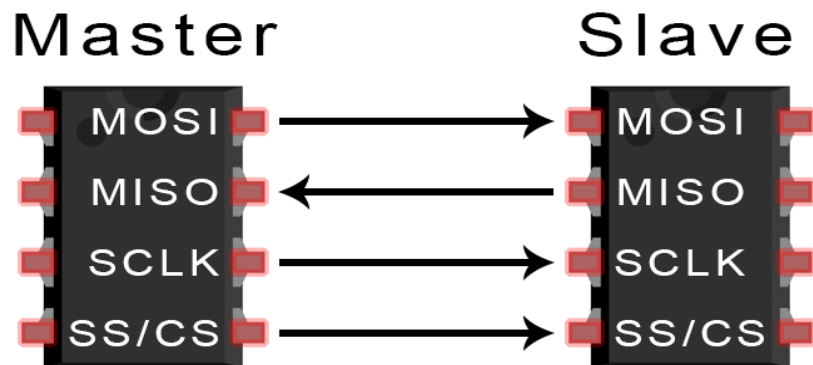
It was consolidated as an efficient protocol with the evolution of microprocessors and microcontrollers. <sup>15</sup>



#### 1.5.3.1. How does it work?

**Figure 4:** Communication of the SPI Protocol

(Source: <https://www.circuitbasics.com/basics-of-the-spi-communication-protocol/>)



The SPI allows you to choose the phase and polarity of the clock setting the protocol in four different modes from 0 to 3 and these modes define in which clock edge (rising or falling) the data will be received or sent. Master defines the frequency and polarity of the clock signal (SCLK), and it activates the slave select – chip select (SS/CS) line of the slave device in which is wished to communicate putting it in low level. This allows the that he connected slave in this line stay active and ready for communication.<sup>15</sup>

With the SS line active and configured clock, the master starts to send data through MOSI (Master Out Slave In) and the slave answers through MISO (Master In Slave Out). Sending and receiving the signal are simultaneous in full duplex mode, synchronized with the clock signal.<sup>15</sup>

When the transmission is done, the master deactivates the SS line indicating that the slave should end the communication, and the master can then activate another SS line of another slave if needed to continue the communication.<sup>15</sup>

#### 1.5.3.2. Limitations of the SPI communication protocol

Compared with other protocols, this protocol requires more wires as each devices needs a dedicated line for the SS/CS, and this can increase the circuit complexity.<sup>15</sup>

The SPI doesn't include internal mechanisms to detect errors such as CRC (cyclic Redundancy Check) which requires the need to implement external verifications to guarantee the integrity of the data.<sup>15</sup>

SPI is efficient for short distances such as in printed circuit board, but it can suffer degradation of the signal in longer distances affecting the quality of communication. <sup>15</sup>

#### 1.5.4. CAN - Controller Area Network

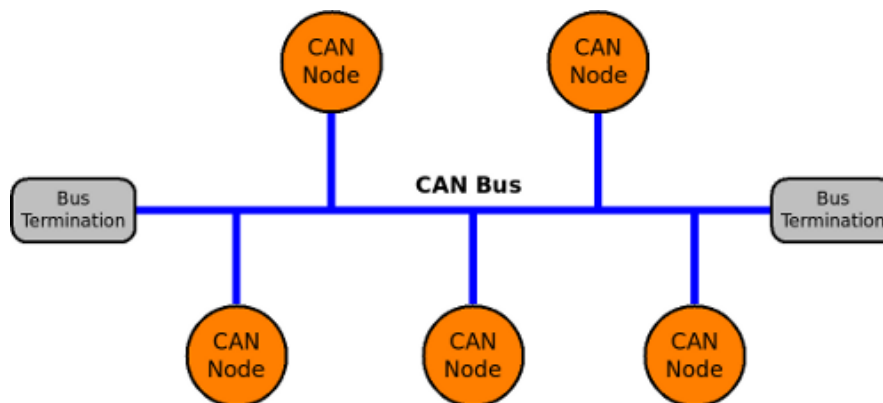
CAN Bus was created by a German company, Robert Bosch in the 90's initially for buses and trucks and now days it is widely used in the automotive, naval and agriculture industry. <sup>16</sup>

This communication is synchronous and multi master, allowing any device to act like master or slave. It uses the CSMA/CD with NDA (Carrier Sense Multiple Access/ Collision Detection with Non- Destructive Arbitration) technic to control the access the bus, allowing the messages with high priority to be transmitted without interruptions. <sup>16</sup>

##### 1.5.4.1. How does it work?

**Figure 5:** CAN Communication Protocol

(Source: [\*About the CAN Protocol and How to Debug and Transmit CAN Communication - Total Phase\*](#))



The CAN protocol works based on model or oriented messages where the data are organized in packets called frames. Each frame has an ID that defines the priority of the message. In case of collision between the messages, the priority is determined by the ID where the message with lower ID prevails. <sup>16</sup>

The structure of the CAN message includes different fields that guarantee the security and integrity of the transmitted data and more than the ID, each message has a data field that allows transport to 8 bytes per transmission in addition to a code of error verifying which is the Cyclic Redundancy Check (CRC). This field helps to guarantee that the data gets to its destination without corruption of data as it allows us to detect errors that happen during the transmission and at the end of the frame there is a confirmation field that confirms that at least a node of the

network has received the message correctly and this increases the security in the communication.<sup>16</sup>

The CAN protocol has many types of frames to deal with different communications and the two mains are data frames that are used to effectively transmit data and the remote frame that allows a node to ask for data from another without transmitting additional information. There are also Error Frames that indicate the errors detected during the transmission and the overload frames that help to space out the messages allowing more efficient processing.<sup>16</sup>

It is very robust in error detection, which makes it more secure and trustworthy. Besides the CRC, it uses a Bit Stuffing technic that inserts extra bits to keep the synchronization between the devices and avoid wrong interpretation of the transmitted data.<sup>16</sup>

When there is any error one of the node networks send an error frame showing that to the other nodes that the message shouldn't be processed ensuring that only correct data will be considered.<sup>16</sup>

#### 1.5.4.2. Limitations of the CAN Protocol.

The CAN protocol was projected to operate with 1 Mbps of transmission rate in the classic version (CAN 2.0) and this speed is enough for many automotive and industrials applications, but it is not enough for systems that need high bandwidth as advanced sensors, high-definition cameras and others. The CAN FD version (Flexible Data Rate) improved the transmission rate but still it is not enough for applications that need high speed.<sup>16</sup>

In the classic version, the maximum data length is 8 bytes of message per frame and this limit makes the bigger information to be fragmented in many messages and this can increase the transmission rate and decrease the efficiency in the applications that need to send a bigger volume of data.<sup>16</sup>

CAN is effective for network of medium size but when there is a high node number, the complexity in the network management also increases. With many devices connected, the latency (the time it takes for data to pass from one point on a network to another<sup>17</sup>) can increase as the priority arbitration increases the waiting time for messages with low priority<sup>16</sup>.

### 1.6. Case Study

#### 1.6.1. Microcontrollers and industrial robotics

Microcontrollers are important for industrial robotics having a role in control and automation in many operations acting as the brain of the system. Industrial robots are defined as mechanical devices that are programmable to do tasks, and they depend on the microcontrollers to execute commands and react to information given by the sensors.<sup>18</sup>

The connection between microcontrollers and industrial robots is especially relevant in the 4.0 industrial era where automatization, data exchange and system integration were essential. In automotive fabrication, for example the robots that were controlled by the microcontrollers execute complex tasks like welding metallic pieces and assembling components in a highly efficient and precise manner. And this didn't just increase the production speed but also improved the quality and security of the processes.<sup>18</sup>

The microcontrollers are the nucleus that allows the robots to realize advanced operations by adapting to demands in the market every day more competitive and automatization.<sup>18</sup>

#### 1.6.2. Climatization Systems

The HVAC systems are a set of integrated technologies projected to control automatically the heater, ventilation and air conditioning functions of a house or building. The main advantage of these systems is the energetic efficiency and comfort of the people in the place where it was implemented as there as precise and automatized adjustments with the Ambiental conditions.<sup>19</sup>

These systems include many components such as smart thermostats, temperature sensors and humidity, actuators, and a system that collects and manages the data to adjust decisions in real time like a microcontroller. These components work together to guarantee that the internal ambience remains ideal.<sup>19</sup>

The base of any HVAC system is central control. This system monitors the Ambiental external conditions through the sensors and it adjusts automatically the heater or air conditioner as it needed. For example, when the sensor detects that the temperature in the living room is below the desired temperature, the system will increase the refrigeration to compensate.<sup>19</sup>

#### 1.6.3. Control of electric motors

The control of electric motors is a fundamental application in automatization and robotic systems allowing them to manage the precision and speed, direction and torque of the motors in a big device variety.<sup>20</sup>

Microcontrollers such as Arduino, ESP32 and the STM32 series are widely used to control DC motors, stepper motors and AC motors thanks to its capacity to execute fast instructions e generate input and output signal to actuate in the motors. <sup>20</sup>

To control the electric motor, the microcontroller uses pulse with modulation techniques (PWM) sending intermittent signals to the control motor's terminals. Varying the width of the pulse, the microcontroller adjusts the quantity of potency that will be sent for the motor, this will control the speed of rotation. This method is very common in DC motors as it allows the microcontroller to regulate the speed with no need for a complex circuit. <sup>20</sup>

### 1.7. Future of microcontrollers in engineering

Integration of microcontrollers with artificial intelligence - The combination of microcontrollers with artificial intelligence promises to revolutionize mechanical engineering, enabling autonomous systems capable of learning and adapting to different operating conditions. <sup>21</sup>

Advanced technologies adoption - Advancing semiconductor technologies and the introduction of new microcontroller architectures will continue to expand their capabilities, enabling even more sophisticated and efficient applications. <sup>21</sup>

Sustainability and energetic efficiency - Microcontrollers have played a key role in creating more sustainable and energy-efficient systems, helping to reduce the environmental impact of the mechanical industry. <sup>21</sup>

## 1.8. Microcontrollers- Specifications

**Table 1:** Microcontrollers and their Parameters

(Source: *Made by me*)

Microcontrollers	I/O pins	Bus width	Flash memory	SRAM	Power Consump.	Speed Grade
ATmega328P <sup>22</sup>	23	8 bits	32 KB	2KB	1.1 W	20 MHz
PIC16F877A <sup>23</sup>	33	8 bits	8 KB	368 B	1.65 W	20 MHz
RP2040 <sup>24</sup>	30	32 bits	16 MB	264 kB	0.11 W	133 MHz
STM32F4 <sup>25</sup>	144	32 bits	1 MB	192 kB	0.33 W	168 MHz
ATmega2560 <sup>26</sup>	86	8 bits	256 KB	8 kB	0.9 W	16 MHz
ESP32C3 <sup>27</sup>	22	32 bits	16 MB	8 KB	? W	160 MHz
MSP430F1232 <sup>28</sup>	22	16 bits	8 KB	256 B	0.44 W	8 MHz
ATSAMD20J <sup>29</sup>	52	32 bits	256 KB	32KB	0.254 W	48 MHz

The prices of the components on the site were checked on November 1<sup>st</sup> of 2024. The prices can change with time.

The data shown in this tables refers to the specifications described in the datasheet that I showed in the references and annexes as I found different versions of datasheets for the microcontroller and the power consumption was calculated as there was no direct specification in the datasheet using the following formula:  $P [W] = V [V] * I [A]$  , were P is the power consumption in Watt, V is the operation voltage in volts and I is the current in Amperes.

## 2. METHODOLOGY

### 2.1. Type of research

The research done on this project can be considered as applied and experimental. It is applied because I wish to solve a practical problem and develop a real solution for automatized residential with the use of microcontrollers, sensors and wireless communication systems and experimental because it involves the development of a model and the process of practical tests to evaluate the system's performance. The experimental research aims to observe and test how different technologies interact in a domestic environment and how they can be applied to reach proposed goals.

### 2.2. System Description

#### 2.2.1. Illumination

The illumination system will be applied in the smart home configuration to help in the practicality of the tenant's life and to improve energy efficiency. Sensors and communication modules will be applied to make an automatized system controlled in a remote way.

Components to use:

Light Dependent Resistor (LDR) – It is a type of resistor that can change its resistance according to the light intensity around it. It will be integrated with PIR sensors so in this case when the outside is clear, the lights will not be on, improving efficiency and the costs. It will be placed at the entrance of the house and will be placed where it can detect the light intensity without any confusion. For example, it will not be placed where there are some shadows.

Relays – it's an electronic component that allows the opening and closing of an electric circuit, blocking or not the current flow. They are used as an electronic switch making the possibility to control the lighting from a distance with an app or with voice assistant. The microcontroller sends the relay the signal and the relay cuts or continues the current flow.

System functioning:

The system acts in an integrated and smart way. When the LDR detects darkness, it sends a signal to the microcontroller that will turn on the lighting of the ambience. And the other intern light will be controlled by the relay and whenever the light is turned on or off in the application, the relay will turn on or off the light.

In addition to automatic control, the tenant can control the system manually with an app that can be put in the phone making the lighting controlled remotely.

#### 2.2.2. Security

The security system is an important factor to protect the heritage and tranquility of the owner that's why with the microcontroller it is created a system integrated with sensors and alarms to detect intrusion and monitor the places in real time.

Components to use:

Passive InfraRed sensor (PIR) – they are sensors that detects the presence of a human, according to the change of infrared radiation. They detect the infrared that is irradiated by objects that emit heat. It is going to be positioned in strategic places to improve the security of houses as the entrance, windows that are accessible easily.

Magnetic Sensors – those sensors are activated when it detects a magnetic field generated by a permanent magnet. It is normally sold two parts where one is in fact and magnetic sensor and the other one is the magnet in this case in the door the sensor will be placed in the wall and the magnet in the door and when the magnetic field is not generated it shows that the door is opened and it sends a signal to the microcontroller and it activates an alarm and sends a notification for the tenant. To avoid false alarms every time the door or window is opened, this system will operate only if the tenant activates the app the security mode. This mode can be activated when no one is home.

Buzzer – it is a sound alarm in case of detecting the opening of the door when the system is in security mode.

System functioning:

In the application, the tenant must activate the security mode and when this is done, meaning that the door is not supposed to be opened. In case the system is in security mode, and the PIR sensor detects motion and the door opens, a message will be sent to the tenant and a buzzer alarm will be emitted.

#### 2.2.3. HVAC

Dealing with the heating, ventilation and air conditioning is an important component in smart home giving thermic comfort and energetic efficiency. They are responsible for regulating the temperature, humidity and air quality in the intern ambient. Traditionally, it's used in a manual



way or simple controls but with the automatization it is possible to optimize the functioning adjusting to the needs of the residents and external conditions.

Components to use:

Temperature and humidity sensors – they are sensors that detect, measure and report the humidity and temperature of the house allowing the system to adjust the heating or air conditioning in an automatic way.

Relays – this component will make the microcontroller control the actuation of the HVAC equipment such as heaters, fans and AC units.

System Functioning:

The microcontroller collects the data sent by the sensors and it decides based on algorithm pre-defined. The microcontroller reads the values of temperature and humidity continuously comparing them to the desired configuration and based on this information, it can activate or deactivate the heater or AC keeping the temperature in and comfortable way.

#### 2.2.4. Voice Assistant

The voice assistant is a revolution in how we interact with technology daily. With the capability to control the devices, access the information and execute the tasks by voice commands, this system gives convenience, and accessibility. With the microcontroller this technology is not just accessible, but also allows us to adapt to the functionalities and specific needs.

The voice assistants are for example AI programs that can interpret and answer voice commands. And constructing it with the microcontroller makes it suitable to meet the needs of the specific house and the systems that are implemented.

System Functioning:

The module captures the audio and commands, and it converts it to a signal that the microcontroller understands and then it processes the command identifying the correspondent action. The microcontroller executes the command according to the integrated circuit and then the system plays the audio confirming the execution of the command.

#### 2.2.5. Application for Control

The app for controlling smart houses is a tool that allows interaction between the residents and the automatized dispositives. It uses communication modules to make the connection between

the app and the software. They are easily accessible by smartphones, tablets or computers giving a friendly interface to manage the systems as illumination, climatization and security.

### 2.3. Hardware

The hardware is an important part of the project because it is the part that the components that will be chosen are chosen and among the various options, I choose what suits best my project and always seek for the efficiency and about the costs too.

#### 2.3.1. Microcontroller – Arduino Uno

For my project, the chosen microcontroller is Arduino UNO. It is one of the most popular and accessible in the world used to create different types of electronics projects. It is a hardware platform for open codes that allows us to make interactive projects from easy light blink to complex robots.

**Figure 6:** Arduino UNO

(Source: [\*Elektronik und Technik bei reichelt elektronik günstig bestellen\*](#))



##### 2.3.1.1. Characteristics

- The heart of the Arduino UNO is the microcontroller ATmega328P.
- It has many analogs and digital pins that can be configured as input or output to control de devices.
- It can be powered by an external power supply or by computer by a USB port.
- Communication with the computer is made by the USB port allowing the programming and monitoring of the Arduino.
- The Arduino is known for its simplicity and for its intuitive programming language and big online community that gives support and project examples.

#### 2.3.1.2. Why did I choose the Arduino Uno for my project?

I chose the Arduino Uno – Atmega328P to be the microcontroller for the smart home project for many reasons as the versatility, facility to use, and big support community online. One of the main advantages of Arduino Uno is the simplicity which is ideal for who is starting in the development of integrated systems and its big amount of documentation and tutorials makes it easy to program and to integrate the sensor and other devices that are important for residential automation.

More than that, the Arduino Uno has a low cost which makes it viable for project with low budget without lowering the quality. Its compatibility with a wide range of sensors and modules allows it to manage many aspects of the house like illumination, climatization and security.

Another important fact is that the microcontroller can be programmed in simple language based on C/C++, allowing the personalization of the system according to the specifications needed. And the availability of the pre-programmed libraries and the developer's community makes it easier to solve some problems and add new features in an easy and efficient way.

Its capacity to communicate with other dispositions such as with cables or wireless allows the integration of the system which allows the remote control by apps or virtual assistant increasing the practicality in the house management.

The disadvantages I found for the use of the Arduino are that its processing and storage capacity is limited. The microcontroller has only 32KB of flash memory, 2 KB of SRAM and 16 MHz of frequency. But for this specific project that will realize simple tasks it is enough but, for projects that are more complex, and it need simultaneous processing of multiples devices or integration of advanced algorithms it won't be a good option.

Another disadvantage is that it doesn't have Wi-Fi or Bluetooth connectivity on the circuit board, different from other microcontrollers that have it directly connected to the microcontroller. But for that problem, I will use a Bluetooth module and if needed it is possible to use a Wi-Fi module but, in this project, I will use only the Bluetooth module. This increases the cost and adds a little bit of complexity to the project.

