Automatic Lights

Final Report

Carolina Ferreira	up201905810
João Amaral	up202200195
Tiago André	up201606796
Tiago Eusébio	up201904872

Contents

1 Introduction **3**

2 Planning **3**

3 Development **6**

3.1 Arduino 6

3.2 Raspberry Pi 7

3.3 Android 7

4 Objectives **8**

5 Git & Wiki **9**

6 Conclusion **9**

1 Introduction

The objective of this project is to develop a system that can automate the process of turning lights on and off in a room based on user entry and exit. The system must be able to detect a set of specific users when they enter and leave the room. The primary goal is to improve energy efficiency and convenience for users by eliminating the need to manually turn the lights on and off.

To achieve this objective, the project will utilize various technologies, including sensors, to detect the entry and exit of users. Additionally, an Android application will be developed that allows users to control the system manually, monitor the status of the lights in the room, and send a welcome message to users upon their entry into the room.

By accomplishing the previous objectives, the system will provide a convenient and efficient solution for managing lighting in a room while reducing energy consumption.

2 Planning

The development of the project was planned considering the functional requirements and the established non-functional requirements. The system architecture was designed in order to identify the components needed to achieve the desired functionality.

Initially, the functional requirements were identified, which include checking the light level in the room before turning on the lighting. The light should not be turned on if the light level is above a certain threshold and also should not be turned off if there is a known user present in the room. In addition, it has been established that user detection should be performed in less than 5 seconds, while remote control to turn the lights on/off should occur in less than 4 seconds after the button is pressed. Similarly, information on light status provided by arduino should be obtained in less than 4 seconds.

Next, non-functional requirements were considered to ensure the reliability, safety, ease of use, and energy efficiency of the system. Also taken into account were aspects such as compatibility with different types of lamps, ease of maintenance, and scalability to allow future expansions of the project.

Based on these requirements, the system architecture was designed, involving essential components. The user detection system is responsible for detecting the presence of users in the room, using sensors to identify the entry and exit of users. The light control system is responsible for turning the lights on and off, receiving signals from the user detection system and responding accordingly. It also prevents the lights from being turned off if a known user is present in the room by checking the

ambient brightness before turning on the lights. The Android application provides users with a remote control to turn the lights on/off, displaying the current status of the lights in the room and communicating with the light control system to send commands and receive status updates. Finally, the user information system stores the information of the users authorized to access the room, allowing the Android application to display a personalized welcome message when the user enters the room.

The development of the project followed a structured approach, with the components being implemented and integrated as planned. Throughout the process, performance criteria and requirements were considered to ensure that the system met the expectations set.

