KU LEUVEN

FACULTY OF ENGINEERING TECHNOLOGY

CAMPUS GROUP T LEUVEN

Secure Access Management Solution for Lab Environments

Engineering Experience 3

Nur ALDA-ONGGAR

Gijs FITEN

Javad NEMATLI

Dias SERZHAN

Vincent VANSANT

Coach(es): Gorik STEVENS

Technical paper submitted to obtain the degree of Bachelor of Science in Engineering Technology: Electronics Engineering

Academic Year 2022 - 2023

KU LEUVEN

Faculty of Engineering Technology Campus GROUP T Leuven — Technical paper submitted to obtain the degree of Bachelor of Science in Engineering Technology: Electronics Engineering
Academic Year 2022 - 2023

Secure Access Management Solution for Lab Environments

Engineering Experience 3

Alda-Onggar Nur, Fiten Gijs, Nematli Javad, Serzhan Dias, Vansant Vincent

Bachelor in Electronics Engineering, Faculty of Engineering Technology, Campus GROUP T Leuven, Andreas Vesaliusstraat 13, 3000 Leuven, Belgium

Coach(es): Gorik STEVENS

Electronics Engineering, Faculty of Engineering Technology, Campus GROUP T Leuven, Andreas Vesaliusstraat 13, 3000 Leuven, Belgium, gorik.stevens@KULeuven.be

ABSTRACT

A tool access system for KU Leuven's Group T campus enables the staff to focus on more pressing matters at the campus and enables the students to use the tools at their disposal without the constant supervision of a teacher. Our system implements a robust and efficient method of giving the student access to whatever tools they need if they have the correct certifications and training for that tool. The system also logs any activity in the tool room to allow the teaching staff to quickly check any recent activity when necessary. A web interface was also created to give the teaching staff an easy-to-use interface to review logs and change or create access levels. This system is implemented with robustness and efficiency as primary focuses, the system is not dependent on a Wi-Fi connection to function and can run for a whole semester without needing intervention by the staff.

1 INTRODUCTION

Tool rooms serve as essential spaces on campuses, providing students with the necessary tools and machines to facilitate project work and experimentation. These dedicated areas play a crucial role in fostering teamwork and collaboration among budding engineering students. Ensuring the safety of these rooms is of utmost importance, as it guarantees a secure working environment for students with varying levels of expertise.

Nonetheless, ongoing faculty supervision is imperative due to potential hazards and the risk of equipment misuse. As a result, access to tool rooms is limited to students who have staff members present to supervise and ensure the safety of both the equipment and the students themselves. The constant presence of a faculty member is impractical and cost-ineffective, therefore there is a crucial need for a centralized system that can manage the students' access to the tools and machinery.

Initially, the project is designed for the tool room situated in KU Leuven Campus Group T's Module 4, with the potential for future scalability and adaptability to other engineering campuses and workshops. Campus Group T has students specializing in diverse engineering disciplines and exemplifying a wide range of skill sets and levels of expertise. Therefore, the importance of an access level management system becomes more evident.

This paper is structured as follows: The requirement analysis highlights the system's essential features and specifications and is presented in section 1.1. The system's design and the materials used in its construction are discussed in chapter 2. The implementation is the primary focus of chapter 3, which is divided into hardware implementation (section 3.1) and software implementation (section 3.2). The system's evaluation is presented in chapter 4, including the test and methods used to assess its efficiency and performance. In chapter 5, the evaluation data is discussed, highlighting the system's strengths and weaknesses and whether or not the system meets the requirements. The conclusion comes in chapter 6.

1.1 Requirements analysis

In this paper, the requirements for a hierarchical system composed of a Master Node, multiple Slave Nodes, and a web application are analyzed for equipment management and control. The Master Node coordinates communication with Slave Nodes, which handle equipment-specific tasks, while the web application offers administrators a user-friendly interface for system control and monitoring.