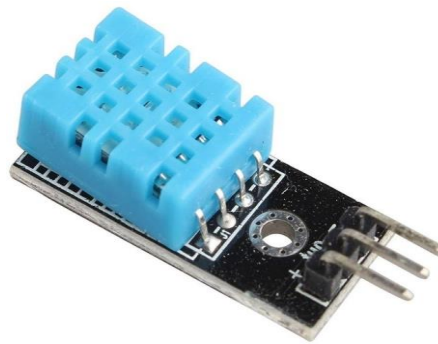
#### 2.3.2. Temperature sensor

The temperature sensor that will be used will be the DHT11. This sensor allows us to measure the temperature and humidity with the Arduino. This module uses digital pins to send information. It can be found individually, or which is the case used in this project, insert in a

PCB (printed circuit board) which is a board made with electric conductors, metal interconnection, insulators and other components connected according to a design using copper tracks etched into the surface of the board.

**Figure 7:** DHT11 Sensor

(Source: [\*#1 Electronic Online Store Canada For Hobby And Education\*](#).)



#### 2.3.2.1. Characteristics

- The temperature measurement range is from 0 °C to 50 °C with  $\pm 2$  °C precision and the humidity measurement range is from 20% to 90% relative humidity and with  $\pm 5\%$  relative humidity precision.
- It uses a digital interface which means it sends the data directly to the microcontroller from only one communication pin.
- It has an easy protocol, allowing us to make easy connections with the microcontroller.
- The energy consumption is low, it is around 0.3 mA in measurement mode and 60  $\mu$ A in standby mode.

#### 2.3.2.2. Why did I choose the DHT11 sensor?

This sensor was chosen because of a combination of facts such as costs, simplicity and the features appropriate for the ambient monitoring. As the main goal of this project is to give a project with an efficient and accessible system of a smart home, DHT11 fills the basic needs of temperature and humidity control.

The DHT11 is very easy to integrate with the Arduino platform and other microcontrollers. Its interface allows the reading of the data with only one communication pin, simplifying the configuration of the hardware and program. This decreases the development time and makes it easier to implement the system even for those with less experience in electronics.

Even though the measurement precision range is limited, these specifications are enough for a smart home project where the goal isn't extra precise measurements but the automation and basic control of the systems. As it is in my project, it will monitor the temperature of the ambience and present it so the climatization devices can be adjusted and the tenants can be notified if the weather conditions are uncomfortable or dangerous.

Another fact is the low energy consumption and in a smart home system, it will be operating for 24 hours per day and the energy consumption is a crucial component and this sensor has a low energy consumption, around 0.3 mA in measuring mode and 0.06 mA in standby mode.

And easy ways to buy it in the stores and its big documentation and community support makes the sensor a practical choice in the disponible terms as in technical support ways if some adjust or improvement is needed in the project.

### 2.3.3. Magnetic sensor

Magnetic sensors are widely used in electronics projects to monitor the opening and closing of doors, windows and other mobile elements using magnetic field detection. It's ideal for security systems on doors or windows checking if they are open with no magnetic field detected or closed when there's a magnetic field. This sensor combines two principal elements, the permanent magnet and the reed switch (a magnetic switch). This sensor was chosen for the security system on the door because of its efficiency in detecting the magnetic field, simplicity of integration with microcontrollers and its versability in automation projects. This sensor offers a trustable and accessible solution to monitor whether the doors were opened or not.

**Figure 8: Magnetic Sensor for Doors**

(Source: <https://www.autocorerobotica.com.br/sensor-magnetico-com-fio-para-portas-e-janelas> )



#### 2.3.3.1. Characteristics

- It can be installed on the surface of the door or window in a visible way or embedded, inside of the door or window structure being more discreet ideal for places where the aesthetic is more important.
- It can relate to wires in the control system or wireless and they work with batteries, and they communicate through wireless technology such as Wi-Fi or radio frequency.
- The output can be configured as NO (normally opened) or NC (normally closed) depending on the signal type that the system needs.
- The sensors are projected to activate or deactivate from a specific distance between the magnet and the reed switch, and this specification varies from the sensor model.

#### 2.3.3.2. Why did I choose the magnetic sensor?

The Magnetic sensor is ideal to monitor the doors because it works as a digital detector activating or deactivating the output when the magnetic field is present or not.

When it's installed on the door, the sensor automatically detects if the door is open because the magnet will go away or close when the magnet approximates giving the system precise and immediate information for the system.

This sensor is compatible with the Arduino uno platform which facilitates integration with the control system. The sensor has a digital output that makes it easier the configuration with the security system that will send notifications or activate an alarm when the door is opened without authorization.

And comparing it with other security sensors such as cameras or other security sensors, the magnetic sensor is a low-cost option but still gives a high trustability. Even with the accessible price, it offers the security needed to protect the smart home from intrusion.

As energy consumption efficiency is an important part of the project, this sensor is a good option because it has low energy consumption, improving the energy efficiency of our project.

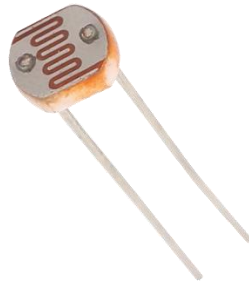
#### 2.3.4. Light sensor

This is an electronic component that detects light intensity in the ambient and it converts this information into electric signal that can and will be interpreted by electronic devices like the microcontroller. The material used in the sensor LDR (light dependent resistor) that is the one used in the project, is sensitive to the light and when the light falls over it, the photons excite the electrons in the material decreasing the electric resistance. With less light the LDR

resistance is big which means that little electric current goes through it and with a lot of light, the LDR resistance decreases, allowing more electric current to go through the circuit.

**Figure 9: Light Sensor**

(Source: <https://market.samm.com/ldr-light-sensor-5mm>)



#### 2.3.4.1. Characteristics

- The LDR has relatively slow answering time when compared to other light sensors. It takes some milliseconds to adjust its resistance when there is an abrupt light intensity.
- The LDR is more sensitive to visible light and its sensibility falls in the ultraviolet and infra-red light.
- They are generally fabricated with cadmium sulfide (CdS) which is a semiconductor material that changes its electrical properties according to light responses.
- There are some LDR made of other materials as lead sulfate (PbS), but the CdS is more common because its response in the visible light range.
- The LDR is a passive component which means it doesn't generate electricity, only changes its resistance.

#### 2.3.4.2. Why choose the LDR for my project?

The LDR can detect automatically the light intensity of the ambient light which means that the sensor can turn on the light of the entrance of the house, which is located outside of the house, when it's night it turns the light on and turns it off in the morning when the street is clear without the need of manual intervention. This level of automation does not only give convenience, but it also eliminates the waste of energy as the light will be turned on only when it's needed.

One of the main reasons that I chose this sensor for the outside is the energy economy as in an external environment the lights usually are on for long periods and sometimes when it is not needed. With the LDR the illumination is controlled by natural lighting which reduces energy consumption.

Another factor is the low cost for my choice as it is important to choose sensors that are efficient and accessible. And they are components easy to install which makes the circuit easier to manage and easier to program.

#### 2.3.5. Presence sensor

The presence sensor PIR (Passive Infrared Sensor) is an electronic component widely used to detect motion based on the infrared radiation emitted by the bodies. It uses a pyroelectric sensor that detects the changes in the level of infrared radiation and when it detects the differences like the presence of a body in movement it activates an output signal.

**Figure 10:** Presence Sensor

(Source: [\*HC-SR505 Mini Infrared PIR Motion Sensor Infrared Detector Module buy online at Low Price in India - ElectronicsComp.com\*](#))



##### 2.3.5.1. Characteristics

- It is sensitive to the heat of living being as human beings and animals.
- Almost of the PIR sensors use the Fresnel Lense that expands the detection field e focusses the infrared radiation on the sensor. This lens basically allows the sensor to detect movement in a larger area.
- The vision field of a PIR sensor is generally between 90° a 180° depending on the design of the lens and the sensor and the detection range can be between 5 to 15 meters but there are sensors that have a bigger detection range.



#### 2.3.5.2. Why did I choose the PIR sensor for my project?

The PIR sensor is a great choice for the security system because it has an excellent capacity for movement detection based on infrared radiation emitted for bodies such as human beings or animals.

The main advantage of using this sensor is the energy economy that the PIR sensor gives as they activate the devices only when they detect movement, so this helps with the waste of energy and even more than that is that the sensor by itself has low energy consumption making improving the energy consumption value.

Security is an important factor in a house and adding the PIR sensor increases the security of the property, as when it detects the movement while the system is the security mode it will activate the alarm and send a notification so the tenant can make fast decisions to protect the house.

Another reason is the ease of installing the sensor in the circuit and the low cost also as it is an accessible device keeping the project at a low price.

#### 2.3.6. Buzzer

The buzzer is an electronic device that emits sounds and gives an audible alert when it receives an electric signal. When an electric current is applied in the buzzer it vibrates to produce sound. Some buzzers come with an internal oscillator which means that it can generate sound by itself by only receiving power when others need to be controlled by a microcontroller or an external circuit.

**Figure 11: Buzzer**

(Source: [Active Buzzer / Geek Electronics](#))



#### 2.3.6.1. Characteristics

- The buzzers are small which makes it easier to be integrated into any circuit.
- The typical frequency is around 1 kHz and 5 kHz, and the volume and frequency can be different according to the buzzer's model.

#### 2.3.6.2. Why did I add the buzzer to my project?

Using the buzzer is a strategy to improve the functionality of the security system. When the house is in the security mode and there is any violation of what is supposed to be the presupposed situation, the buzzer will be activated alerting the people around or scaring the intruder away.

And the buzzer is extremely efficient in energy consumption matter and the installation and integration in the circuit is easy making it a good solution for the security matter.

#### 2.3.7. Bluetooth Module

The Bluetooth module is a device that allows wireless communication between electronics devices using Bluetooth technology. This communication is made through short range radios waves generally in the 10 to 100 meters range depending on the module potential.

**Figure 12:** Bluetooth Module

( Source: [\*HC05 Bluetooth Module - 3.3-6V input voltage.\*](#) )



#### 2.3.7.1. Characteristics

- The Bluetooth module is compatible with mobile devices such as smartphones, tablets and laptops which makes it easier to create interactive systems.
- Depending on the Bluetooth version and transfer speed of data it can range from 1Mbps to 50 Mbps.

#### 2.3.7.2. Why use the Bluetooth module in my project?

The Bluetooth module HC-06, which is the one I will use in the project, is extremely easy to integrate with microcontrollers. It uses serial communication (UART) which means that it is easy to connect the module to the microcontroller with only some pins and an easy software configuration.

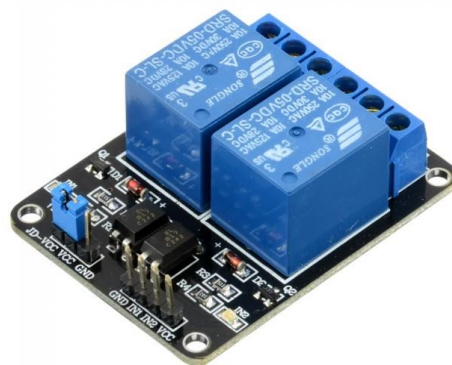
The HC-06 will facilitate the remote control of some parts of the smart home, such as for example lighting or security mode activation.

#### 2.3.8. Relays

Relays are electronics components that allow control devices such as lamps, motors and others using low potency signals given by microcontrollers or digital systems. It works like a electric control switch allowing electronic circuit and microcontrollers to control the devices connected directly to electric grid. The electromechanical switch is made with a coil, a moving contact and two fixed contacts where one is normally opened (NO) and the other is normally closed (NC). When the current is applied to the coil, it generates and magnetic field that moves the moving contact, changing the switch state.

**Figure 13: Relay Module**

(Source: [5V 2 Channel Relay Module 10A / Buy in Australia / CE05114 / Core Electronics](#))



##### 2.3.8.1. Characteristics

- When the relay is off, the NO contact is opened that means that there is no connection between the controlled device and the power and when the relay is activated, the NO contact closes, allowing the current flow and connecting the controlled device.
- When the relay is deactivated, the NC contact is closed allowing the current flow, and when the relay is activated, the NC contact is opened stopping the current flow and turning off the controlled device.
- It can control devices with alternating current (AC) and direct current (DC).
- The relay modules are projected to be easy to use with microcontrollers as they generally have input pins clearly identified as VCC, GND and IN that makes it easy for the connection and integration of the system.
- The relay module is available in different versions with 1 channel, 2 channel, 4 channel or more, allowing to control multiples devices with the same module.

#### 2.3.8.2. Why chose the relay module for my project?

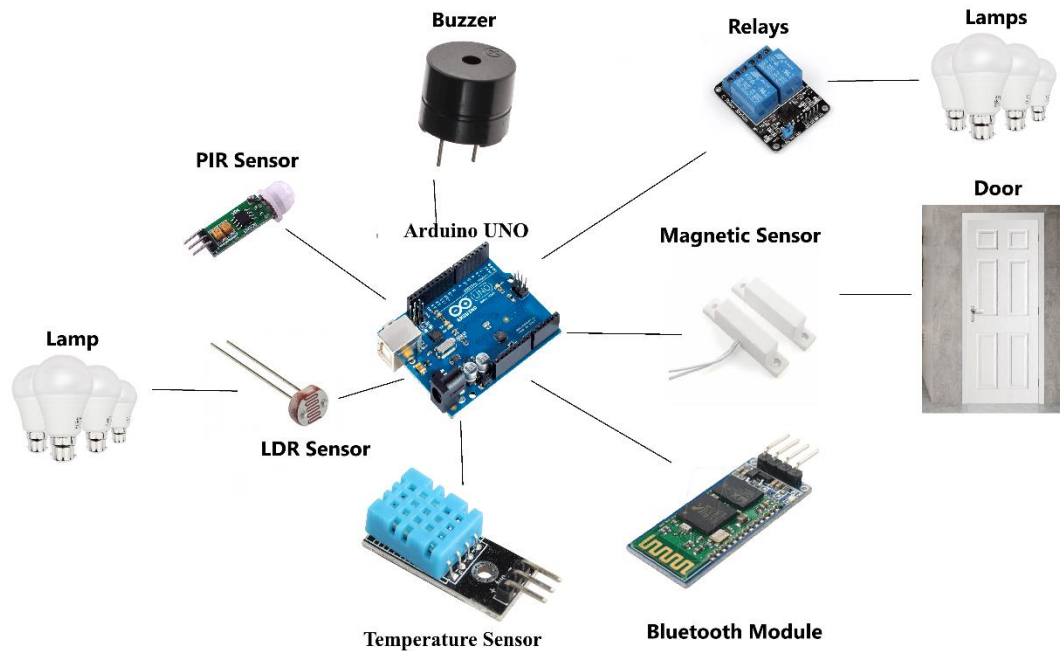
The relay module is very easy to integrate into the project as it has a simple interface and with 3 pins for control (IN, VCC, GND) which makes it easier for the connection between the microcontroller and the program.

The relay gives flexibility in implementing functions in the smart home. And programming it to turn on and off the lights remotely gives comfort and efficiency for the project. And this module price is low and as it has options of channels, it gives an economic solution for the automatization of lamp control.

### 2.3.9. Hardware system design

**Figure 14:** System Design

(Source: *made by me*)



### 2.4. Software

Choosing the software is an important step because that's what will control and integrate all the devices used and will make the project work as we define. The software is the logical part of a system, and it is made with programs and applications that will execute functions in electronic devices. It is responsible to monitor and control all the devices and components of the system such as illumination, security, climatization, and others.

Choosing the software is important to consider aspects such as compatibility, with the devices, easy to use, capability of integration with other systems and data security.

The chosen software is Arduino IDE (Integrated Development Environment) and it is a strategic decision as it offers an easy platform, accessible and strong to develop the control systems. It is a programming tool that facilitates the code creation for the Arduino UNO as it is widely used for projects of residential automatization.

The interface is very friendly and intuitive, making it ideal for those that are starting to develop automatization projects because even with not much experience in programming, it is possible to create codes to control the devices like led, sensors, relays and others that are wanted. The

programming language is C/C++ and there is a wide documentation available online and that allows even beginners and advanced developers to set the smart home in fast and efficient way.

The Arduino IDE is free and open-source software which means that it is cost effective.

## 2.5. Application

For remote control and a better understanding for the tenants, an application was created to make communication between human- machines possible. The app was created using the **MIT App Inventor** and according to the needs they created pages to control the system and codes to talk between the microcontroller and the app through the Bluetooth module.

**Figure 15:** App Starting Screen

(Source: *Made by me in the MIT App Inventor*)



### 3. RESULT AND IMPLEMENTATION

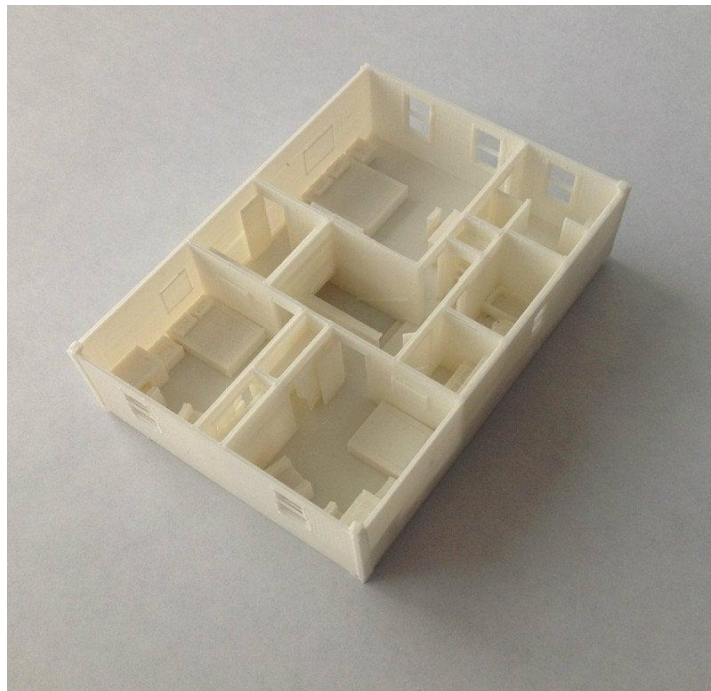
#### 3.1. 3D printing of the Model

To create the model for better visualization of the project I decided to use an additive technology to create a house in miniature and then implement the connections with the components, program and application.

From the web site [\*Thingiverse - Digital Designs for Physical Objects\*](#) I got a free model of a house to 3D printed it in the university 3D printer.