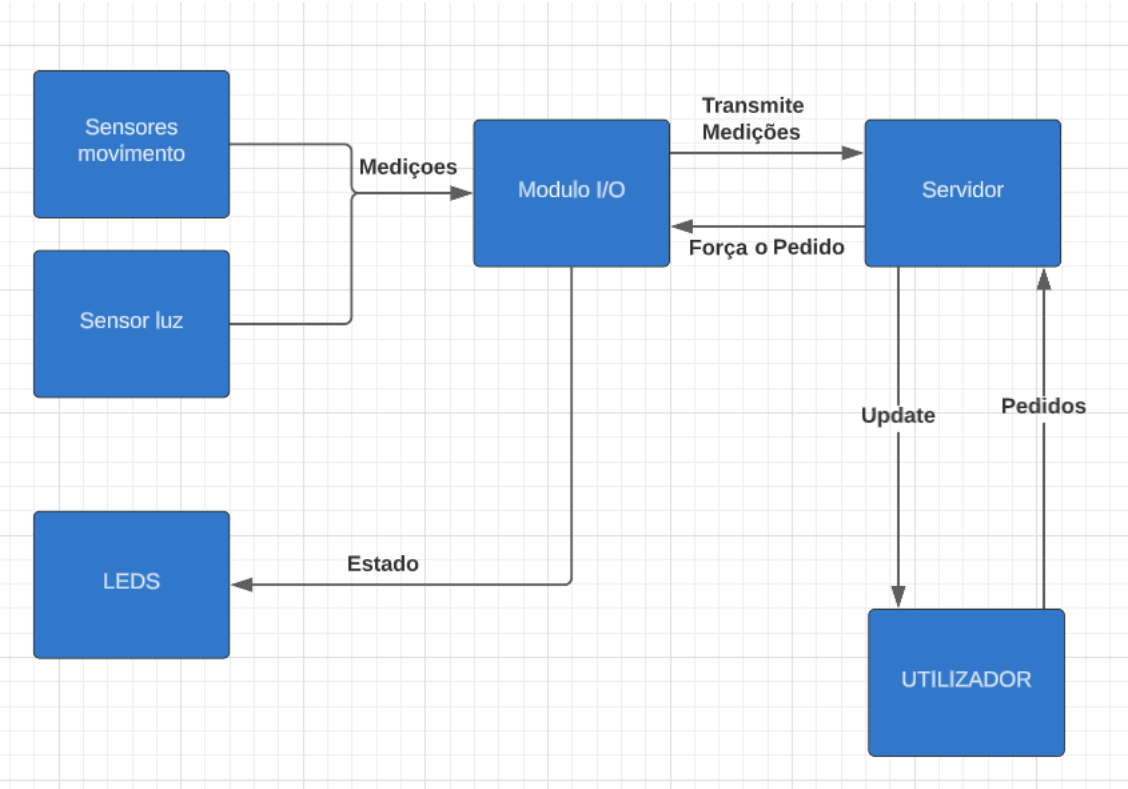

Figure 1: Specification Diagram

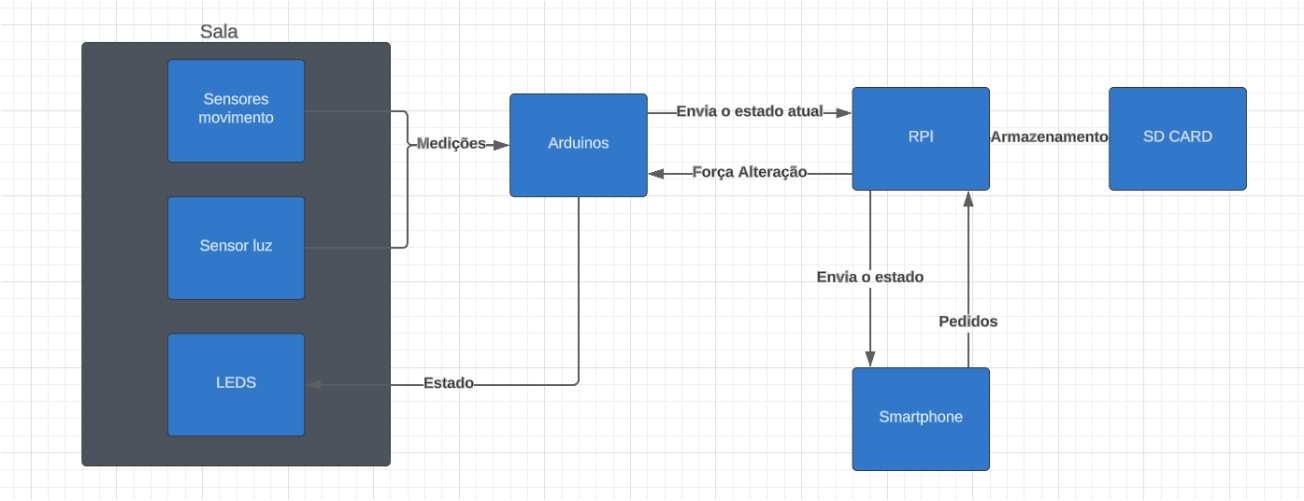

Figure 2: Hardware Diagram

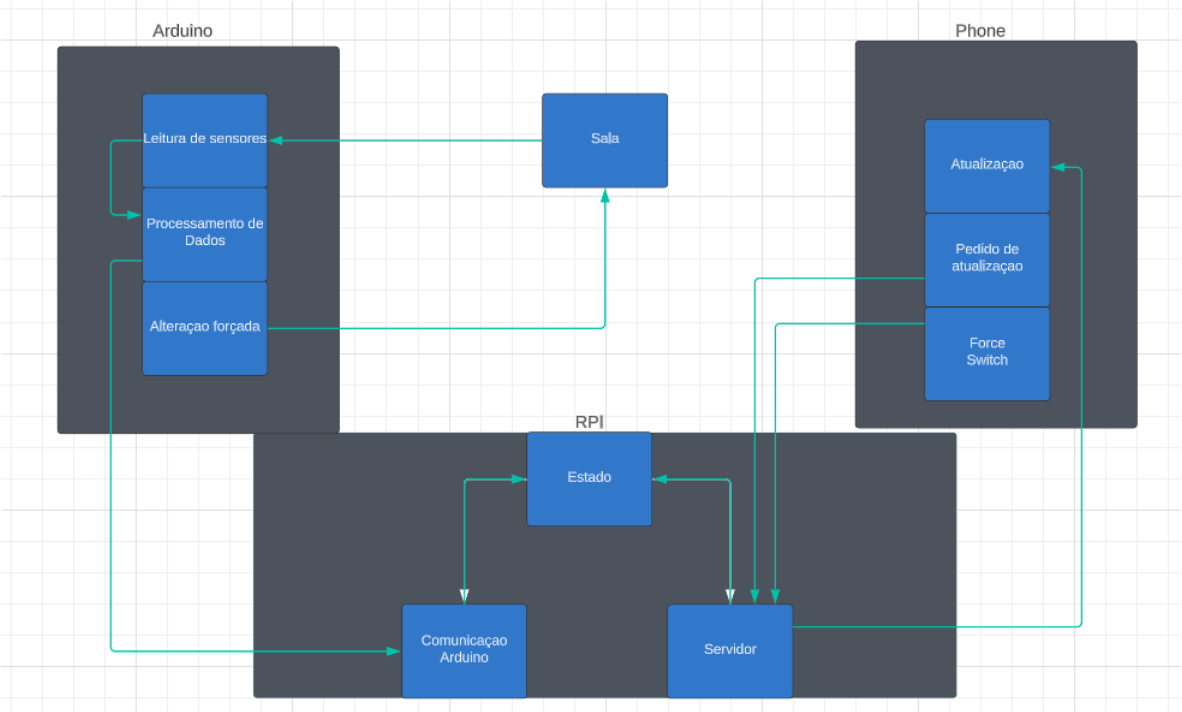

Figure 3: Software Diagram

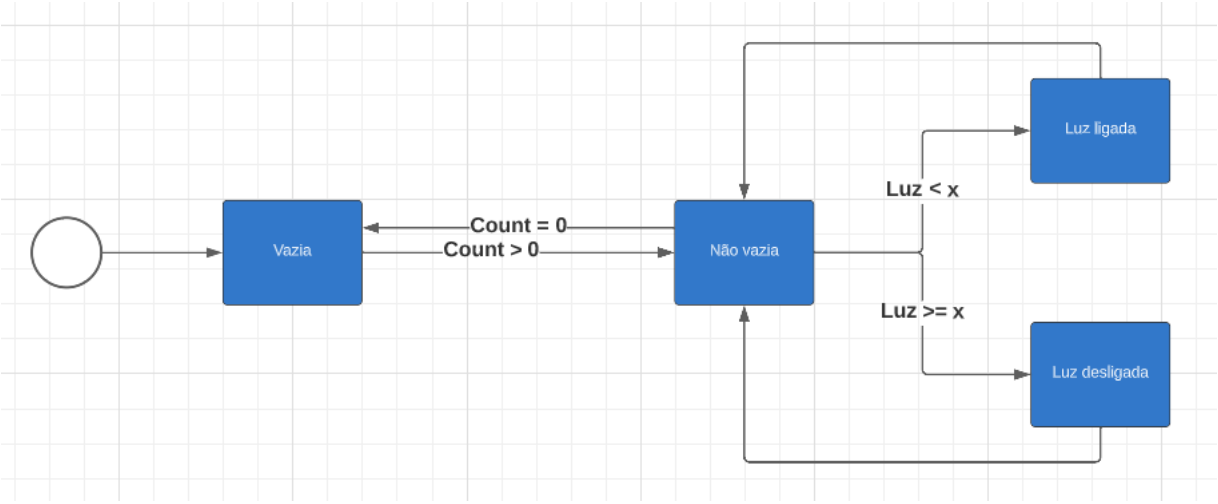

Figure 4: State Diagram

3 Development

3.1 Arduino

In our project, we worked on the Arduino with two presence sensors that detect the entrance and exit of people inside the room thus defining the presence or absence of people. We also used a photo-resistor sensor sensor to detect the brightness in the room.

Initially, the Arduino ignores the values read by the light sensor (photo-resistor) since initially there are no people in the room, there is no need to check if the existing natural light is sufficient or not to be necessary to activate the artificial light, this same ideology is maintained if the control variable (count) is zero (0), since this means that the room is empty.

If the motion sensors are activated, depending on the order in which they are activated the count variable increases or decreases, if this variable has a value greater than zero, the photoresistors' values will start to be counted, if these values are less than a certain value defined by us the light will turn on, turning off if the opposite happens.

Every time the light state changes, the Arduino writes a message with the new state on the port to keep the server updated about the current state.

Besides that the arduino also has a piece of code about forced on/off that the application has access to, basically the application at any time can ask to turn the light on or off, in this part of the code if the arduino receives a message, it will ignore the rest of the code changing the state of the light to the request for five seconds (5s), after these seconds it will execute its code normally again from the main structure.