The following subsections discuss the essential functionalities and additional requirements for the Master Node, Slave Nodes, and web application. The objective is to develop a robust, scalable, and secure system that addresses user needs and accommodates various equipment and scenarios.

1.1.1 Master Node

The base functionality of the Master Node involves communicating with multiple Slave Nodes using the nRF24L01. Additionally, it is responsible for determining whether a key has access to a device by consulting a local database and subsequently granting or denying access as appropriate.

For non-essential functionality, the Master Node should support Wi-Fi connectivity, enabling remote control, user authentication, and session management. This allows authorized users to manage devices, access levels, and the local database. Additionally, the Master Node should offer logging capabilities with accurate timestamps and ensure regular log uploads to a remote database, handling local log deletions after successful transfers.

Beyond these basic functionalities, the Master Node should encompass several design philosophies, including robustness, scalability, security, and task management. The Master Node software should be designed to gracefully handle errors and edge cases, ensuring reliable operation. It should also be capable of accommodating an increasing number of Slave Nodes and user interactions without compromising performance. Security measures should be implemented to prevent unauthorized access and safeguard sensitive data. Lastly, the Master Node should utilize a multitasking architecture, such as FreeRTOS, to efficiently manage various tasks and processes.

1.1.2 Slave Nodes

The Slave Nodes utilize a microcontroller from the STM8S family. It is selected for its low-power and robust design and SPI interface. Wireless communication between Slave Nodes and the Master Node is essential because of their location. A nRF24L01 is chosen to enable wireless communication.

Accurate and reliable sensors, such as a hall sensor for displacement, is required. Relays must be selected based on the voltage and current ratings of the powered devices.

A stable and reliable power supply, such as a 3-pin DC barrel jack for battery connection, is crucial for proper node operation. Two voltage regulators are needed to convert 18V to 5V and 5V to 3.3V without much thermal losses.

Components should be chosen to ensure suitably sized boards, with resistors and capacitors primarily using 0805 SMD packages. Moreover, the through hole components are large and should only be used when the SMD packages fall short. Finally, boards should be housed in protective enclosures to shield them from environmental hazards and accidental contact.

1.1.3 Web Application

The web application must meet several key requirements, including secure user authentication, allowing administrators to log in with their unique credentials while preventing security breaches and unauthorised use of the administration capabilities. Additionally, administrators should have the ability to manage access levels for students, including assigning or revoking specific equipment permissions. They must also be able to view and manage the equipment inventory, add or delete equipment details, and set access levels for each tool.

Furthermore, the web application must enable remote control of electrically powered tools, allowing administrators to turn them on and off in the equipment room, ensuring safety and energy efficiency. Finally, the application should grant administrators access to detailed logs of equipment usage, including information about the equipment used and the duration of usage, as well as provide an overview of usage statistics to help identify trends.

2 DESIGN AND MATERIALS

The equipment room access control system is designed to monitor and control access to the room and its electric tools. It integrates four principal components: the Master node, Relay slave, Hall sensor slave, and a Web application. Each of these components contributes to a full system that ensures access to the tools is granted only to those with appropriate credentials, tracks tool usage, and enables administrators to review activity logs and adjust user access levels.

The Master node (Figure 2.1) serves as the central building block, managing communications with the slave nodes and the web application, and controlling access based on user privileges. When a slave node reads an iButton KeyFOB using an iButton reader [1], the corresponding ID is transmitted to the Master node via the nRF24L01 [2] wireless module. The Master node then verifies the ID's access level against the database. Depending on the verification outcome, the Master node sends a message back to the originating slave node, either granting or denying access.