**Figure 16:** 3D Drawing of the House

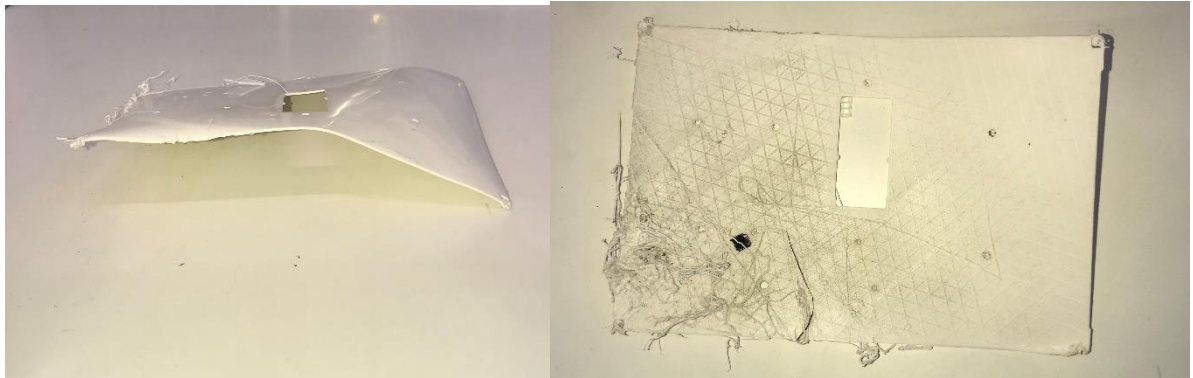
(Source: [\*Two-Story Spec House by pwc-phil - Thingiverse\*](#))



The printed model was the following and to create it I used the Ultimaker 3D printer, it would take around 24 hours to complete. But it is very common to make errors to happen in extremely long printing because of the complexity of the process. And the first try in the model printing was unsuccessful.

**Figure 17: Workpiece Printed Wrong**

(Source: *Made by me*)



The errors that occurred and made the 3D print not successful are the following:

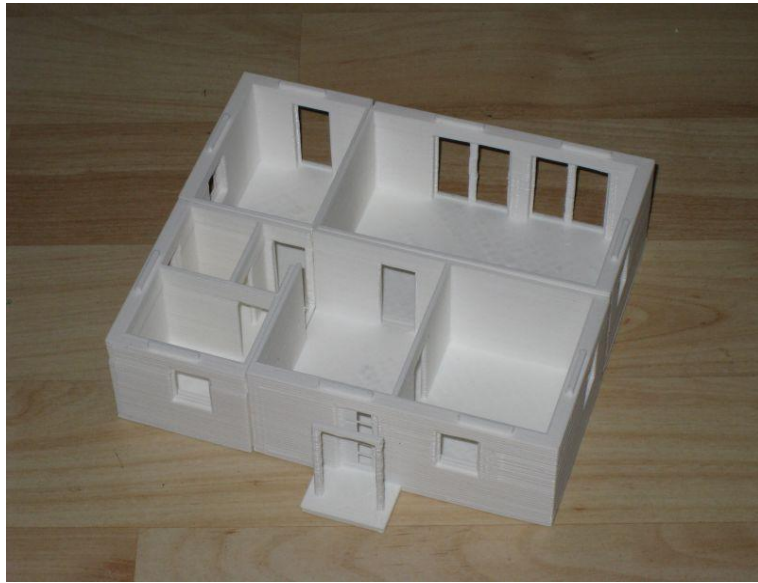
- The first layer was not well adhered to the table and if the first layer is not perfectly attached to the table, there is a detachment risk in the process of printing it. That's why in the picture it is notable that it is not a plan surface which took to the next problem. When I started the process I had to leave it to do the process as it is a long process of almost one day the monitoring through the whole process is a hard so it was working by itself and after 24 hours when it was checked only 3 layers was done “correctly” because the first layer was not well attached to the table, the nozzle was not align to the layers so it couldn't create other layers and the filaments were not creating layers.

To get better results, I decided to use another 3D printing machine which is the CraftBot + to choose another model where it is smaller and there is the possibility to print each room alone and then connect the rooms. And this is a better solution because with smaller dimensions, the printing sessions are smaller, and the possibility of monitoring is better and the chances of things going wrong are smaller.



**Figure 18:** Model for 3D Printing

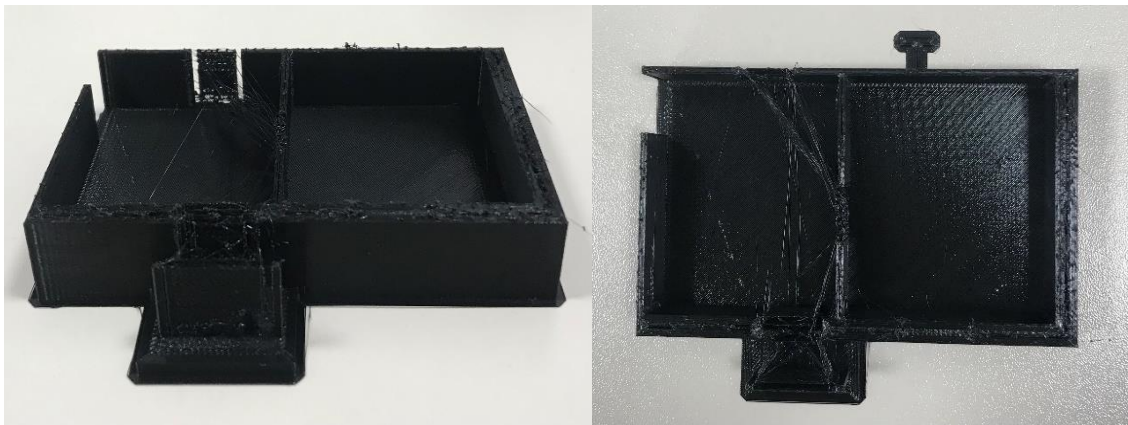
(Source: [\*House by Domonoky - Thingiverse\*](#))



After printing the house in parts there was another problem printing the room, that is the entrance as the filament ended and the machine just continued the movement but without the material making the layers. So, I had to print it again making sure the filament was enough.

**Figure 19:** Room Printed Wrong

(Source: *made by me*)



After almost 23 hours printing the model, I got the result. The house was divided into 4 parts so the printing process would be easier and after that I used the glue to glue the rooms so I could start the process to make the installation.

### 3.2. Software Codes

#### 3.2.1. Illumination control

##### ➤ Library

```
2  #include "SoftwareSerial.h"
```

- `#include "SoftwareSerial.h"` – this line includes the SoftwareSerial library in the program and this library allows the Arduino to create new additional serial ports using digital pins to send and receive data.

##### ➤ Pin setup

```
10  #define PRESENCE_PIN 3

13  #define RELAY_LAMP1_PIN 7
14  #define RELAY_LAMP2_PIN 8
15  #define BLUETOOTH_RX_PIN 10
16  #define BLUETOOTH_TX_PIN 11
17  #define LED_ENTRANCE A0
18  #define LED_MAINROOM A1
19  #define LED_KITCHEN A2
20  #define LED_LIVING_ROOM A3
21  #define LED_BATHROOM A4
22  #define LIGHT_SENSOR_PIN A5

26  SoftwareSerial BTSerial(BLUETOOTH_RX_PIN, BLUETOOTH_TX_PIN);
```

Those lines are the directive of the preprocessor of C/C++, and they are processed before the compilation of the code. In this case I defined in these lines the pins or the components I will use.

- `#define PRESENCE_PIN 3` – defining that the PIR sensor is connected to pin 3.
- `#define RELAY_LAMP1_PIN 7` – defining that the relay 1 is connected to pin 7.
- `#define RELAY_LAMP2_PIN 8` -- defining that the relay 2 is connected to pin 8.
- `#define BLUETOOTH_RX_PIN 10` – it defines that the data receiver of the Bluetooth module is connected in the digital pin 10.
- `#define BLUETOOTH_TX_PIN 11`– it defines that the data transmitter of the Bluetooth module is connected in the digital pin 11.
- `#define LED_ENTRANCE A0` -- defining that the LED is connected to the analog pin 0.
- `#define LED_ROOM A1` -- defining that the LED is connected to the analog pin 1.
- `#define LED_KITCHEN A2` -- defining that the LED is connected to the analog pin 2.

- `#define LED_LIVING_ROOM A3` -- defining that the LED is connected to the analog pin 3.
- `#define LED_BATHROOM A4` -- defining that the LED is connected to the analog pin 4.
- `#define LIGHT_SENSOR_PIN A5` -- defining that the LDR is connected to the analog pin 5.
- `SoftwareSerial BTSerial(BLUETOOTH_RX_PIN, BLUETOOTH_TX_PIN);` -- this will define the pins that the Bluetooth module will communicate with the Arduino.

➤ Pin setup

```

42 | pinMode(PRESENCE_PIN, INPUT);
44 | pinMode(LIGHT_SENSOR_PIN, INPUT);
46 | pinMode(BLUETOOTH_RX_PIN, INPUT);
47 | pinMode(BLUETOOTH_TX_PIN, OUTPUT);
48 | pinMode(RELAY_LAMP1_PIN, OUTPUT);
49 | pinMode(RELAY_LAMP2_PIN, OUTPUT);
50 | pinMode(LED_ENTRANCE, OUTPUT);
51 | pinMode(LED_MAINROOM, OUTPUT);
52 | pinMode(LED_KITCHEN, OUTPUT);
53 | pinMode(LED_LIVING_ROOM, OUTPUT);
54 | pinMode(LED_BATHROOM, OUTPUT);

56 | digitalWrite(RELAY_LAMP1_PIN, HIGH);
57 | digitalWrite(RELAY_LAMP2_PIN, HIGH);
58 | digitalWrite(LED_BATHROOM, LOW);

```

- `pinMode(PRESENCE_PIN, INPUT);` -- it configures the PIR sensor pin's as input.
- `pinMode(LIGHT_SENSOR_PIN, INPUT);` -- it configures the LDR sensor pin's as input.
- `pinMode(BLUETOOTH_RX_PIN, INPUT);` -- it configures the Bluetooth receptor as an input.
- `pinMode(BLUETOOTH_TX_PIN, OUTPUT);` it configures the Bluetooth transmitter as an output.
- `pinMode(RELAY_LAMP1_PIN, OUTPUT);` -- it configures the relay 1's pin as output.
- `pinMode(RELAY_LAMP2_PIN, OUTPUT);` -- it configures the relays 2's as output.
- `pinMode(LED_ENTRANCE, OUTPUT);` -- it configures this LED as output.
- `pinMode(LED_MAINROOM, OUTPUT);` -- it configures this LED as output.
- `pinMode(LED_KITCHEN, OUTPUT);` -- it configures this LED as output.

- `pinMode(LED_LIVING_ROOM, OUTPUT);` -- it configures this LED as output.
- `pinMode(LED_BATHROOM, OUTPUT);` -- it configures this LED as output.
- `digitalWrite(RELAY_LAMP1_PIN, HIGH);` -- this line initiates the relay 1 in an ON state.
- `digitalWrite(RELAY_LAMP2_PIN, HIGH);` -- this line initiates the relay 2 in an ON state.
- `digitalWrite(LED_BATHROOM, LOW);` -- this line defines that in the beginning of the program, the LED of the bathroom is OFF.

#### ➤ Variables

```
31  bool ldr = false;
32  int ldrStatus = 0;
```

- `bool ldr = false;` -- the variable LDR is declared as a bool which means that it can store any Booleans values (true or false). And initially, it is defined as false that indicated that the extra conditions of the code of LED control are deactivated
- `int ldrStatus = 0;` -- the ldrStatus is declared as a variable of int type (integer) and it starts at value 0.

#### ➤ Loop code

```
75  ldrStatus = analogRead(LIGHT_SENSOR_PIN);
119 //entrance light
120 if (ldrStatus <= 750 || ldr) {
121     digitalWrite(LED_ENTRANCE, HIGH);
122 } else if (ldrStatus > 750 && !ldr) {
123     digitalWrite(LED_ENTRANCE, LOW);
124 }
185     case 'C' :
186         ldr = true;
187         break;
188
189     case 'c' :
190         ldr = false;
191         break;
192 }
193 delay(500);
```

- `ldrStatus = analogRead(LIGHT_SENSOR_PIN);` -- In this line the code uses the command `analogRead()` to read the value of the LDR and the value is stored.
- `if (ldrStatus <= 750 || ldr) {`

`digitalWrite(LED_ENTRANCE, HIGH);` -- this condition is verified if the value of the `ldrStatus` is less or equal to 750 and if the ambience is dark (less or equal to 750), the code executes the command `digitalWrite` that turns on the LED connected to the pin.

- `else if (ldrStatus > 750 && !ldr) {`  
`digitalWrite(LED_ENTRANCE, LOW);` -- this second condition is verified only if the first condition is not met. This code verifies if the value of the LDR status is bigger than 750 and if the variable LDR is deactivated and if both conditions are true (the ambience is clear and LDR is false), the command `digitalWrite` is executed turning off the led.

- `case 'C' :`  
`ldr = true;` -- if the command C is received, the LDR variable is defined as true.

- `break;` -- this line will interrupt the execution of the loop making the program leave the block code and continue the execution of other codes.

- `case 'c' :`  
`ldr = false;` -- if the command c is received, the LDR variable is defined as false.

```
37  char command = 0;
127  if (BTSerial.available() > 0) {
128
129      command = BTSerial.read();
130
131      Serial.println(command);
132      Serial.println("-----");
133      switch (command) {
134          //ENTRANCE, MAINROOM, KITCHEN, LIVING_ROOM, BATHROOM
135
136          case 'M' : //main room light
137              digitalWrite(LED_MAINROOM, HIGH);
138              break;
139          case 'm' :
140              digitalWrite(LED_MAINROOM, LOW);
141              break;
142          case 'K' : //kitchen light
143              digitalWrite(LED_KITCHEN, HIGH);
144              break;
145          case 'k' :
146              digitalWrite(LED_KITCHEN, LOW);
147              break;
148          case 'L' : //living room
149              digitalWrite(LED_LIVING_ROOM, HIGH);
150              break;
151          case 'l' :
152              digitalWrite(LED_LIVING_ROOM, LOW);
153              break;
```

```

154     case 'B' : //bathroom
155         digitalWrite(LED_BATHROOM, HIGH);
156         break;
157     case 'b' :
158         digitalWrite(LED_BATHROOM, LOW);
159         break;
160     case 'R' : //Relay_1
161         digitalWrite(RELAY_LAMP1_PIN, HIGH);
162         break;
163     case 'r' :
164         digitalWrite(RELAY_LAMP1_PIN, LOW);
165         break;
166     case 'Q' : //main room light
167         digitalWrite(RELAY_LAMP2_PIN, HIGH);
168         break;
169     case 'q' :
170         digitalWrite(RELAY_LAMP2_PIN, LOW);
171         break;

```

- `char command = 0;` -- the variable command is of the char type which means it will store characters and the initialization with 0 is a way to define the default value meaning that in the initialization it won't show any character.
- `if (BTSerial.available() > 0) {` -- this function verifies if there is available data for the reading in the serial communication with the Bluetooth Module. And if there is data the code inside the if block will be executed
- `command = BTSerial.read();` -- this function reads, and the first byte of the data received from the Bluetooth module and stores it in the command variable.
- `Serial.println(command);` -- this line will make the program show in the monitor the value receives from the Bluetooth module
- `Serial.println("-----");` -- this prints a separation line in the monitor to facilitate the data visualization.
- `switch (command) {` -- this structure switch verifies the value of the command variable and depending on the value it will execute the cases.
- `case 'M' :`
  - `digitalWrite(LED_MAINROOM, HIGH);` -- if the command is M, the main room's LED is turned on.
- `case 'm' :`
  - `digitalWrite(LED_MAINROOM, LOW);` -- if the command is m, the main room's LED is turned off.
- `case 'K' :`

`digitalWrite(LED_KITCHEN, HIGH);` -- if the command is K, kitchen's LED is turned on.

- `case 'k' :`

`digitalWrite(LED_KITCHEN, LOW);` -- if the command is k, kitchen's LED is turned off.

- `case 'L' : //living room`

`digitalWrite(LED_LIVING_ROOM, HIGH);` -- if the command is L, living room's LED is turned on.

- `case 'l' :`

`digitalWrite(LED_LIVING_ROOM, LOW);` -- if the command is l, living room's LED is turned off.

- `case 'B' : //bathroom`

`digitalWrite(LED_BATHROOM, HIGH);` -- if the command is B, bathroom's LED is turned on.

- `case 'b' :`

`digitalWrite(LED_BATHROOM, LOW);` -- if the command is b, bathroom's LED is turned off.

- `case 'R' : //Relay_1`

`digitalWrite(RELAY_LAMP1_PIN, HIGH);` -- if the command is R, the relay 1 is activated.

- `case 'r' :`

`digitalWrite(RELAY_LAMP1_PIN, LOW);` -- if the command is r, the relay 1 is deactivated.

- `case 'Q' : //main room light`

`digitalWrite(RELAY_LAMP2_PIN, HIGH);` -- if the command is Q the relay 2 is activated.

- `case 'q' :`

`digitalWrite(RELAY_LAMP2_PIN, LOW);` -- if the command is q, the relay 2 is deactivated.

### 3.2.2. Security system

#### ➤ Pin directives

```

9   #define MAGNETIC_SENSOR_PIN 2
10  #define PRESENCE_PIN 3
11  #define BUZZER_PIN 5

```

- `#define MAGNETIC_SENSOR_PIN 2` – it defines that the magnetic sensor pin is connected to the digital pin 2.
- `#define PRESENCE_PIN 3` – It defines that the presence sensor pin is connected to the digital pin 3.
- `#define BUZZER_PIN 5` – it defines that the buzzer pin is connected to the digital pin 5.

#### ➤ Variables

```

28  bool securityMode = false;
29  int doorClosed = 1;
30  int motion = 0;
31  bool ldr = false;
33  bool intruder = false;

```

- `bool securityMode = false;` -- this declares a variable named `securityMode` and declares that it is a `bool` variable (Boolean) and indicating that in the beginning of the program this variable is `false` meaning that the security system is deactivated.
- `int doorClosed = 1;` -- this variable indicates the state of the door showing that the door is initially closed (1).
- `int motion = 0;` -- the motion variable indicates the detection of the motion and as it is 0 showing that in the beginning there is no motion detected.
- `bool intruder = false;` -- the variable `intruder` is a flag to detect if there is an intruder in the house. As is `false`, it shows that at the beginning there is no intruder identified.

#### ➤ Loop code

```

73  doorClosed = digitalRead(MAGNETIC_SENSOR_PIN);
74  motion = digitalRead(PRESENCE_PIN);

173  case 'S': //Security mode
174      Serial.println("Security Mode Activated");
175      securityMode = true;
176      break;
177
178  case 's' :
179      Serial.println("Security Mode Deactivated");
180      securityMode = false;
181      intruder = false;
182      noTone(BUZZER_PIN);
183      break;

```



```

101 // Security mode
102 ✓ if (securityMode && doorClosed == LOW ) {
103     Serial.println("Alert: Door Opened!");
104     intruder = true;
105     tone(BUZZER_PIN, 1000);
106 }
107 ✓ else if (securityMode && motion) {
108     Serial.println("Alert: Somebody in the house!");
109     intruder = true;
110     tone(BUZZER_PIN, 1000);
111 ✓ } else if (intruder) {
112     tone(BUZZER_PIN, 1000);
113 ✓ } else {
114
115     noTone(BUZZER_PIN);
116 }

```

- doorClosed = `digitalRead(MAGNETIC_SENSOR_PIN)`; -- it reads the state of the magnetic sensor to know if it is true or false.
- motion = `digitalRead(PRESENCE_PIN)`; -- it reads the state of the presence sensor to know if it is true or false.
- `case 'S':`

`Serial.println("Security Mode Activated");`

`securityMode = true;` -- If the system receives the message “S” it executes the first part of the block which is print the message "Security Mode Activated" in the monitor and the variable securityMode is configured as true.