3.2 Raspberry Pi

In this project, the Raspberry Pi serves as a central server, acting as an intermediary between the Arduino and the Android app. The server is developed using Python as the primary programming language, and Flask is utilized for its deployment. Flask is a lightweight web framework for Python that provides a simple and flexible way to create web applications. It allows for easy routing, request handling, and response generation, making it an ideal choice for this project's server implementation.

The Raspberry Pi facilitates communication between the Arduino and the server through the serial port. This communication is crucial for exchanging data and instructions between the Arduino, which controls the physical light, and the server. By establishing a serial connection, the server can continuously receive updates from the Arduino, ensuring that it always has the latest information on the light's state.

On the other hand, the communication between the Android app and the server occurs via Wi-Fi. The Android app sends requests to the server, such as retrieving the current status of the light or issuing commands to turn it on or off. These requests are made using HTTP methods, primarily **GET** and **POST** requests.

After the user logs into the Android app, they need to click the *Status* button to check the current state of the light. Once the status is displayed, the second button is enabled, allowing the user to turn the light on or off. This process is achieved by sending **GET** requests for the *Status* and **POST** requests for *Light ON* and *Light OFF* to the server.

Inside the server, the received **JSON** messages are processed, and appropriate actions are taken based on the user's requests. The server then generates the necessary response, which is sent back to the Android app.

By leveraging the capabilities of the Raspberry Pi as a server, the project achieves a seamless flow of information between the Arduino and the Android app. This setup enables users to monitor the current state of the light and easily control it from their mobile devices, providing convenience and efficient light management.

3.3 Android

The Android application aims to offer users remote control of the lights in a room. With it, you can turn the lights on/off via the application, receive information about the current status of the lights, and communicate with the lights control system to send commands and receive status updates.

The application consists of three main pages: the login page, where users can enter the application; the registration page, for those who want to register; and the main page, which displays the current status of the lights and allows you to turn them on or off.

The main functionality of the application is implemented in the class *MainActivity*, in the *onCreate()* method, which is executed when the activity is created, some important steps occur. First, the layout of the activity is set using the *ActivityLightStatusBinding*. Next, the visual content of the activity is set by calling *setContentView()*.

Click event configuration is done for the *bt_light* and *bt_status* buttons. When the *bt_light* button is clicked, a **POST** request is made to the server to turn the light on or off depending on the current state set by the *isOn* variable. The *bt_status* button makes a **GET** request to get the current state of the light.

The *isOff()* and *isOn()* methods are responsible for updating the user interface according to the light state obtained from the server's responses. These methods modify the text, image, and visual appearance of the *t_light* button and the *iv_img* image, representing the current state of the light (on or off).

The *get_status()* method is called to get the current state of the light from the server via a **GET** request. As with the *isOff()* and *isOn()* methods, it updates the user interface according to the obtained state.

In both **POST** and **GET** requests, the server's responses are handled in the *onResponse()* method. Depending on the content of the response, the *isOff()* or *isOn()* methods are called to update the user interface. If the response is not as expected, the interface is set to the *off* state and the *bt_light* button is disabled.

4 Objectives

On of the main objective of this project was to develop a system that allows users to remotely control a light using an Android app while receiving real-time updates on the light's status. Although the project has made significant progress towards achieving its goals, there were some limitations and unmet objectives.

On the Arduino side, the initial plan was to establish communication with the Raspberry Pi using Bluetooth. This would have provided a wireless connection between the Arduino and the server, allowing for greater flexibility in the placement of the Arduino and eliminating the need for a physical serial connection. However, due to technical challenges or constraints, the implementation of Bluetooth communication could not be realized within the project timeline. As a result, the Arduino and the Raspberry Pi had to rely on a serial connection for data exchange.

Similarly, on the Raspberry Pi side, the project aimed to develop a multi-client connection capability. This would have allowed multiple Android devices to connect to the server simultaneously, enabling multiple users to control the light simultaneously. However, due to certain limitations or complexities in implementing a robust multi-client connection, this objective could not be fully accomplished within the project scope.

Despite these limitations, the project has successfully implemented several key functionalities.

5 Git & Wiki

- Automatic Lights - Git repository
- Automatic Lights - Wiki

6 Conclusion

To conclude at the end of this project we ended up with a system where the three components are able to communicate between themselves passing messages through the server/RaspberryPi between the android application and the arduino keeping always available the state of the light as well as its change depending on natural conditions or application requests, unfortunately we weren't able to fulfill one of the requirements that would be to identify special people (known people) by requiring operations through bluetooth communication/configurations in the only option we thought we were capable of, configurations that we weren't able to understand enough to execute within the time allotted for this project.

So that's the only major flaw in the project