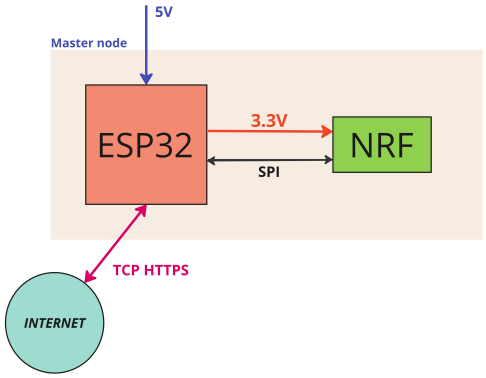

Figure 2.1: Master node block diagram

The Relay slave node (Figure 2.2) is implemented to manage access to power tools. Upon successful verification of an iButton ID [3] by the Master node, the Relay slave node activates or deactivates a relay, thereby allowing or denying usage of the power tool connected to it. This mechanism ensures that only users with the required access level can operate the power tools.

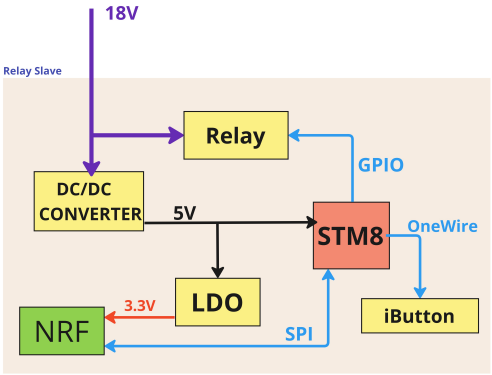

Figure 2.2: Relay slave block diagram

The Hall sensor slave node (Figure 2.3) regulates access to the tool drawers. It also utilizes an iButton reader to identify an iButton key FOB and submits the ID to the Master node for verification. Once access is granted, the user can open the drawer. Additionally, the Hall sensor within this node monitors the state of the drawer, detecting any changes when opened or closed. This data is then sent to the Master node, which stores it in the database with a timestamp, allowing for subsequent tracking of tool usage.

The web framework (Figure 2.4) connects the hardware system with users, offering an engaging platform for administrators to oversee the tool room system. Utilizing either a web or Command Line Interface (CLI) application, administrators can examine activity logs, modify user access levels, and observe the condition of tools in the room. The application interacts with the Master node via HTTPS (web app) or TCP (CLI) and shares a unified database, guaranteeing smooth data synchronization throughout the system.

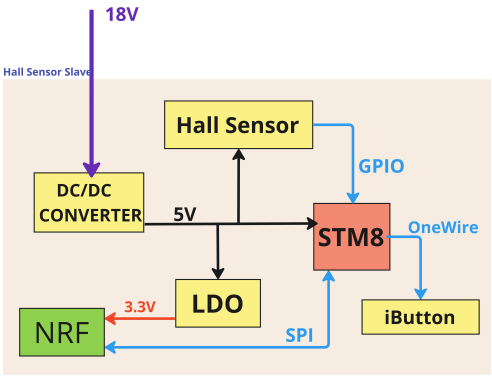

Figure 2.3: Hall sensor slave block diagram

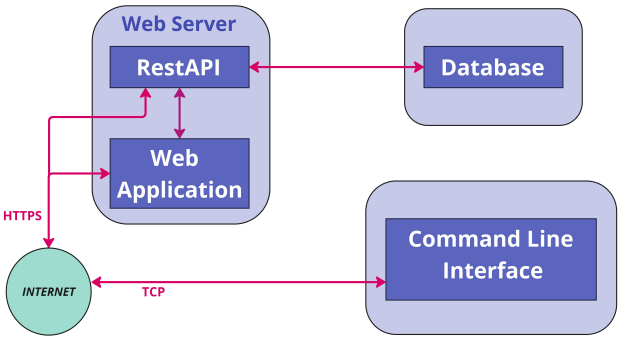

Figure 2.4: Web framework

Figure 2.5 represents the whole system combined, including the communication between all previously described parts.