- `case 's' :`

`Serial.println("Security Mode Deactivated");`

`securityMode = false;`

`intruder = false;`

`noTone(BUZZER_PIN);` -- If the system receives the message “s” it executes the second part of the block which is print the message "Security Mode deactivated" in the monitor and the variable securityMode is configured as false and the intruder variable is reset to false meaning that any signal of anterior intrusion was deactivated and if the buzzer alarm should stop.

- `if (securityMode && doorClosed == LOW ) {`

`Serial.println("Alert: Door Opened!");`

`intruder = true;`

`tone(BUZZER_PIN, 1000);` -- it verifies if the securityMode is activated and if the

variable doorClosed is low(if the door is opened) and if both variables are true, the system will

print in the monitor "Alert: Door Opened!" and turns the variable intruder as true and it activates the buzzer.

- `else if (securityMode && motion) {`

- `Serial.println("Alert: Somebody in the house!");`

- `intruder = true;`

- `tone(BUZZER_PIN, 1000);` -- It verifies if the securityMode is activated and if the variable motion is true (if motion is detected) and if both variables are true, the system will print in the monitor "Alert: Somebody in the house!" and turns the variable intruder as true and it activates the buzzer.

- `else if (intruder) {`

- `tone(BUZZER_PIN, 1000);` -- If the intruder variable was activated I any of the before verification, the system will keep emitting the buzzer sound alerting continuously to the intrusion. `securityMode = !securityMode;` -- it inverts the actual securityMode variable.

- `else {`

- `noTone(BUZZER_PIN);` -- if none of the conditions before were true, the code deactivates the buzzer.

### 3.2.3. Temperature Reading

#### ➤ Library

```
3  #include "DHT.h"           // temperature sensors
4  #include <Wire.h>           //Display
5  #include <LiquidCrystal_I2C.h> // Display
```

- `#include "DHT.h"` -- This line includes the DHT library which is used to control and read the temperature and humidity sensor.
- `#include <Wire.h>` -- This line includes the wire library that allows communication with the device that uses the I2C (inter-integrated Circuit) which is the LCD display.
- `#include <LiquidCrystal_I2C.h>` -- This line includes the LiquidCrystal\_I2C library used to control the LCD displays.

#### ➤ Pins setup

```
12  #define DHT_PIN 6
24  #define DHTTYPE DHT11
```

```
27   DHT dht(DHT_PIN, DHTTYPE);
```

```
36   float temp = 0;
```

```
45   pinMode(DHT_PIN, INPUT);
```

- `#define DHT_PIN 6` – defines the pin 6 for the DHT11 sensor.
- `#define DHTTYPE DHT11` – defines the DHT sensor type as DHT11.
- `DHT dht(DHT_PIN, DHTTYPE);` – It creates an object called `dht` that represents the temperature sensor.
- `float temp = 0;` -- it declares the variable `temp` of a float type that starts from 0.
- `pinMode(DHT_PIN, INPUT);` -- Defines the DHT pin as an input.

#### ➤ Loop code

```
92   temp = dht.readTemperature();  
93   lcd.setCursor(0, 0);  
94   lcd.print("My Smart Home");  
95   lcd.setCursor(0, 1);  
96   lcd.print("Temperature: ");  
97   lcd.print(temp);  
98   lcd.print("°C");  
99   delay(2000);
```

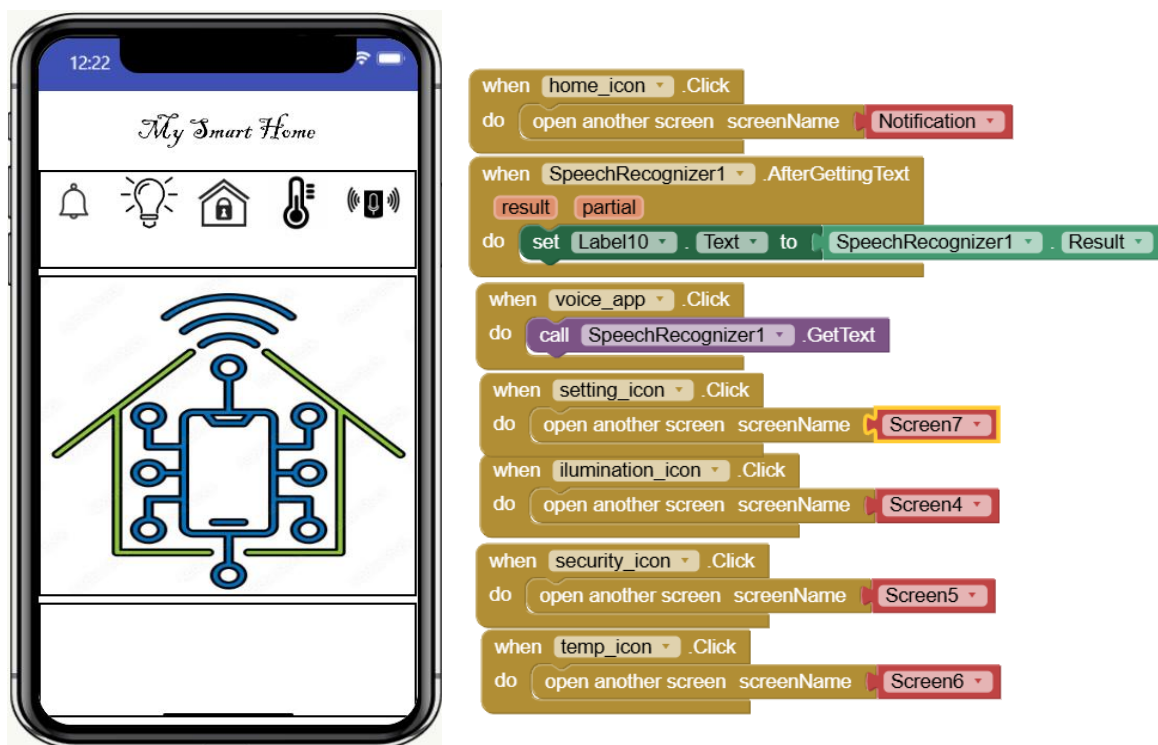
- `temp = dht.readTemperature();` -- it reads the sensor temperature and stores the value of `temp` variable.
- `lcd.setCursor(0, 0);` -- it positions the cursor in the beginning of the first line of the lcd display.
- `lcd.print("My Smart Home");` -- it exhibits the sentence “My smart home” in the first line of the display.
- `lcd.setCursor(0, 1);` -- it positions the cursor in the beginning of the second line of the lcd display.
- `lcd.print("Temperature: ");` -- it exhibits “temperature” in the second line of the lcd display.
- `lcd.print(temp);` -- exhibits the value of the variable `temp` in the second line.
- `lcd.print("°C");` -- adds the °C after the value of temperature read.
- `delay(2000);` -- it waits two seconds before repeating or continuing the code.

### 3.3. App Screens and Codes

- I. Main Screen – in this screen, the tenant can choose what they want to control, and the code are basically just to open the desired page when someone click the icons for example when someone click in the illumination icon, it should open the illumination control screen. And there is a code for the voice recognizer where when someone presses the voice icon, it will activate the voice recognizer and then with the label show the result of what was recognized.

**Figure 20:** Main Page of the App and the Code Blocks

(Source: *Made by me*)

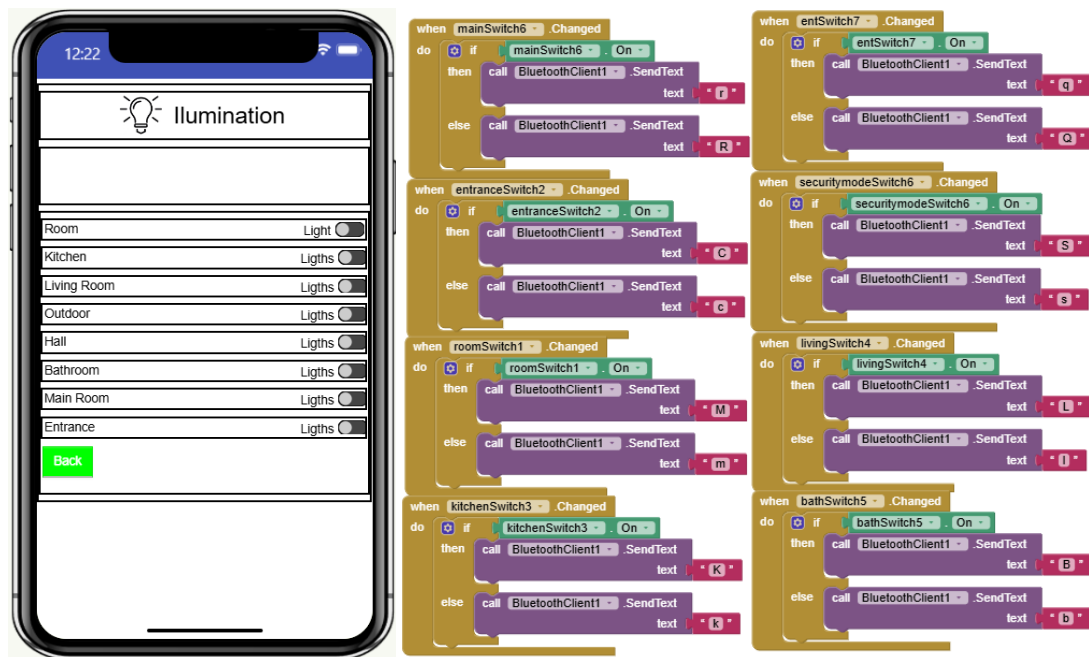


## II. Illumination control

The next screen is the illumination control screen, and it is where the tenant can turn on and off the light remotely. When the tenant turns on the light by changing the switch state, the app communicates through the Bluetooth module sending a message as for example when I press the room's switch, the app sends the message to the Arduino as "M" and as seeing before in the Arduino code, when the Arduino receives this message, it turns on the main room's light.

**Figure 21:** Main Page of the App and the Code Blocks

(Source: *made by me*)

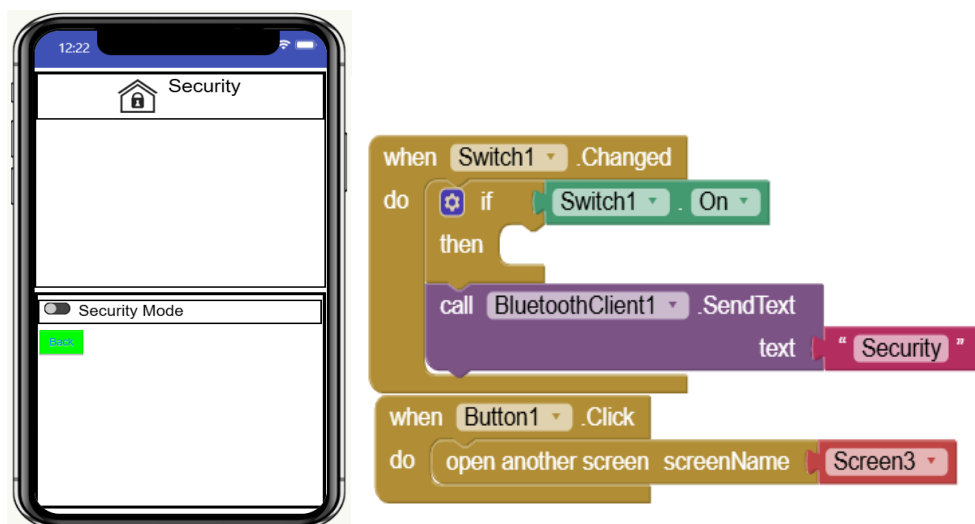


### III. Security Screen

This screen is to activate the security mode or turn it off, and the code says that when the switch changed, and it is on then the app should communicate through the Bluetooth to the Arduino sending a message saying “security” and then the Arduino will understand that the security mode is on and then run the code as the security mode on.

**Figure 22:** Security Page and Code Block

(Source: *Made by me*)

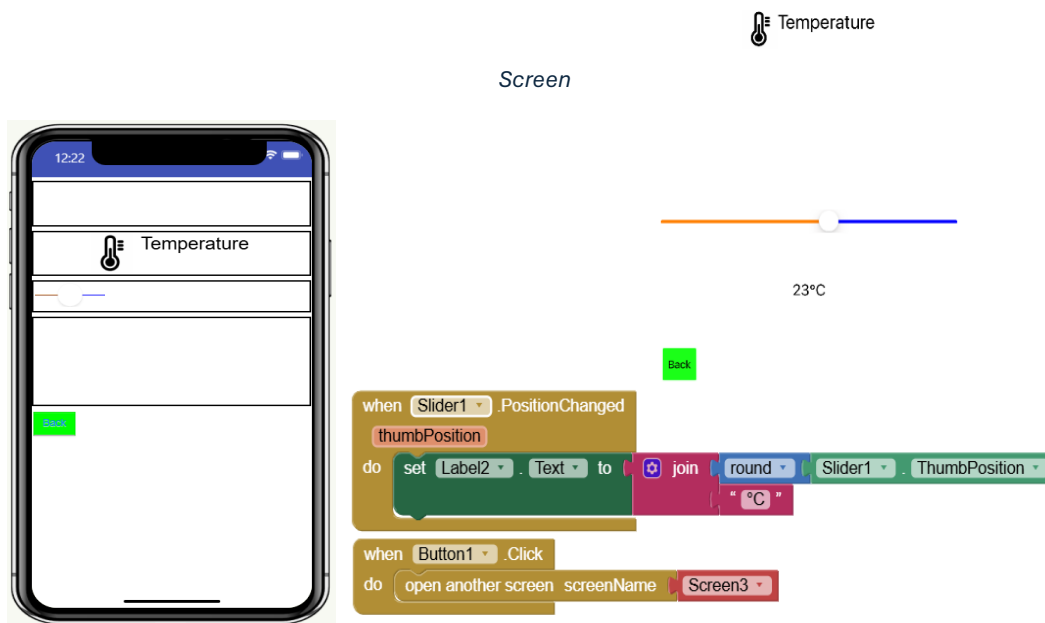


#### IV. Temperature Screen

This is the temperature screen where the original idea was to be able to control the heater/ air conditioner through the application also, and when the slider is changing, it is possible to set the temperature that is wanted in the air conditioner or heater. But this part of the program is not functional as in the project we will not have the controlling of the HVAC, the climatization part as I explained in the Arduino code will be the reading of the temperature and it will be shown in a display.

**Figure 23:** Temperature Control

(Source: *Made by me*)



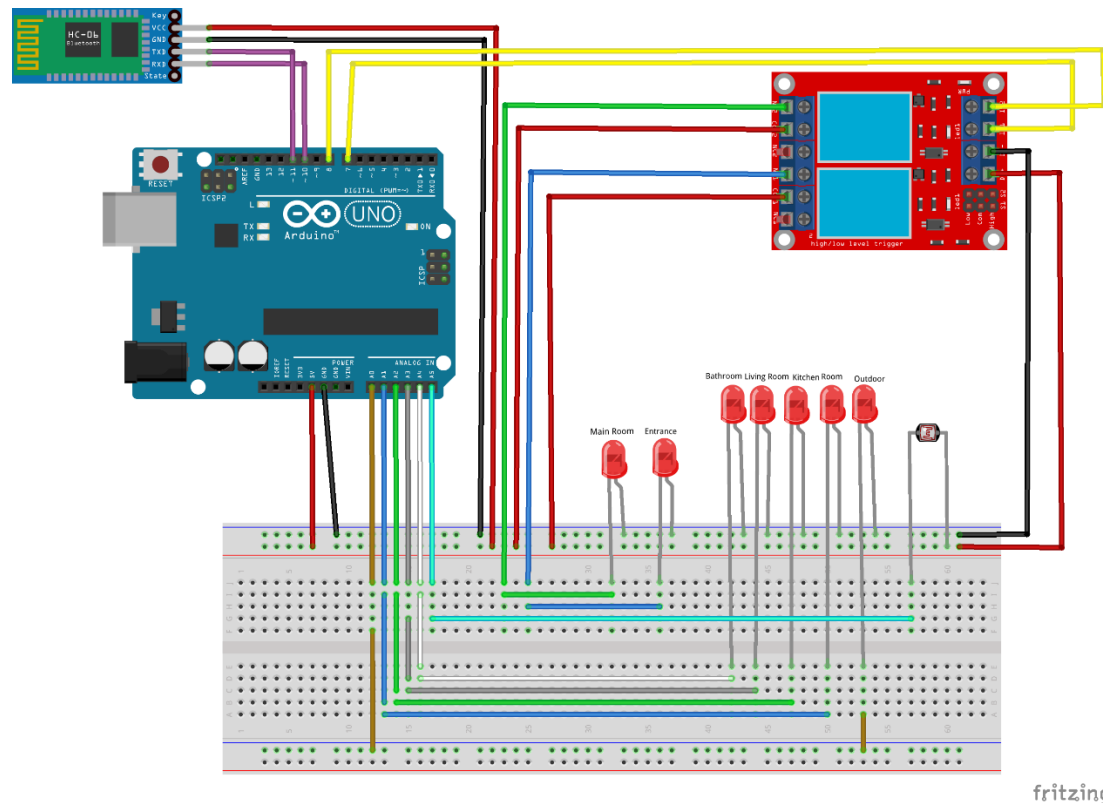
#### 3.4. Schematic of the connections

To make the connections scheme I used the software Fritzing to draw the scheme so it was easier to make the installation, where I could just follow the drawing and avoid complications when doing it.

### 3.4.1. Schematic Diagram of Illumination Circuit

**Figure 24:** Schematic Diagram of Illumination

(Source: *Made by me*)



This scheme shows the way the illumination system was installed with the model. The components are the Bluetooth module, relay, LED, Arduino, LDR sensor and the breadboard for better understanding of the connections. The VCC and Ground of the Bluetooth module are connected to the 5 V and ground of the Arduino to give energy to the module and the TXD pin is connected to 11 pin of the Arduino and RXD is connected to the 10 pin of the Arduino as pre-defined in our code. The relay input (IN1 and IN 2) is connected to the 7 and 8 pins of the Arduino and the DC+ and DC- are connected to the 5 V and Ground of the Arduino so the relay will have the power. The NO1 (normally opened contact of channel 1) pin is connected to the positive leg of the LED that is controlled by the relay. And the other leg of the LED is connected to the ground of the Arduino. Same logic for the other LED that will be connected to the second channel of the relay.

For the Outdoor LED, that has a specification that will be activated with the light sensor so the connection is the negative pole of the LED is connected to the ground and the positive pole is connected to the Arduino pin A0 as defined in the code and the LDR is connected to the A5 of

the Arduino Pin and the other pin is connected to the VCC and the function in this case is determined by the code itself, not the connection.

The COM (common) is connected to the 5V of the Arduino as the COM pin is where the current of the electric power is connected while the NO pin is the output that is connected to the LED.

When the relay is in its initial state, the circuit between the COM and NO is opened meaning that the current is not passing through and the LED is off.

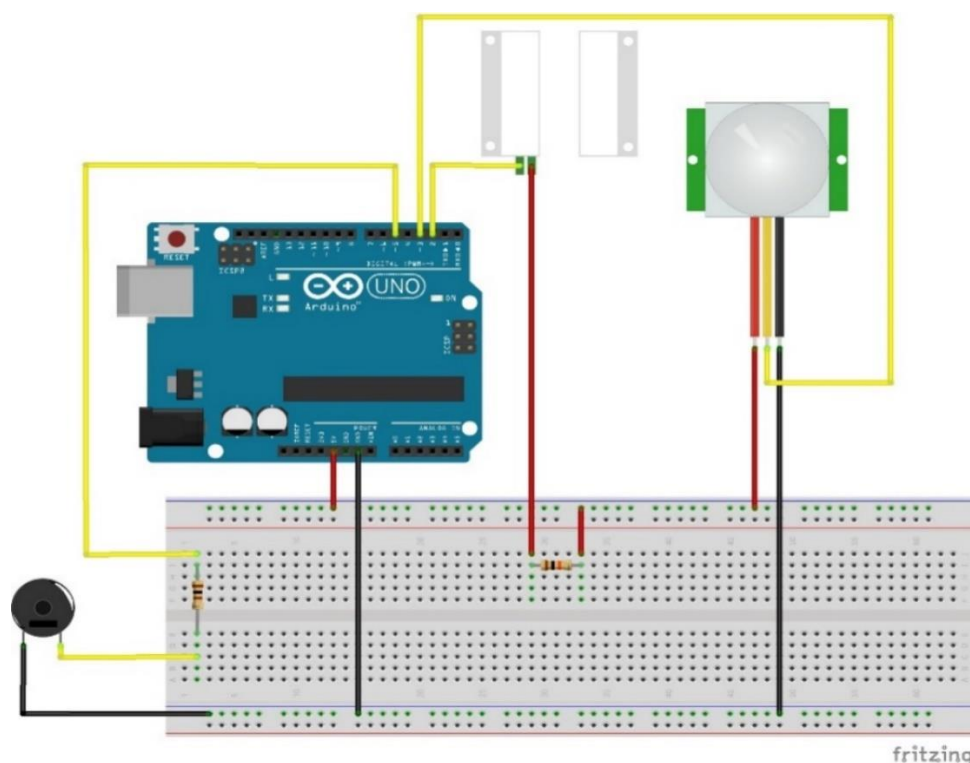
When the signal is sent from the Arduino to the relay, the internal coil of the relay is activated. This generates a magnetic field that will attract moving contact by closing the circuit between the COM and NO making the current pass through and turning on the LED.

The other LEDs are connected to the respective Arduino pin as determined in the code and the other negative leg is connected to the ground and they will be controlled through the app and Bluetooth module. When in the app the bathroom LED is turned on for example, the Bluetooth module will receive a message that will make the LED turn on because it is defined in the code.

#### 3.4.2. Schematic Diagram of Security Circuit

**Figure 25:** Schematic Diagram of Security Circuit

(Source: *Made by Me*)





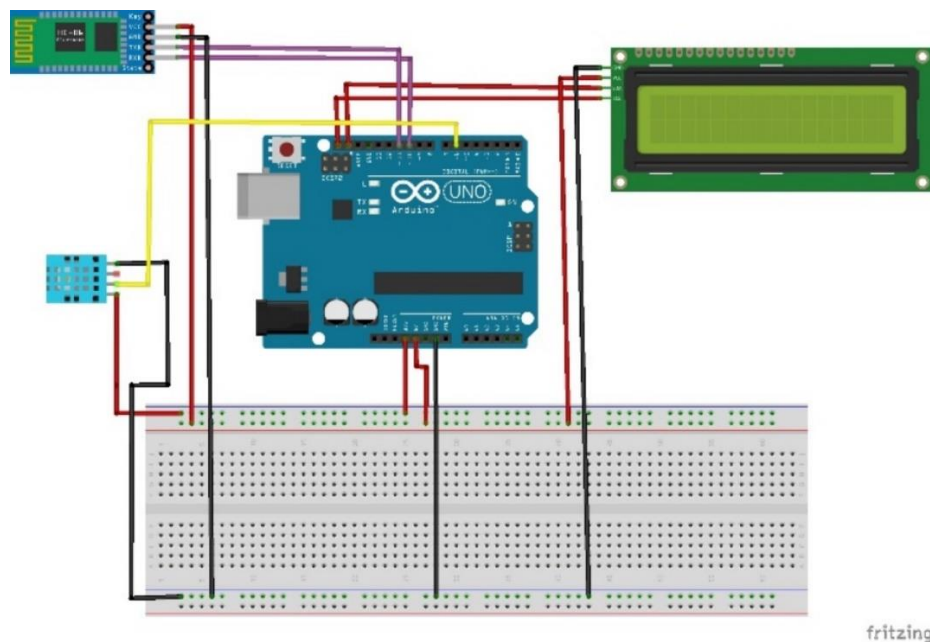
For the security system, I used a buzzer two resistors of 100 and 10k ohm, the presence sensor, magnetic door sensor, Arduino and breadboard for better understanding of the connections. The PIR sensor for presence detection is connected to the Arduino through the signal port and it's connected in the pin 3 of the Arduino as it is determined in the code, and the ground and Vcc are connected to the 5V and ground of the Arduino so it can have the power to work.

The signal pin of the magnetic door sensor is connected to pin 2 of the Arduino as determined by the code and the ground port is connected to the ground of the Arduino and the buzzer as the magnetic door is connected to the Arduino pin 5 and the ground is with the Arduino's ground.

#### 3.4.3. Schematic Diagram of Temperature Circuit

**Figure 26:** Schematic Diagram of the Temperature Reader

(Source: *Made by me*)



In the figure shows the connection that was made with the DHT11, Bluetooth module, LCD display and Arduino. As I showed in the figure, the DHT11 4 ports and I connected the ground port in the ground of Arduino, the power port in the 3 V power of Arduino, and the data signal port is connected in the digital port 6 of the Arduino as in the program it is defined as the DHT11 port. The Bluetooth module is also connected to the Arduino, the voltage and ground pins of the HC-06 are connected to the 5 V and ground of the Arduino and the TX pin (data transmitter) is in the pin 11 of the Arduino and RX (data receiver) is in the pin 10 of the Arduino as pre-determined in the code program.

And the logic follows, to give the power supply to the LCD display, the Vcc and ground are connected to the power and ground of the Arduino and the SDA and SCL of the LCD are connected to the SDA and SCL pins of the Arduino.

### 3.5. Errors

#### Application

When the test where performed, I realized that there were some problems with the communication of Bluetooth and the application. As I describe in the 3.4. *Application* point, I created in the MIT App Inventor and application for control of the illumination, security and climatization in different screens and I put in each page a Bluetooth client interface that would be responsible for the communication between the tenants through the app and the microcontroller.

In the beginning this logic made sense as each system had its characteristics and needed an interface dedicated to control and this would make the app more organized and intuitive and with separated pages it would be easier to explain in this project what the code was and the idea as it would be separated.

As this, I organized the app in different screens but when I was implementing the Bluetooth connection in the testing phase, with the HC-06 Bluetooth module, I found an unexpected challenge.

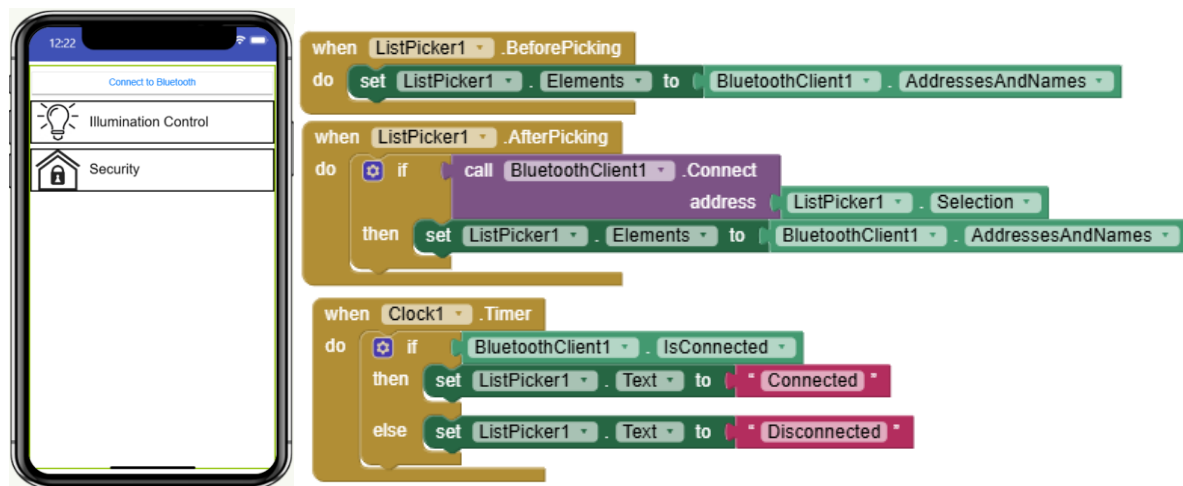
When I tried to connect the Bluetooth module to the app, I saw that the MIT App Inventor ends automatically the Bluetooth connection in one screen when I change the screen to control another system.

This technical limitation prevented the app from maintaining a continuous Bluetooth connection and functional between the screens making it inviable the simultaneous control between the illumination and security system with structure in different screen as planned.

To get around this limitation and make sure that the app would maintain a stable connection with the HC-06, I decided to reformulate the interface instead of separating each system in a different screen, I integrated all of them in one screen and with the codes, instead of navigating in between the screens, the user only alternates the layouts making the system they want to control visible and hiding the other system's layout. This arrangement allows the Bluetooth module to maintain a continuous connection with the app.

**Figure 27:** Home Layout of the App

(Source: *Made by Me*)



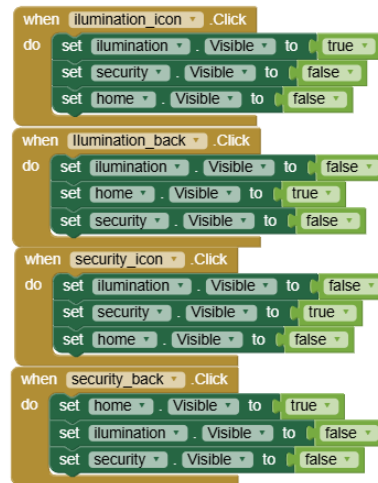
To connect to Bluetooth, I added the list picker interface (Connect to Bluetooth Bottom).

When ListPicker1.BeforePicking – this block shows that before the user selects the item of the pick list the app will define ListPicker1.Elements to BluetoothClient1.AddressesAndNames meaning that the app will search in the mobile device every detectable Bluetooth device, and they respectively name, and this data will be added to the list and the user can visualize it in the screen. And to make the connection, when ListPicker1.AfterPicking defines when the user selects the Bluetooth device the call BluetoothClient1.Connect address ListPicker1.Selection will be activated meaning that app will try to have a connection with the device that was chosen and if the connection is established correctly, the set ListPicker1.Elements to BluetoothClient1.AddressesAndNames will be executed and again, the list will be renewed and this is useful to make sure the list will be always synchronized.

To monitor the connection, I added the clock and the block when Clock1.Timer is executed repetitively in time interval pre-defined allowing the app to monitor continuously the connection state of the Bluetooth and if BluetoothClient1.IsConnected defines that if there is connection with Bluetooth, then the text of the list picker button will change to Connected and if the connection is lost, then the text changes to Not Connected.

**Figure 28:** App Layout configuration Code Block

(Source: *Made by Me*)



To change the layout, I use the logic that when the user is in the home layout, and they click for example, in the illumination icon, then the security layout and home layout visibility will be set to false. This way the only visible layout will be the wanted one in this case the illumination. and this logic was used for the other layout.

And I added a back button in the security and illumination layout, and I used the same logic in both layouts, when the back button is clicked, then the app makes visible the home layout. The block code for illumination and security system is the same that I explained in the 3.4 Application point and the layout is basically the same also.

**Figure 29:** Illumination and security Layout in the app

(Source: *Made by Me*)



### 3.6. Model Result

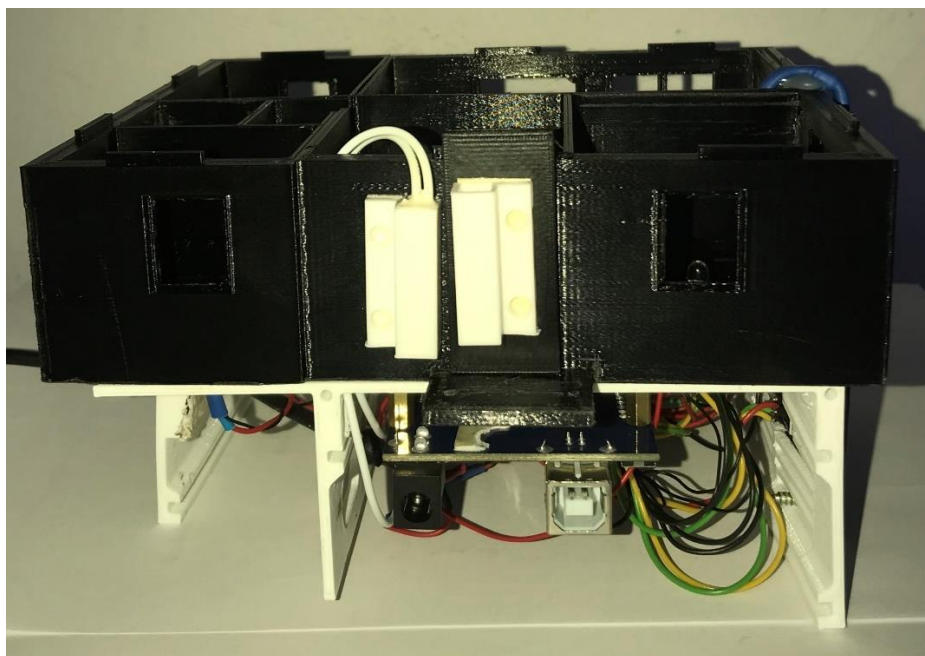
**Figure 30:** LED and the Temperature Sensor Installation

(Source: *By me*)



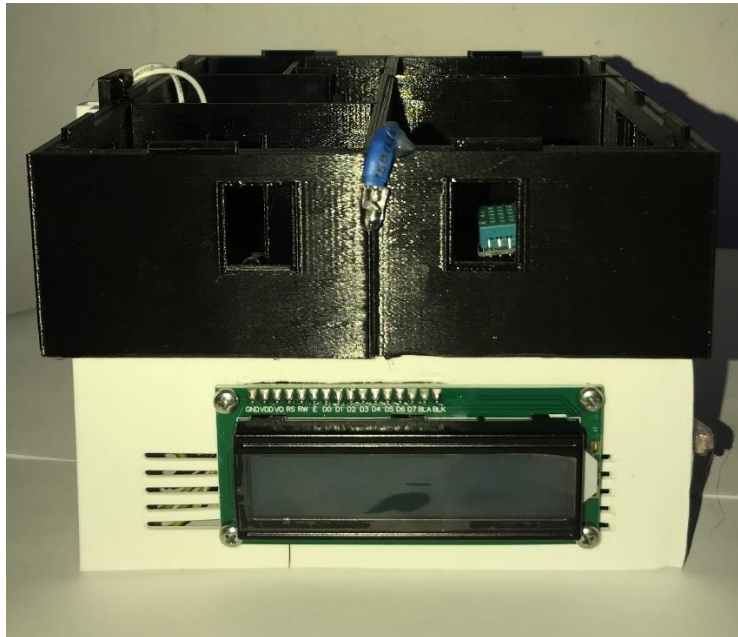
**Figure 31:** Magnetic Sensor Installation in the Door

(Source: *By me*)



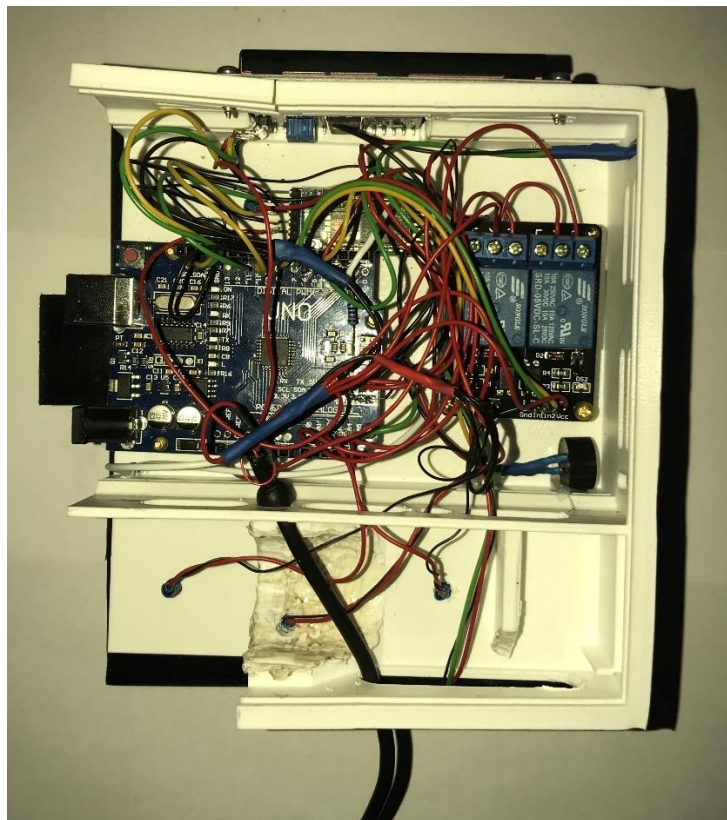
**Figure 32: Display and Outdoor Installation**

(Source: *By me*)



**Figure 33: Wiring Connections**

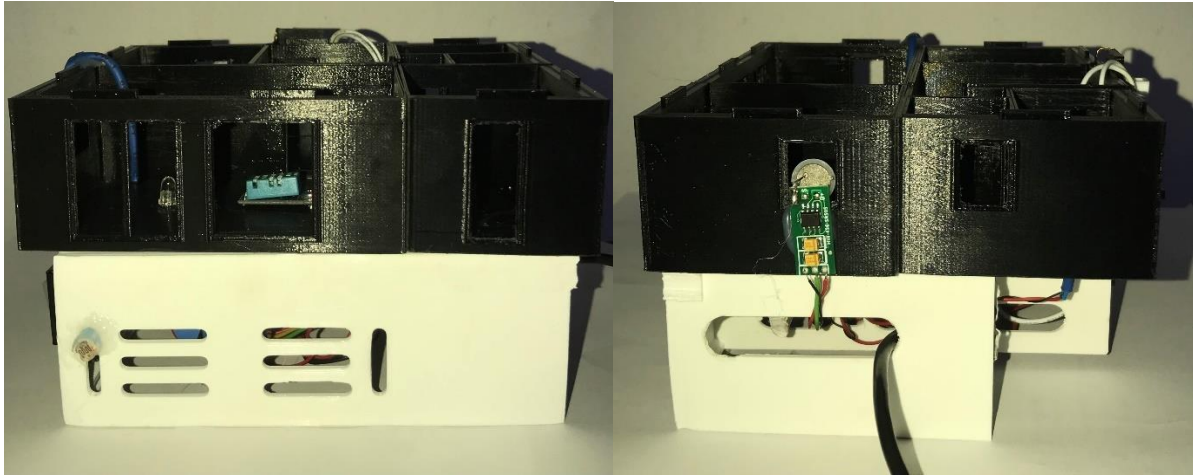
(Source: *By me*)





**Figure 34:** LDR and PIR sensor installation

(Source: *By me*)



The model I developed has many integrated functionalities and aims to give practicality and security to the tenants. The system is controlled by an Arduino Uno, and it manages all the operations, and it communicates with the user through an app.