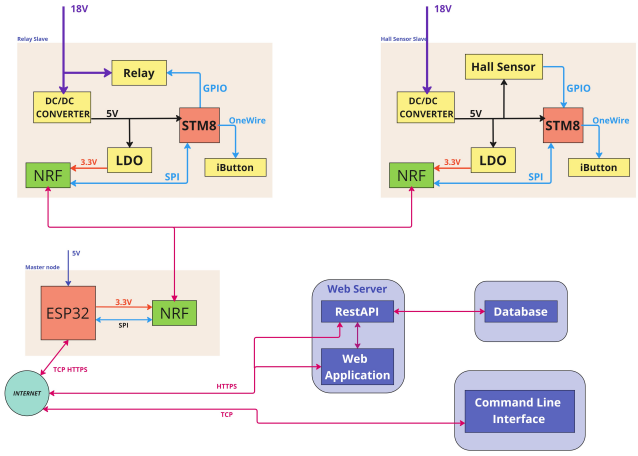

Figure 2.5: Top level block diagram

This complete system improves security and control over access to an equipment room and offers a reliable tracking mechanism to prevent tool theft or abuse and promote user accountability.

3 IMPLEMENTATION

3.1 Hardware implementation

3.1.1 General PCB design

The central foundation of the PCB design is comprised of four core elements: the STM8S105K4 microcontroller [4], a switching DC/DC buck regulator that lowers the power supply voltage, between 14-20V, down to 5V, a linear regulator that further reduces this voltage from 5V to 3.3V, and a DC Barrel jack that facilitates power connection. This two-stage design was chosen to allow for better efficiency over a normal linear voltage regulator. Depending on the specific requirements of each PCB, additional components are incorporated into the design.

The STM8S105K4 microcontroller is the main building block of our PCB. This microcontroller was chosen for its excellent functionality and capabilities in its price range. The low power consumption and good availability were also deciding factors in choosing this specific microcontroller. For each pin of the microcontroller, the female header was included on the PCB. This was done to ensure easy access and flexibility in hardware configuration while developing these boards.

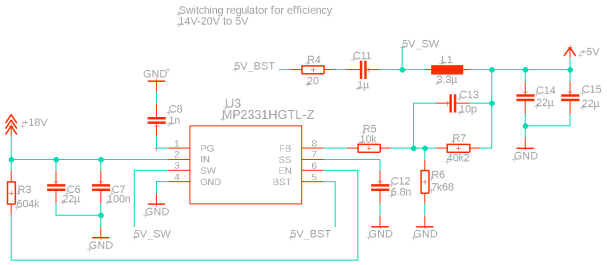

Figure 3.1: Switching regulator schematic

Power is supplied to the PCBs via the DC Barrel jack, providing an 18V input from the battery. Given the varying voltage requirements of the different components on the board, two stages of voltage regulation are employed. The first stage uses a switching voltage regulator (Figure 3.1) to step down the 18V input to a more manageable 5V. This 5V output is utilized to power the STM8S105K4 microcontroller and, in the case of the Hall sensor slave, the hall sensor itself.

The second stage of voltage regulation employs a linear regulator, which steps down the 5V to 3.3V. This lower, and more stable voltage is used to power the nRF24L01 transceiver module, which is required for wireless communication between the Master node and the slave nodes.

Both the Hall sensor slave and Relay slave PCBs incorporate OneWire connectors in their designs. However, the Relay slave, designed for power control rather than sens-

ing, does not incorporate any sensors in its architecture. The relay in this slave module is powered directly from the 18V supply to allow for higher power delivery to the tool.

In creating the board layout, great care was given to maintaining signal integrity. The bottom layer of the board is dedicated entirely to a ground plane, creating a common reference point for all signals. High current paths, such as the one leading to the relay and the 5V line, are routed using polygons to ensure minimal resistance and heat generation.