The illumination system is divide in some sectors as I used a relay to controls two LEDS and when I turn on the LED in the App the relay is activated making the current flow to the led and turning on the light and to control other LED, as the room, bathroom and the others, I used a code where when in the app I switch on the main room's light for example, the app will send a message through the Bluetooth module saying "M" and the code will understand that the LED should be on and then it will execute the code and if I turn off in the app, the app sends a message saying "m" and the microcontroller understands that the LED should be off and then it executes the code making this an effective way to control the lights of my model.

The LDR that is positioned in the outside wall, is connected to the entrance light that is also outside, and this light is more of a presence illumination. When the LDR detects that the ambience is dark as for example when I cover the LDR with my hand, it activates the LED and when I remove my hand, it will turn off the LDR making it a good thing for when it is night to illuminate the house outside without having to manually control it.

For the security system, I installed the magnetic sensor to the door. The Magnet I installed in the door and the sensor is on the wall of the model and when in the app I activate the security mode, when I open the door the buzzer alarm will be activated.

For the temperature monitoring system unfortunately is not functional. After many tests and even a change and reinstalling of the components to check if it was any problem with the components, I unfortunately couldn't find the problem. When I tried the system in an independent way, without the other system, only the temperature monitoring devices, it worked as I planned but as soon as I integrated it with all the systems, the display wouldn't show anything. I connected the lcd with another power supply, so it wasn't the case that the energy wasn't enough for all the systems. And I changed the Arduino and the LCD to check if the defect was with the devices, but it was making the same errors. I made some measurements to check if the current was passing through correctly and everything was good.

I don't consider that the code can be the problem as in an independent way, the code was working correctly so I could presume that the installation is also correct so I can say that the code and the circuit for this specific situation is correct but somehow, when integrated it is not functional.

This technical challenge can suggest that it can have limitations in the processing capacity or conflict between the modules when they are operating simultaneously in the Arduino as the microcontroller has limited resources to deal with multiple systems. As a result, the temperature monitoring system couldn't be integrated into the final model of the smart home even though it works correctly in an independent way.

### 3.7. Cost Analysis

The cost analysis of this project is about the evaluation of the detailed cost of the needs to implement this project. In these types of projects, the cost can vary considerably depending on the elements or the specifications of the project. But as I quoted in the introduction, one of the aims of this work is to have a low-cost project or at least reasonable because to do any project, the cost should be considered because sometimes it might not make sense to have a project if financially it is not worth it.

To calculate the total cost to implement this project, I chose the **HESTORE** store - <https://www.hestore.hu/> to buy the components.



**Table 2:** Price of the Components according to HESTORE Store(Source: *Made by me*)

Element	Description	Number of elements	Price HUF
Arduino UNO	ARCL-UNO-CH340	1	2139
PIR sensor	PIRMINI-SR505	1	511
LDR sensor	DS5-5528-LDR	1	83.15
Magnetic sensor	CD100S	1	2581
Buzzer	SFN-1207PA5.0	1	176
Temperature sensor	DHT11-M0	1	643
Bluetooth Module	MHC-06/6	1	2827
Relay Module	RELC-2CH	1	888
LED	OSW5DK3131A-12V	7	159
100 $\Omega$ resistor	100 R 1%	1	27,33
10000 $\Omega$ resistor	10 K 1%	1	28,61
<b>Total</b>	11 017.09 HUF		

I consulted the site to check the prices on November 1<sup>st</sup> of 2024.

This table lists the main electronic components used in the creating of this model of smart home that are important for the device system's automatization. It is important to mention that the table only brings the cost of the basic and visible components excluding additional materials like wires, connectors and all the necessary material for the complete installation of the model. Besides that, the real implementation of this project would have a bigger cost as it would require components with bigger potency and capacity to make the operation continuous.

The total price justifies the project as the implementation of it brings comfort, and energetic efficiency and another point is security, which is an important part of the system.

The automatization of the house adds more value to the property, giving it a differential in the mobiliary market and by this way the by making this project to your house, it doesn't not only contribute to the experience, but it also adds patrimonial value.

#### 4. Conclusion

In this project I developed a system of residential automatization, and many technologic components were used with the goal of creating a more secure, efficient and comfortable for the tenants. The research and model construction allowed me to have a practical understanding of the advantages and challenges of house automatization, especially referring to the use of microcontrollers as an accessible solution for modern residences.

Throughout the project, I worked on the implementation of the essential functionalities as automatic illumination control that uses light sensors and remote control of the lights through an app. This system makes the routine of the people that live in the house easy, and it contributes to energy consumption.

Security was another important system. I used movement sensors that detect suspect activities when the security mode is on, send immediate alerts to the property owner by mobile devices and start a sound alarm through the buzzer. This mechanism doesn't only give extra protection but also allows to have a fast answer to possible threats increasing the tranquility in the tenants.

Another initial goal was to implement control of the Heating, Ventilation and Air Conditioning system (HVAC). I wanted to control the air conditioner directly, for example to regulate the ambient temperature. But for the high cost and the complexity of the technic, I chose to simplify the system. I made a more accessible solution by integrating a temperature sensor that monitors the ambience and exhibits the data in a display. This adaptation gives the tenants a clear notion of the internal temperature allowing manual adjustment of the climatization devices as needed and it maintains a low budget.

Another resource that was initially planned but not implemented was the voice command to execute specific tasks. I wanted the voice assistant to control some features of the automatization but because of technical limitations and compatibility with the hardware this system functionality wasn't complete in this project. I chose to only include the idea of this integration as a future proposal suggesting a potential expansion of this system which will make this automatization more practical and intuitive for the users.

Even though the interface to monitor energy consumption was an important initial idea, I couldn't implement this system because of time and resources limitations. This idea was to allow the tenants to visualize the parts where there is more energy consumption and allow them

to make decisions to decrease the cost and avoid waste of energy. This system remains as a proposal for future versions of this system.

For better visualization and to improve my knowledge in a practical way, I worked in a functional model of a residence, and it was printed using 3D printing machine. I made tests to evaluate the efficiency of each component and the integration between the systems which proved the efficiency and applicability of proposed system. The practical step was very important to see the viability of the project in a real scenario and another important thing that the model gave was the possibility to adjust, optimize the systems and make a functional project.

By the end of this work, I could conclude that residential automatization based on microcontrollers is not only a viable and economic solution but also is a technology with big potential to make smart residences and sustainable. This project showed that it is possible, with relatively low investment, to make common residences in automatization ambient that promote security, comfort and energy economy.

I also could conclude that the fast development the IoT technology can be given, in parts to the low cost compared to the traditional automatization systems. Instead of using big components like control boxes, physical support and big wires, the IoT devices are projected to be small, modular and with minimal installation. This device has wireless communication deleting the need for complex wiring and minimizing significantly the cost of the system. I realized that making the same system with other components could bring a 10 times more expensive price compared to the price showed the table 2 (11 017.09 HUF).

I conclude my initial goals and adapt other goals, and this project opens ways for future innovations and improvements in the residential automatization area. With the fast evolution of technology, new opportunities appear to make these systems even more accessible and efficient. My experience in electronics and technical projects was essential for the success of this work and the knowledge gathered during the development of this system motivated me to explore even more this field seeking to contribute to solutions that gives practicality and sustainability for people's daily life.

## **SUMMARY**

This thesis presents the development of a residential automatization system with the goal to improve the security, the efficiency and comfort of a modern house. This project uses microcontroller technology to create an accessible solution and versatile solution for residential automatization. The main goal is the integration of many technological components such as illumination control, security measures and climatization control.

The chapter starts with the presentation of the project's context and with the justification of the chosen topic following for the presentation of the literature.

The first step to this project's development was the research and the creation of a solid theory base. With research based on microcontrollers, concepts about this subject were revised. And the literature review allowed us to identify the best practices and the adequate technologies for the system's development and to understand the limitations and challenges that could appear.

After that analysis, I chose the components for the system that I wanted to project. The choice of microcontrollers, sensors and modules was based on cost, availability, compatibility and efficiency in data collection.

The next step was to develop the software. I projected the software to collect and manage the data received from the components. This involved the programming of the microcontroller.

After the software, it is the application development time to control remotely the system and to maintain communication between the human and software. I created the application to be intuitive and allow the user to control the system.

The last step involved physical model creation so I could install the components, and check if it is a functional system. Using 3D printing, I made a residential model where I installed the sensors, actuator, LED and microcontroller, in a strategic and trying to look like reality to optimize the data collection. The installation was followed by practical tests to guarantee the

efficiency of communication and the functionality of the system. After the tests and installation, the system was adjusted to improve the performance of the system.

The conclusion shows that residential automatization using microcontrollers is a practical solution and economic with a substantial potential to create smart ambience and sustainability. With the fast evolution of technology, more opportunities come to the improvement and accessibility in smart home systems making ways for continuous exploration in this dynamic field.

#### 4.1. References

1. IEEE RAS UFCG (2020). “*O Que É Um Microcontrolador?*” consulted on 07/07/2024. Source: <https://edu.ieee.org/br-ufcgras/o-que-e-um-microcontrolador/>
2. Clovis Estamos [N.d]. “*Qual a diferença entre Microprocessador e Microcontrolador* ” consulted on 17/10/2024. Source: <https://blog.novaeletronica.com.br/qual-diferenca-entre-microprocessador-e-microcontrolador/>
3. Steve Aycock(2021).” *A história dos microcontroladores*” consulted on 07/07/2024. Source: [https://www.ehow.com.br/historia-microcontroladores-info\\_42970/#google\\_vignette](https://www.ehow.com.br/historia-microcontroladores-info_42970/#google_vignette)
4. Bit by Bit: Embedded Systems [N.d]. “*Estrutura básica de um microcontrolador genérico*”. Consulted on 07/07/2024. Source: <https://bit-by-bit.gitbook.io/embedded-systems/sistemas-microcontrolados/estrutura-basica-de-um-microcontrolador-generico>
5. PCB Portugal [N.d].” *Fundamentos do microcontrolador: estrutura, princípio de funcionamento e aplicações*”. Consulted on 07/07/2024. Source: <https://pcbportugal.com/Fundamentos-do-microcontrolador-estrutura-principio-de-funcionamento-e-aplicacoes.html>
6. Fábio Guimarães (2020).” *Arquitetura de um microcontrolador – Aula 3 – MC*”. Consulted on 07/07/2024. Source: <https://mundoprojetado.com.br/arquitetura-de-um-microcontrolador/>
7. Rosana Guse (2013).” *Controlando um motor DC com driver Ponte H L298N*” consulted on: 10/07/2024. Source: <https://www.makerhero.com/blog/motor-dc-arduino-ponte-h-l298n/>
8. Sebastián Vidal (2023).” *Sistemas de controle O que é? tipos, função e muito mais*”. Consulted on 10/07/2024. Source: <https://tecnobits.com/pt/sistemas-de-controle%2C-quais-s%C3%A3o-tipos%2C-fun%C3%A7%C3%B5es-e-muito-mais/>
9. Rafael Levi (2023).” *Automação com Arduino — Parte 3: Sensores e atuadores*”. Consulted on 10/07/2024. Source: <https://medium.com/@rafaellevisa/automa%C3%A7%C3%A3o-com-arduino-parte-3-sensores-e-atuadores-31b5aec34117>
10. Rehana Atar (2023).” *Internet of Things (IoT) with Microcontrollers: An Introduction*” consulted on 10/07/2024. Source: <https://ra-electronics.com/internet-of-things-iot-with-microcontrollers-introduction/>

11. Acervo Lima [N.d], “*Vantagens e desvantagens do microcontrolador*”. Consulted on: 10/07/2024 Source: <https://acervolima.com/vantagens-e-desvantagens-do-microcontrolador/>
12. Francesco Sacco (2014).” *Comunicação SPI – Parte I*”. Consulted on 17/10/2024. Source: <https://embarcados.com.br/spi-parte-1/>
13. Leandro Roisenberg (2023).” *Entenda a UART: Guia Completo sobre Comunicação Serial e Aplicações em Eletrônica*”. Consulted on: 17/10/2024. Source: <https://blog.lri.com.br/entenda-a-uart-guia-completo-sobre-comunicacao-serial-e-aplicacoes-em-eletronica/>
14. Eletronica Marker (2024).” *Como Funciona o Protocolo I2C: O Guia Completo para Entender a Comunicação Serial*”. Consulted on: 17/10/2024. Source: <https://eletronicamaker.com/como-funciona-o-protocolo-i2c/>
15. Eletronica Marker (2024).” *Como Funciona o Protocolo SPI?*”. Consulted on: 17/10/2024. Source: <https://eletronicamaker.com/como-funciona-o-protocolo-spi/>
16. Web Automotivo [N.d].” *Rede CAN – O que é e como funciona*”. Consulted on: 17/10/2024. Source: <https://www.webautomotivo.com.br/rede-can-o-que-e-e-como-funciona/>
17. Cloudflare [N.d]. “*What is latency? | How to fix latency*”. Consulted on; 17/10/2024. Source: <https://www.cloudflare.com/learning/performance/glossary/what-is-latency/>
18. Knauf Industries (2023).” *Robôs industriais - desenvolvimento da robotização da produção na indústria automotiva*”. Consulted on: 10/07/2024. Source: <https://knaufautomotive.com/pt-br/robos-industriais-desenvolvimento-da-robotizacao-da-producao-na-industria/>
19. Innovar Controls (2024).” *O que são sistemas de automação HVAC e como funcionam?*”.consulted on: 10/07/2024. Source: <https://innovarcontrols.com/o-que-sao-sistemas-de-automacao-hvac/>
20. Pedro Estolano de Oliveira Filho (2024).” *DESENVOLVIMENTO DE PROTÓTIPO UTILIZANDO MICROCONTROLADORES PARA CONTROLE DE EFICIÊNCIA ENERGÉTICA EM RESIDÊNCIAS UNIFAMILIARES*”. DOI:10.69849/revistaft/pa10202409241512. Consulted on 17/10/2024. Source: <https://revistaft.com.br/desenvolvimento-de-prototipo-utilizando-microcontroladores-para-controle-de-eficiencia-energetica-em-residencias-unifamiliares/>

21. Adam J. Fleischer (2024). “10 principales tendencias en microcontroladores para 2024”. Consulted on 07/07/2024. Source: <https://resources.altium.com/es/p/10-top-trends-microcontrollers-2024>
22. Octopart [N.d].” *Microchip ATMEGA328P-PU Datasheet*”. Consulted on: 10/07/2024. Source: [https://octopart.com/datasheet/atmega328p-pu-microchip-77760224?msclkid=fb7d7101e59c1bab7dff178f61ee1380&utm\\_source=bing&utm\\_medium=cpc&utm\\_campaign=b\\_cpc\\_emea-hu\\_search\\_dsa\\_english\\_en\\_usd\\_all-categories&utm\\_term=semiconductors&utm\\_content=Discrete%20Semiconductors%20DSA](https://octopart.com/datasheet/atmega328p-pu-microchip-77760224?msclkid=fb7d7101e59c1bab7dff178f61ee1380&utm_source=bing&utm_medium=cpc&utm_campaign=b_cpc_emea-hu_search_dsa_english_en_usd_all-categories&utm_term=semiconductors&utm_content=Discrete%20Semiconductors%20DSA)
23. Octopart [N.d].” *Microchip PIC16F877A-I/P Datasheet*”. Consulted on 10/07/2024. Source: <https://octopart.com/datasheet/pic16f877a-i%2Fp-microchip-484833>
24. Octopart [N.d].” *Arduino NANO RP2040 CONNECT WITHOUT HEADERS Datasheet*”. Consulted on 10/07/2024. Source: [https://octopart.com/datasheet/nano+rp2040+connect+without+headers-arduino-118546171?msclkid=4d0c4c92196419abf47a2a1c98112869&utm\\_source=bing&utm\\_medium=cpc&utm\\_campaign=b\\_cpc\\_emea-hu\\_search\\_dsa\\_english\\_en\\_usd\\_all-categories&utm\\_term=octopart&utm\\_content=All%20Pages%20DSA](https://octopart.com/datasheet/nano+rp2040+connect+without+headers-arduino-118546171?msclkid=4d0c4c92196419abf47a2a1c98112869&utm_source=bing&utm_medium=cpc&utm_campaign=b_cpc_emea-hu_search_dsa_english_en_usd_all-categories&utm_term=octopart&utm_content=All%20Pages%20DSA)
25. STMicroelectronics (2011).” *STM32 F4 series High-performance Cortex-M4 MCU*”. Consulted on: 10/07/2024. Source: [https://media.digikey.com/pdf/Data%20Sheets/ST%20Microelectronics%20PDFS/STM32F4\\_BR.pdf](https://media.digikey.com/pdf/Data%20Sheets/ST%20Microelectronics%20PDFS/STM32F4_BR.pdf)
26. Octopart [N.d]. “*Microchip ATMEGA2560-16AU Datasheet*”. Consulted on 10/07/2024. Source: [https://octopart.com/datasheet/atmega2560-16au-microchip-77760073?msclkid=51d199f16f5b183feda6ff7fe4365d0e&utm\\_source=bing&utm\\_medium=cpc&utm\\_campaign=b\\_cpc\\_emea-hu\\_search\\_dsa\\_english\\_en\\_usd\\_all-categories&utm\\_term=semiconductors&utm\\_content=Discrete%20Semiconductors%20DSA](https://octopart.com/datasheet/atmega2560-16au-microchip-77760073?msclkid=51d199f16f5b183feda6ff7fe4365d0e&utm_source=bing&utm_medium=cpc&utm_campaign=b_cpc_emea-hu_search_dsa_english_en_usd_all-categories&utm_term=semiconductors&utm_content=Discrete%20Semiconductors%20DSA)
27. Octopart [N.d].” *Espressif Systems ESP32-S2-MINI-1-N4 Datasheet*”. Consulted on 10/07/2024. Source: [https://octopart.com/datasheet/esp32-s2-mini-1-n4-espressif+systems-117306146?msclkid=ceb87436920f1552659e12b81fafe449&utm\\_source=bing&utm\\_medium=cpc&utm\\_campaign=b\\_cpc\\_emea-hu\\_search\\_dsa\\_english\\_en\\_usd\\_all-categories&utm\\_term=semiconductors&utm\\_content=Discrete%20Semiconductors%20DSA](https://octopart.com/datasheet/esp32-s2-mini-1-n4-espressif+systems-117306146?msclkid=ceb87436920f1552659e12b81fafe449&utm_source=bing&utm_medium=cpc&utm_campaign=b_cpc_emea-hu_search_dsa_english_en_usd_all-categories&utm_term=semiconductors&utm_content=Discrete%20Semiconductors%20DSA)