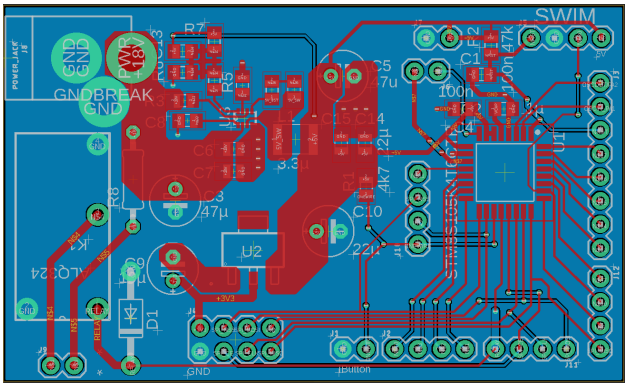

Figure 3.2: General PCB layout

3.1.2 Relay slave

The ”Relay slave” (Figure 3.3) is used to connect or disconnect the power of the machines in the equipment room. As for the other slave node, there is a power section that powers the slave. According to the access level, the STM8 gives an output to a pin to which the relay is connected. According to the output of the STM8, the relay either opens or closes, thus allowing or denying power to the machines or tools.

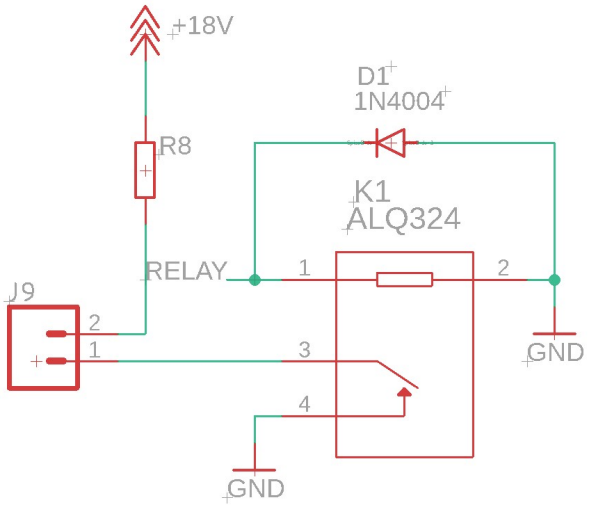

Figure 3.3: Relay schematic

3.1.3 Hall sensor slave

The design of the ”Hall Sensor slave” (Figure 3.4) follows the same general principle as the other slave node. A power section powers all parts of the slave, then the STM8 reads a Hall Sensor to determine if the drawer that this slave is mounted to is open or closed. A Hall Sensor is an active sensor that can detect the presence of a magnetic field. By mounting a magnet to the side of the drawer and positioning the slave node accordingly, the magnetic field of the magnet can be detected, allowing for the determination of whether the drawer is in its nominal position or not.

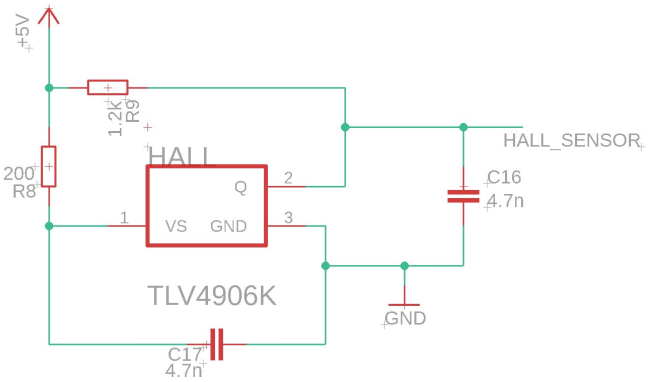

Figure 3.4: Hall Sensor schematic

3.2 Software implementation

Considering the two different microcontrollers employed, distinct software solutions are required for each architecture. As previously mentioned, the slave nodes incorporate an STM8S microcontroller, which is programmed in C using the ST Visual Develop (STVD) Integrated Development Environment (IDE). The master node, on the other hand, utilizes a more powerful ESP32 microcontroller [5], programmed in C with the ESP-IDF in Visual Studio Code.

Additionally, a web application and a CLI interface with the same functionality were developed for administrators. Both the web application and the CLI interface enable administrators to manage the system and access its features. The web application was created using the Flutter framework and the Dart programming language, while the CLI interface is accessible through a raw TCP connection using tools like PuTTY. The complete implementation is outlined below.

3.2.1 STVD

The IDE provided by ST Microelectronics for programming STM8 devices is the ST Visual Develop (STVD) IDE [6].

This is a general purpose C development environment suited for creating STM8 applications. Coupled with the STM8 Standard Peripheral Library, all necessary functions are available. For this project a few General Purpose Input/Output (GPIO) pins are needed to connect the actuators and the OneWire iButton and the SPI interface to communicate with the nRF24 wireless module.

3.2.2 OneWire

The OneWire protocol [3] developed by Dallas Semiconductor is a communication protocol that uses only a single communication line for both power delivery and communication. This coupled with a ground connection enables it to reduce the clutter of wires that usually comes along with digital communication protocols. The implementation uses a GPIO pin to give the necessary commands to the iButton reader and read the response given by it. The program first issues a presence pulse to the reader, if an iButton device is present it will react to the presence pulse. When the reaction is detected by the STM8, the command 0x33 is sent to ask the iButton for its device-specific ROM value. This 64-bit long value, unique for every single OneWire device, contains all the information we need for this application. It starts with an 8-bit device family ID, followed by a unique 48-bit device ID, and lastly an 8-bit long CRC code. This value is read and the CRC code is checked to make sure we read the value correctly. Once read, the value is stored on the STM8 for further processing or transmission. This unique ID is used to identify each specific key FOB and grant the correct privileges when scanned.

3.2.3 nRF24L01

As previously mentioned, the communication between the master and slave nodes occurs over radio waves at approximately 2.4 GHz, utilizing the nRF24L01 wireless module. This module features an SPI interface for communication with microcontrollers, and by executing the appropriate commands, reliable information transfer between master and slave nodes can be achieved. A lightweight messaging protocol was developed for communication between the master and slaves, which supports messages needed for access as well as pinging and a means for the slave node to communicate its current state and any potential errors (refer to Appendix B for a more detailed description). This information can then be logged by the master node.

The process begins by enabling the necessary clocks, pins, and interfaces for communication with the nRF24L01. Following this, the nRF24L01 is configured to operate at the required frequency, and relevant registers are enabled to facilitate both sending and receiving. Finally, data can be read from and written to the nRF24L01 modules' built-in transmission and receiving queues, enabling communication with the master node. Furthermore, all registers are configured to enable comprehensive interrupt handling on the master node, encompassing auto TX_DS, RX_DR, and MAX_RT capabilities.

3.2.4 Hall sensor

The Hall sensor is an active component employed in our system to monitor the status of the drawer. Working in combination with a magnet, the sensor detects the magnet's pole and outputs either a '0' or '1' accordingly. The system has been designed such that the Hall sensor generates a '1' when the drawer is open and a '0' when it is closed. This sensor is interfaced with a GPIO pin of the STM8S105K4 microcontroller, which facilitates the reading of its value, thus enabling real-time drawer status monitoring.

3.2.5 Database

A MySQL relational database is used to store the necessary data as it provides relational integrity while offering ACID guarantees. As can be seen from the provided ERD(Figure 3.5) of the database, 5 tables and 3 views have been made. In the database, admin credentials are stored in the 'admins' table, logs in the 'logs' table, equipment information in 'equipment_types', and access levels in the 'access_levels' table. A table named 'equipment_levels' is used for mapping the equipment to its corresponding access levels. Moreover, 3 views are used by the web application to display tables with different types of equipment and their access levels.