28. Texas Instruments [N.d]. “*MSP430x11x2, MSP430x12x2 Mixed Signal Microcontroller*”. Consulted on 10/07/2024. Source: <https://www.ti.com/lit/ds/symlink/msp430f1232.pdf?ts=1730674687126>
29. Atmel Corporation [N.d].” *ARM-Based Microcontroller DATASHEET*”. Consulted on 10/07/2024. Source: [https://www.keil.com/dd/docs/datashts/atmel/samd20/atmel-42129-sam-d20\\_datasheet.pdf](https://www.keil.com/dd/docs/datashts/atmel/samd20/atmel-42129-sam-d20_datasheet.pdf)

#### 4.2. List of Figures

<b>Figure 1 :</b> Architecture of PIC16F887 Microcontroller .....	8
<b>Figure 2:</b> Communication UART .....	12
<b>Figure 3:</b> Communication in I2C Protocol Communication .....	13
<b>Figure 4:</b> Communication of the SPI Protocol .....	15
<b>Figure 5:</b> CAN Communication Protocol.....	16
<b>Figure 6:</b> Arduino UNO .....	24
<b>Figure 7:</b> DHT11 Sensor .....	26
<b>Figure 8:</b> Magnetic Sensor for Doors .....	27
<b>Figure 9:</b> Light Sensor .....	29
<b>Figure 10:</b> Presence Sensor .....	30
<b>Figure 11:</b> Buzzer .....	31
<b>Figure 12:</b> Bluetooth Module.....	32
<b>Figure 13:</b> Relay Module .....	33
<b>Figure 14:</b> System Design .....	35
<b>Figure 15:</b> App Starting Screen .....	36
<b>Figure 16:</b> 3D Drawing of the House .....	37
<b>Figure 17:</b> Workpiece Printed Wrong .....	38
<b>Figure 18:</b> Model for 3D Printing .....	39
<b>Figure 19:</b> Room Printed Wrong .....	39
<b>Figure 20:</b> Main Page of the App and the Code Blocks .....	50
<b>Figure 21:</b> Main Page of the App and the Code Blocks .....	51
<b>Figure 22:</b> Security Page and Code Block.....	51
<b>Figure 23:</b> Temperature Control .....	52
<b>Figure 24:</b> Schematic Diagram of Illumination.....	53

<b>Figure 25:</b> Schematic Diagram of Security Circuit .....	54
<b>Figure 26:</b> Schematic Diagram of the Temperature Reader .....	55
<b>Figure 27:</b> Home Layout of the App .....	57
<b>Figure 28:</b> App Layout configuration Code Block.....	58
<b>Figure 29:</b> Illumination and security Layout in the app .....	58
<b>Figure 30:</b> LED and the Temperature Sensor Installation .....	59
<b>Figure 31:</b> Magnetic Sensor Installation in the Door.....	59
<b>Figure 32:</b> Display and Outdoor Installation.....	60
<b>Figure 33:</b> Wiring Connections .....	60
<b>Figure 34:</b> LDR and PIR sensor installation .....	61

#### 4.3. List of tables

<b>Table 1:</b> Microcontrollers and their Parameters.....	20
<b>Table 2:</b> Price of the Components according to HESTORE Store .....	63

#### 4.4. List of Annexes

<b>Annex 1:</b> ATmega328P Microcontroller Datasheet .....	73
<b>Annex 2:</b> PIC16F877A Microcontroller Datasheet .....	75
<b>Annex 3:</b> NANO RP2040 Microcontroller Datasheet.....	77
<b>Annex 4:</b> STM32F4 Microcontroller Datasheet.....	78
<b>Annex 5:</b> ATmega2560 Microcontroller Datasheet .....	79
<b>Annex 6:</b> ESP32C3 Microcontroller Datasheet .....	80
<b>Annex 7:</b> MSP430F1232 Microcontroller Datasheet.....	81
<b>Annex 8:</b> ATSAMD20J Microcontroller Datasheet.....	82
<b>Annex 9:</b> Arduino IDE Code .....	83



## ATmega48A/PA/88A/PA/168A/PA/328/P

### ATMEL 8-BIT MICROCONTROLLER WITH 4/8/16/32KBYTES IN-SYSTEM PROGRAMMABLE FLASH

#### DATASHEET

#### Features

- High Performance, Low Power Atmel®AVR® 8-Bit Microcontroller Family
- Advanced RISC Architecture
  - 131 Powerful Instructions – Most Single Clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 20 MIPS Throughput at 20MHz
  - On-chip 2-cycle Multiplier
- High Endurance Non-volatile Memory Segments
  - 4/8/16/32KBytes of In-System Self-Programmable Flash program memory
  - 256/512/512/1KBytes EEPROM
  - 512/1K/1K/2KBytes Internal SRAM
  - Write/Erase Cycles: 10,000 Flash/100,000 EEPROM
  - Data retention: 20 years at 85°C/100 years at 25°C<sup>(1)</sup>
  - Optional Boot Code Section with Independent Lock Bits
    - In-System Programming by On-chip Boot Program
    - True Read-While-Write Operation
  - Programming Lock for Software Security
- Atmel® QTouch® library support
  - Capacitive touch buttons, sliders and wheels
  - QTouch and QMatrix® acquisition
  - Up to 64 sense channels
- Peripheral Features
  - Two 8-bit Timer/Counters with Separate Prescaler and Compare Mode
  - One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
  - Real Time Counter with Separate Oscillator
  - Six PWM Channels
  - 8-channel 10-bit ADC in TQFP and QFN/MLF package
    - Temperature Measurement
  - 6-channel 10-bit ADC in PDIP Package
    - Temperature Measurement
  - Programmable Serial USART
  - Master/Slave SPI Serial Interface
  - Byte-oriented 2-wire Serial Interface (Philips I<sup>2</sup>C compatible)
  - Programmable Watchdog Timer with Separate On-chip Oscillator
  - On-chip Analog Comparator
  - Interrupt and Wake-up on Pin Change

Atmel-62711-AVR-ATmega-Datasheet\_102014

- Special Microcontroller Features
  - Power-on Reset and Programmable Brown-out Detection
  - Internal Calibrated Oscillator
  - External and Internal Interrupt Sources
  - Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby, and Extended Standby
- I/O and Packages
  - 23 Programmable I/O Lines
  - 28-pin PDIP, 32-lead TQFP, 28-pad QFN/MLF and 32-pad QFN/MLF
- Operating Voltage:
  - 1.8 - 5.5V
- Temperature Range:
  - -40°C to 85°C
- Speed Grade:
  - 0 - 4MHz@1.8 - 5.5V, 0 - 10MHz@2.7 - 5.5V, 0 - 20MHz @ 4.5 - 5.5V
- Power Consumption at 1MHz, 1.8V, 25°C
  - Active Mode: 0.2mA
  - Power-down Mode: 0.1µA
  - Power-save Mode: 0.75µA (Including 32kHz RTC)

## Annex 2: PIC16F877A Microcontroller Datasheet



# PIC16F87XA

## 28/40/44-Pin Enhanced Flash Microcontrollers

### Devices Included in this Data Sheet:

- PIC16F873A
- PIC16F874A
- PIC16F874A
- PIC16F877A

### High-Performance RISC CPU:

- Only 35 single-word instructions to learn
- All single-cycle instructions except for program branches, which are two-cycle
- Operating speed: DC – 20 MHz clock input  
DC – 200 ns instruction cycle
- Up to 8K x 14 words of Flash Program Memory,  
Up to 368 x 8 bytes of Data Memory (RAM),  
Up to 256 x 8 bytes of EEPROM Data Memory
- Pinout compatible to other 28-pin or 40/44-pin  
PIC16CXXX and PIC16FXXX microcontrollers

### Peripheral Features:

- Timer0: 8-bit timer/counter with 8-bit prescaler
- Timer1: 16-bit timer/counter with prescaler,  
can be incremented during Sleep via external  
crystal/clock
- Timer2: 8-bit timer/counter with 8-bit period  
register, prescaler and postscaler
- Two Capture, Compare, PWM modules
  - Capture is 16-bit, max. resolution is 12.5 ns
  - Compare is 16-bit, max. resolution is 200 ns
  - PWM max. resolution is 10-bit
- Synchronous Serial Port (SSP) with SPI  
(Master mode) and I<sup>2</sup>C™ (Master/Slave)
- Universal Synchronous Asynchronous Receiver  
Transmitter (USART/SCI) with 9-bit address  
detection
- Parallel Slave Port (PSP) – 8 bits wide with  
external RD, WR and CS controls (40/44-pin only)
- Brown-out detection circuitry for  
Brown-out Reset (BOR)

### Analog Features:

- 10-bit, up to 8-channel Analog-to-Digital  
Converter (A/D)
- Brown-out Reset (BOR)
- Analog Comparator module with:
  - Two analog comparators
  - Programmable on-chip voltage reference  
(VREF) module
  - Programmable input multiplexing from device  
inputs and internal voltage reference
  - Comparator outputs are externally accessible

### Special Microcontroller Features:

- 100,000 erase/write cycle Enhanced Flash  
program memory typical
- 1,000,000 erase/write cycle Data EEPROM  
memory typical
- Data EEPROM Retention > 40 years
- Self-reprogrammable under software control
- In-Circuit Serial Programming™ (ICSP™)  
via two pins
- Single-supply 5V In-Circuit Serial Programming
- Watchdog Timer (WDT) with its own on-chip RC  
oscillator for reliable operation
- Programmable code protection
- Power saving Sleep mode
- Selectable oscillator options
- In-Circuit Debug (ICD) via two pins

### CMOS Technology:

- Low-power, high-speed Flash/EEPROM  
technology
- Fully static design
- Wide operating voltage range (2.0V to 5.5V)
- Commercial and Industrial temperature ranges
- Low-power consumption

Device	Program Memory		Data SRAM (Bytes)	EEPROM (Bytes)	I/O	10-bit A/D (ch)	CCP (PWM)	MSSP		USART	Timers 8/16-bit	Comparators
	Bytes	# Single Word Instructions						SPI	Master I <sup>2</sup> C			
PIC16F873A	7.2K	4096	192	128	22	5	2	Yes	Yes	Yes	2/1	2
PIC16F874A	7.2K	4096	192	128	33	8	2	Yes	Yes	Yes	2/1	2
PIC16F876A	14.3K	8192	368	256	22	5	2	Yes	Yes	Yes	2/1	2
PIC16F877A	14.3K	8192	368	256	33	8	2	Yes	Yes	Yes	2/1	2

## 17.0 ELECTRICAL CHARACTERISTICS

### Absolute Maximum Ratings †

Ambient temperature under bias .....	-55 to +125°C
Storage temperature .....	-65°C to +150°C
Voltage on any pin with respect to VSS (except VDD, MCLR, and RA4) .....	-0.3V to (VDD + 0.3V)
Voltage on VDD with respect to VSS .....	-0.3 to +7.5V
Voltage on MCLR with respect to VSS (Note 2) .....	0 to +14V
Voltage on RA4 with respect to VSS .....	0 to +8.5V
Total power dissipation (Note 1) .....	1.0W
Maximum current out of VSS pin .....	300 mA
Maximum current into VDD pin .....	250 mA
Input clamp current, I <sub>IK</sub> (V <sub>I</sub> < 0 or V <sub>I</sub> > VDD) .....	± 20 mA
Output clamp current, I <sub>OK</sub> (V <sub>O</sub> < 0 or V <sub>O</sub> > VDD) .....	± 20 mA
Maximum output current sunk by any I/O pin .....	25 mA
Maximum output current sourced by any I/O pin .....	25 mA
Maximum current sunk by PORTA, PORTB and PORTE (combined) (Note 3) .....	200 mA
Maximum current sourced by PORTA, PORTB and PORTE (combined) (Note 3) .....	200 mA
Maximum current sunk by PORTC and PORTD (combined) (Note 3) .....	200 mA
Maximum current sourced by PORTC and PORTD (combined) (Note 3) .....	200 mA

**Note 1:** Power dissipation is calculated as follows:  $P_{dis} = V_{DD} \times (I_{DD} - \sum I_{OH}) + \sum \{(V_{DD} - V_{OH}) \times I_{OH}\} + \sum (V_{OL} \times I_{OL})$

**2:** Voltage spikes below VSS at the MCLR pin, inducing currents greater than 80 mA, may cause latch-up. Thus, a series resistor of 50-100Ω should be used when applying a "low" level to the MCLR pin rather than pulling this pin directly to VSS.

**3:** PORTD and PORTE are not implemented on PIC16F873A/876A devices.

† NOTICE: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at those or any other conditions above those indicated in the operation listings of this specification is not implied. Exposure to maximum rating conditions for extended periods may affect device reliability.

## Annex 3: NANO RP2040 Microcontroller Datasheet



### Arduino® Nano RP2040 Connect

Product Reference Manual

SKU: ABX00053

#### Description

The feature packed **Arduino® Nano RP2040 Connect** brings the new **Raspberry Pi RP2040** microcontroller to the Nano form factor. Make the most of the dual core **32-bit Arm® Cortex®-M0+** to make Internet of Things projects with Bluetooth and WiFi connectivity thanks to the **U-blox® Nina W102** module. Dive into real-world projects with the onboard accelerometer, gyroscope, RGB LED and microphone. Develop robust embedded AI solutions with minimal effort using the **Arduino® Nano RP2040 Connect**!



#### Target Areas

Internet of Things (IoT), machine learning, prototyping.

#### Features

- **Raspberry Pi RP2040** Microcontroller
  - 133MHz 32bit Dual Core Arm® Cortex®-M0+
  - 264kB on-chip SRAM
  - Direct Memory Access (DMA) controller
  - Support for up to 16MB of off-chip Flash memory via dedicated QSPI bus
  - USB 1.1 controller and PHY, with host and device support
  - 8 PIO state machines
  - Programmable IO (PIO) for extended peripheral support
  - 4 channel ADC with internal temperature sensor, 0.5 MSa/s, 12-bit conversion
  - SWD Debugging
  - 2 on-chip PLLs to generate USB and core clock
  - 40nm process node
  - Multiple low power mode support
  - USB 1.1 Host/Device
  - Internal Voltage Regulator to supply the core voltage
  - Advanced High-performance Bus (AHB)/Advanced Peripheral Bus (APB)

# STM32 F4 DSC 32-bit Cortex-M4

ST is widening its target applications arena with the STM32 F4 series. Based on the Cortex-M4 core, this series opens the door to the digital signal controller (DSC) market. This extension to our STM32 product portfolio offers devices with pin-to-pin and software compatibility with the STM32 F2 series, but with more performance, DSP capability, a floating point unit, more SRAM, and peripheral improvements such as full duplex I<sup>2</sup>S, less than 1  $\mu$ A RTC and 2.44 MSPS ADCs. The ARM Cortex-M4 core features built-in single-cycle multiply-accumulate (MAC) instructions, optimized SIMD arithmetic and saturating arithmetic instructions. The adaptive real-time ART Accelerator™ combined with ST's 90 nm technology provides linear performance up to 168 MHz, unleashing the full performance of the core. These features expand the number of addressable applications in the industrial, consumer and healthcare segments.

The STM32 F4 series includes devices with 512 Kbytes to 1 Mbyte of on-chip Flash memory, and 192 Kbytes of SRAM, and 15 communication interfaces.

WLCSOP (< 4.5 x 4.5 mm), LQFP64, LQFP100, LQFP144, LQFP176 and UFBGA176 packages are available.

## Block diagram



Notes:

- HS requires an external PHY connected to the ULPI interface.
- Crypto/hash processor on STM32F417 and STM32F415.

## Development tools

As for all STM32 products, a complete development tool offering is available, including the following dedicated kits.

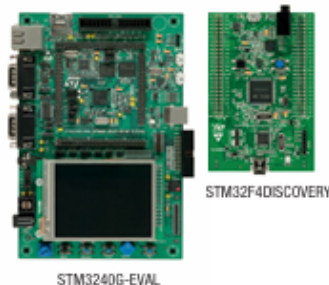
- STM32 F4 Discovery kit (order code: STM32F4DISCOVERY)
- STM32 F4 evaluation board (order codes: STM3240G-EVAL and STM3241G-EVAL<sup>3</sup> for crypto support)
- STM32 F4 starter kits from IAR and Keil (order codes: STM3240G-SK/IAR and STM3240G-SK/KEIL)

Note: 3. Contact your local ST sales office.

## Key figures

- Performance
  - Coremark score: 363.17 at 168 MHz, Coremark/MHz: 2.162
  - Dhrystone score: 210 at 168 MHz
- Power consumption
  - 230  $\mu$ A/MHz at 168 MHz running Coremark benchmark from Flash memory (peripherals off)
  - 1.2 V voltage regulator with power scaling capability
  - 1.7 V<sup>4</sup> to 3.6 V V<sub>CC</sub>
  - <1  $\mu$ A typ RTC
- High-speed data transfer
  - 7 masters, 8 slaves on the multi AHB bus matrix
- Faster peripherals
  - USART: 10.5 Mbit/s
  - SPI: 37.5 Mbit/s
  - ADC: 2.44 MSPS

Note:  
4. 1.7 V available on all packages except the LQFP64





## Annex 5: ATmega2560 Microcontroller Datasheet



### ATmega640/V-1280/V-1281/V-2560/V-2561/V


8-bit Microcontroller with 16/32/64KB In-System Programmable Flash

#### DATASHEET

#### Features

- High Performance, Low Power AVR® 8-Bit Microcontroller
- Advanced RISC Architecture
  - 135 Powerful Instructions – Most Single Clock Cycle Execution
  - 32 × 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 16 MIPS Throughput at 16MHz
  - On-Chip 2-cycle Multiplier
- High Endurance Non-volatile Memory Segments
  - 64K/128K/256KBytes of In-System Self-Programmable Flash
  - 4Kbytes EEPROM
  - 8Kbytes Internal SRAM
  - Write/Erase Cycles: 10,000 Flash/100,000 EEPROM
  - Data retention: 20 years at 85°C/ 100 years at 25°C
  - Optional Boot Code Section with Independent Lock Bits
    - In-System Programming by On-chip Boot Program
    - True Read-While-Write Operation
  - Programming Lock for Software Security
    - Endurance: Up to 64Kbytes Optional External Memory Space
- QTouch® library support
  - Capacitive touch buttons, sliders and wheels
  - QTouch and QMatrix acquisition
  - Up to 64 sense channels
- JTAG (IEEE® std. 1149.1 compliant) Interface
  - Boundary-scan Capabilities According to the JTAG Standard
  - Extensive On-chip Debug Support
  - Programming of Flash, EEPROM, Fuses, and Lock Bits through the JTAG Interface
- Peripheral Features
  - Two 8-bit Timer/Counters with Separate Prescaler and Compare Mode
  - Four 16-bit Timer/Counter with Separate Prescaler, Compare- and Capture Mode
  - Real Time Counter with Separate Oscillator
  - Four 8-bit PWM Channels
  - Six/Twelve PWM Channels with Programmable Resolution from 2 to 16 Bits (ATmega1281/2561, ATmega640/1280/2560)
  - Output Compare Modulator
  - 8/16-channel, 10-bit ADC (ATmega1281/2561, ATmega640/1280/2560)
  - Two/Four Programmable Serial USART (ATmega1281/2561, ATmega640/1280/2560)
  - Master/Slave SPI Serial Interface
  - Byte Oriented 2-wire Serial Interface
  - Programmable Watchdog Timer with Separate On-chip Oscillator
  - On-chip Analog Comparator
  - Interrupt and Wake-up on Pin Change
- Special Microcontroller Features
  - Power-on Reset and Programmable Brown-out Detection
  - Internal Calibrated Oscillator
  - External and Internal Interrupt Sources
  - Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby, and Extended Standby
- I/O and Packages
  - 54/66 Programmable I/O Lines (ATmega1281/2561, ATmega640/1280/2560)
  - 64-pad QFN/MLF, 64-lead TQFP (ATmega1281/2561)
  - 108-lead TQFP, 108-ball CBGA (ATmega640/1280/2560)
  - RoHS/Full Green
- Temperature Range:
  - -40°C to 85°C Industrial
- Ultra-Low Power Consumption
  - Active Mode: 1MHz, 1.8V: 500µA
  - Power-down Mode: 0.1µA at 1.8V
- Speed Grade:
  - ATmega640V/ATmega1280V/ATmega1281V:
    - 0 - 4MHz @ 1.8V - 5.5V, 0 - 8MHz @ 2.7V - 5.5V
  - ATmega2560V/ATmega2561V:
    - 0 - 2MHz @ 1.8V - 5.5V, 0 - 8MHz @ 2.7V - 5.5V
  - ATmega640/ATmega1280/ATmega1281:
    - 0 - 8MHz @ 2.7V - 5.5V, 0 - 16MHz @ 4.5V - 5.5V
  - ATmega2560/ATmega2561:
    - 0 - 16MHz @ 4.5V - 5.5V

## Annex 6:ESP32C3 Microcontroller Datasheet




# ESP32-C3

Product

### A Cost-Effective MCU with a RISC-V Single-Core CPU


### Wi-Fi and Bluetooth 5 (LE) Connectivity for Secure IoT Applications

## Features




#### CPU & Memory

- 32-bit RISC-V single-core processor with a four-stage pipeline that operates at up to 160 MHz
- 384 KB ROM, 400 KB SRAM, 8 KB SRAM in RTC and external Quad SPI/QPI 16 MB flash




#### Connectivity

- 2.4 GHz Wi-Fi 802.11 b/g/n with HT20 / HT40
- Bluetooth 5 (LE) with Long Range support
- Wi-Fi and Bluetooth LE mesh support



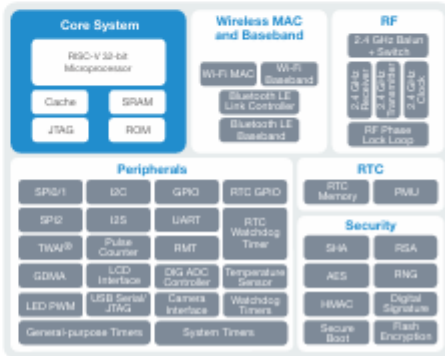
#### Peripherals

- 22 programmable GPIOs: UART, SPI, I<sup>2</sup>C, I<sup>2</sup>S, PWM, ADC, TWAI, Full-speed USB Serial/JTAG controller



#### Security

- RSA-3072-based secure boot
- AES-128/256-XTS-based flash encryption
- Digital signature peripheral and the HMAC peripheral
- Hardware acceleration support for cryptographic algorithms



ESP32-C3 Block Diagram

## Highlights

### RISC-V at the Core

- ESP32-C3 integrates a 32-bit core RISC-V microcontroller with a maximum clock speed of 160 MHz.
- With 22 configurable GPIOs, 400 KB of internal RAM and low-power-mode support, it can facilitate many different use cases involving connected devices.
- The MCU comes in multiple variants with integrated and external flash availability.

### 2.4 GHz Wi-Fi + Bluetooth 5 (LE)

- IEEE 802.11 b/g/n-compliant; Supports 20 MHz, 40 MHz bandwidth in 2.4 GHz band; 1T1R mode with a data rate of up to 150 Mbps
- Bluetooth 5 (LE); Bluetooth mesh; Advertising extensions

Learn More: [espressif.com](https://www.espressif.com)  
Product Selector: [espressif.com/product-selector](https://www.espressif.com/product-selector)  
Contact Us: [espressif.com/sales](https://www.espressif.com/sales)

## Applications

- Smart home (Light-control system)
- Industrial automation
- Health care
- Consumer electronics
- Generic low-power IoT sensor hubs
- Generic low-power IoT data loggers

Share :: Connect :: Innovate

12

80

## MSP430x11x2, MSP430x12x2 MIXED SIGNAL MICROCONTROLLER

SLAS361D – JANUARY 2002 – REVISED AUGUST 2004

- Low Supply Voltage Range 1.8 V to 3.6 V
- Ultralow-Power Consumption:
  - Active Mode: 200  $\mu$ A at 1 MHz, 2.2 V
  - Standby Mode: 0.7  $\mu$ A
  - Off Mode (RAM Retention): 0.1  $\mu$ A
- Five Power Saving Modes
- Wake-Up From Standby Mode in less than 6  $\mu$ s
- 16-Bit RISC Architecture, 125 ns Instruction Cycle Time
- Basic Clock Module Configurations:
  - Various Internal Resistors
  - Single External Resistor
  - 32-kHz Crystal
  - High Frequency Crystal
  - Resonator
  - External Clock Source
- 16-Bit Timer\_A With Three Capture/Compare Registers
- 10-Bit, 200-kSPS A/D Converter With Internal Reference, Sample-and-Hold, Autoscan, and Data Transfer Controller
- Serial Communication Interface (USART0) With Software-Selectable Asynchronous UART or Synchronous SPI (MSP430x12x2 Only)
- Serial Onboard Programming, No External Programming Voltage Needed
- Programmable Code Protection by Security Fuse
- Supply Voltage Brownout Protection
- MSP430x11x2 Family Members Include:
  - MSP430F1122: 4KB + 256B Flash Memory 256B RAM
  - MSP430F1132: 8KB + 256B Flash Memory 256B RAM
 Available in 20-Pin Plastic SOWB, 20-Pin Plastic TSSOP and 32-Pin QFN Packages
- MSP430x12x2 Family Members Include:
  - MSP430F1222: 4KB + 256B Flash Memory 256B RAM
  - MSP430F1232: 8KB + 256B Flash Memory 256B RAM
 Available in 28-Pin Plastic SOWB, 28-Pin Plastic TSSOP, and 32-Pin QFN Packages
- For Complete Module Descriptions, See the *MSP430x1xx Family User's Guide*, Literature Number SLAU049

### description

The Texas Instruments MSP430 family of ultralow-power microcontrollers consist of several devices featuring different sets of peripherals targeted for various applications. The architecture, combined with five low power modes is optimized to achieve extended battery life in portable measurement applications. The device features a powerful 16-bit RISC CPU, 16-bit registers, and constant generators that attribute to maximum code efficiency. The digitally controlled oscillator (DCO) allows wake-up from low-power modes to active mode in less than 6 $\mu$ s.

The MSP430x11x2 and MSP430x12x2 series are ultralow-power mixed signal microcontrollers with a built-in 16-bit timer, 10-bit A/D converter with integrated reference and data transfer controller (DTC) and fourteen or twenty-two I/O pins. In addition, the MSP430x12x2 series microcontrollers have built-in communication capability using asynchronous (UART) and synchronous (SPI) protocols.

Digital signal processing with the 16-bit RISC performance enables effective system solutions such as glass breakage detection with signal analysis (including wave digital filter algorithm). Another area of application is in stand-alone RF sensors.



Please be aware that an important notice concerning availability, standard warranty, and use in critical applications of Texas Instruments semiconductor products and disclaimers thereto appears at the end of this data sheet.

PRODUCTION DATA information is current as of publication date. Products conform to specifications per the terms of Texas Instruments standard warranty. Production processing does not necessarily include testing of all parameters.



POST OFFICE BOX 655303 • DALLAS, TEXAS 75265

Copyright © 2002 – 2004, Texas Instruments Incorporated

1

## Annex 8: ATSAMD20J Microcontroller Datasheet

### Features

---

- Processor
  - ARM Cortex-M0+ CPU running at up to 48MHz
    - Single-cycle hardware multiplier
- Memories
  - 16/32/64/128/256KB in-system self-programmable flash
  - 2/4/8/16/32KB SRAM
- System
  - Power-on reset (POR) and brown-out detection (BOD)
  - Internal and external clock options with 48MHz Digital Frequency Locked Loop (DFLL48M)
  - External Interrupt Controller (EIC)
  - 16 external interrupts
  - One non-maskable interrupt
  - Two-pin Serial Wire Debug (SWD) programming, test and debugging interface
- Low Power
  - Idle and standby sleep modes
  - SleepWalking peripherals
- Peripherals
  - 8-channel Event System
  - Up to eight 16-bit Timer/Counters (TC), configurable as either:
    - One 16-bit TC with compare/capture channels
    - One 8-bit TC with compare/capture channels
    - One 32-bit TC with compare/capture channels, by using two TCs
  - 32-bit Real Time Counter (RTC) with clock/calendar function
  - Watchdog Timer (WDT)
  - CRC-32 generator
  - Up to six Serial Communication Interfaces (SERCOM), each configurable to operate as either:
    - USART with full-duplex and single-wire half-duplex configuration
    - I<sup>2</sup>C up to 400kHz
    - SPI
  - One 12-bit, 350ksps Analog-to-Digital Converter (ADC) with up to 20 channels
    - Differential and single-ended channels
    - 1/2x to 16x gain stage
    - Automatic offset and gain error compensation
    - Oversampling and decimation in hardware to support 13-, 14-, 15- or 16-bit resolution
  - 10-bit, 350ksps Digital-to-Analog Converter (DAC)
  - Two Analog Comparators with window compare function
  - Peripheral Touch Controller (PTC)
    - 256-Channel capacitive touch and proximity sensing
- I/O
  - Up to 52 programmable I/O pins
- Packages
  - 64-pin TQFP, QFN
  - 48-pin TQFP, QFN
  - 32-pin TQFP, QFN
- Operating Voltage
  - 1.62V – 3.63V
- Power Consumption
  - Down to 70µA/MHz in active mode
  - Down to 8µA running the Peripheral Touch Controller

## Annex 9:Arduino IDE Code

```
// Librarys
#include "SoftwareSerial.h" // for bluetooth module HC-06
#include "DHT.h"           // temperature sensors
#include <Wire.h>           //Display
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 16, 2);

//diretives for the pins
#define MAGNETIC_SENSOR_PIN 2
#define PRESENCE_PIN 3
#define BUZZER_PIN 5
#define DHT_PIN 6
#define RELAY_LAMP1_PIN 7
#define RELAY_LAMP2_PIN 8
#define BLUETOOTH_RX_PIN 10
#define BLUETOOTH_TX_PIN 11
#define LED_ENTRANCE A0
#define LED_MAINROOM A1
#define LED_KITCHEN A2
#define LED_LIVING_ROOM A3
#define LED_BATHROOM A4
#define LIGHT_SENSOR_PIN A5

#define DHTTYPE DHT11 //FOR TEMPERATURE SENSORS

SoftwareSerial BTSerial(BLUETOOTH_RX_PIN, BLUETOOTH_TX_PIN);
DHT dht(DHT_PIN, DHTTYPE);
bool securityMode = false;
int doorClosed = 1;
int motion = 0;
bool ldr = false;
int ldrStatus = 0;
bool intruder = false;
float temp = 0;
char command = 0; //stores the received command

void setup() {
  Serial.begin(9600);
  pinMode(PRESENCE_PIN, INPUT);
  pinMode(MAGNETIC_SENSOR_PIN, INPUT);
  pinMode(LIGHT_SENSOR_PIN, INPUT);
  pinMode(DHT_PIN, INPUT);

  pinMode(BLUETOOTH_RX_PIN, INPUT);
  pinMode(BLUETOOTH_TX_PIN, OUTPUT);
}
```

```

pinMode(RELAY_LAMP1_PIN, OUTPUT);

pinMode(RELAY_LAMP2_PIN, OUTPUT);
pinMode(LED_ENTRANCE, OUTPUT);
pinMode(LED_MAINROOM, OUTPUT);
pinMode(LED_KITCHEN, OUTPUT);
pinMode(LED_LIVING_ROOM, OUTPUT);
pinMode(LED_BATHROOM, OUTPUT);
pinMode(BUZZER_PIN, OUTPUT);
digitalWrite(RELAY_LAMP1_PIN, HIGH);
digitalWrite(RELAY_LAMP2_PIN, HIGH);
digitalWrite(LED_BATHROOM, LOW);
digitalWrite(BUZZER_PIN, LOW);

BTSerial.begin(9600);

Serial.println("START");

dht.begin();
}
void loop() {
    //temperature reading
    temp = dht.readTemperature();
    doorClosed = digitalRead(MAGNETIC_SENSOR_PIN);
    motion = digitalRead(PRESENCE_PIN);
    ldrStatus = analogRead(LIGHT_SENSOR_PIN);

    Serial.print("motion = ");
    Serial.println(motion);

    Serial.print("temp = ");
    Serial.println(temp);

    Serial.print("doorClosed = ");
    Serial.println(doorClosed);

    Serial.print("ldrStatus = ");
    Serial.println(ldrStatus);

    Serial.println("-----");

    // temperature reading
    temp = dht.readTemperature();
    lcd.setCursor(0, 0);
    lcd.print("My Smart Home");
    lcd.setCursor(0, 1);
    lcd.print("Temperature: ");
    lcd.print(temp);
    lcd.print("°C");
    delay(2000);
}

```

```

// Security mode
if (securityMode && doorClosed == LOW ) {
    Serial.println("Alert: Door Opened!");
    intruder = true;
    tone(BUZZER_PIN, 1000);
}
else if (securityMode && motion) {
    Serial.println("Alert: Somebody in the house!");
    intruder = true;
    tone(BUZZER_PIN, 1000);
} else if (intruder) {
    tone(BUZZER_PIN, 1000);
} else {

    noTone(BUZZER_PIN);
}
//entrance light
if (ldrStatus <= 750 || ldr) {
    digitalWrite(LED_ENTRANCE, HIGH);

} else if (ldrStatus > 750 && !ldr) {
    digitalWrite(LED_ENTRANCE, LOW);
}

if (BTSerial.available() > 0) {

    command = BTSerial.read();

    Serial.println(command);
    Serial.println("-----");
    switch (command) {
        //ENTRANCE, MAINROOM, KITCHEN, LIVING_ROOM, BATHROOM

        case 'M' : //main room light
            digitalWrite(LED_MAINROOM, HIGH);
            break;
        case 'm' :
            digitalWrite(LED_MAINROOM, LOW);
            break;
        case 'K' : //kitchen light
            digitalWrite(LED_KITCHEN, HIGH);
            break;
        case 'k' :
            digitalWrite(LED_KITCHEN, LOW);
            break;
        case 'L' : //living room
            digitalWrite(LED_LIVING_ROOM, HIGH);

```

```

        break;
    case 'l' :
        digitalWrite(LED_LIVING_ROOM, LOW);
        break;
case 'B' : //bathroom
    digitalWrite(LED_BATHROOM, HIGH);
    break;
    case 'b' :
        digitalWrite(LED_BATHROOM, LOW);
        break;
    case 'R' : //Relay_1
        digitalWrite(RELAY_LAMP1_PIN, HIGH);
        break;
    case 'r' :
        digitalWrite(RELAY_LAMP1_PIN, LOW);
        break;
    case 'Q' : //main room light
        digitalWrite(RELAY_LAMP2_PIN, HIGH);
        break;
    case 'q' :
        digitalWrite(RELAY_LAMP2_PIN, LOW);
        break;

    case 'S': //Security mode
        Serial.println("Security Mode Activated");
        securityMode = true;
        break;

    case 's' :
        Serial.println("Security Mode Deactivated");
        securityMode = false;
        intruder = false;
        noTone(BUZZER_PIN);
        break;

    case 'C' :          //ldr sensor
        ldr = true;
        break;

    case 'c' :
        ldr = false;
        break;
}
}

delay(500);
}

```



## DECLARATION

### on authenticity and public assess of final thesis

Student's name: Ruth Borges  
Student's Neptun ID: YNNI9B  
Title of the document: Data collection and Management with Microcontrollers  
Year of publication: 2024  
Department: Institute of Technology

I declare that the submitted final thesis is my own, original individual creation. Any parts taken from an another author's work are clearly marked, and listed in the table of contents.

If the statements above are not true, I acknowledge that the Final examination board excludes me from participation in the final exam, and I am only allowed to take final exam if I submit another final essay/thesis/master's thesis/portfolio.

Viewing and printing my submitted work in a PDF format is permitted. However, the modification of my submitted work shall not be permitted.

I acknowledge that the rules on Intellectual Property Management of Hungarian University of Agriculture and Life Sciences shall apply to my work as an intellectual property.

I acknowledge that the electric version of my work is uploaded to the repository sytem of the Hungarian University of Agriculture and Life Sciences.

Place and date: Gödöllő, 2024 year November month 11 day

Ruth Borges  
Student's signature

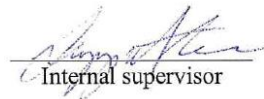
## STATEMENT ON CONSULTATION PRACTICES

As a supervisor of Ruth Borges (Student's name) YNNI9B (Student's NEPTUN ID), I here declare that the final thesis has been reviewed by me, the student was informed about the requirements of literary sources management and its legal and ethical rules.

I recommend/don't recommend<sup>1</sup> the final essay/thesis/master's thesis/portfolio to be defended in a final exam.

The document contains state secrets or professional secrets: yes no<sup>\*2</sup>

Place and date: 2024 year 09 month 08 day

  
Internal supervisor

---

<sup>1</sup> Please underline applicable.

<sup>2</sup> Please underline applicable.