3.2.6 Web Application

The web application is designed for administrators to manage the system. It was developed using Flutter [7] for a cross-platform experience. The application enables the administrators to do the following (please refer to Appendix E for functionality description):

- Delete or add tools
- Change access levels for separate tools
- Add new access levels

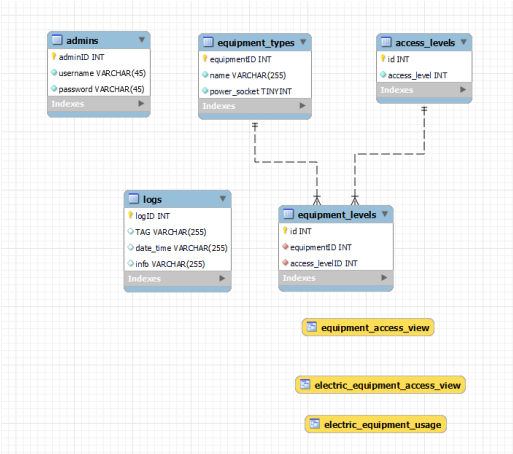

Figure 3.5: ERD of the database

- Remotely turn on or off the electrically powered tools
- See the logs of the system by date
- See the statistics to identify trends

Also, an HTTPS connection was used to establish communication between the master node and the application (refer to Appendix C for additional details on this communication).

Back-End

The back-end of the application was developed using PHP for server-side processing. PHP scripts were written to handle authentication, modification of access levels, and log retrieval from our MySQL database. The database stores information about administrators, access levels, and equipment usage logs.

Front-End

The front end of the application was developed using the Flutter framework in Google's Dart language [8]. It's a vast set of functionalities that allowed us to create a responsive and user-friendly interface. The layout of the application is designed with a clear and intuitive navigation system, allowing administrators to access different functionalities.

3.2.7 Master Node

The Master Node (Figure 3.6) was developed for the ESP32 microcontroller utilizing the ESP-IDF framework. The Master Node can be divided into two primary components.

- base functionality
- non-essential functionality

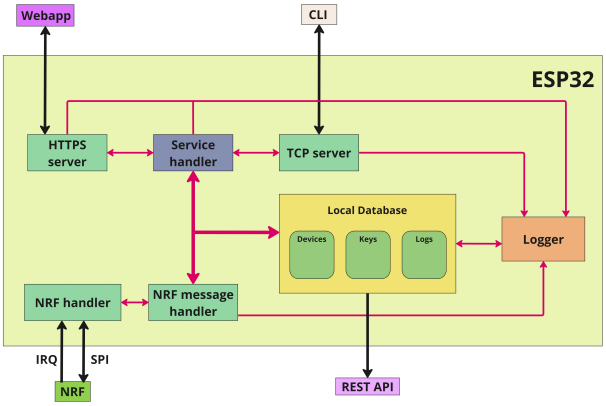

Figure 3.6: Master Node software architecture

Base Functionality

The first component is the base functionality, which supports the Slave Nodes in granting access to tools. This base functionality has been designed with robustness in mind and comprises two aspects.

The first aspect is related to the nRF24L01 control and message handling. A separate thread is dedicated to the constant monitoring of interrupts or internal requests for sending and receiving messages. This task ensures quick handling of messages, allowing for maximum scalability with as many Slave Nodes as possible. No logging is done within this task, as its sole purpose is the direct control of the nRF24L01. To handle the content of the messages and check access levels, an additional thread is running, which acts accordingly based on the received messages. This message handler thread is responsible for communicating with the local database, the logger, and other parts of the system as needed, ensuring seamless integration and coordination among different components.

The second aspect involves the local database copy of all device and key access levels. This copy is stored locally using the SPIFFS filesystem, ensuring that the Master Node's base functionality does **NOT** rely on a Wi-Fi connection. The local database is created with extensive error checking and has been tested extensively for maximal robustness. It also fully supports multithreaded access, allowing different parts of the system to access the database simultaneously without causing conflicts or issues. The database also provides the same functionality for storing logs; however, this feature will be discussed later, as it is not part of the base functionality.

Non-essential Functionality

The second primary component of the Master Node consists of non-essential features, predominantly dependent on a Wi-Fi connection. This component can be further subdivided into two parts. The first part, service handling of administrator requests, can be performed through two different interfaces: a web application that connects to the HTTPS server or a Command Line Interface (CLI) that connects to the TCP server. To provide these functionalities, a multithreaded TCP and HTTPS server is running on the ESP32. Both the TCP and HTTPS servers use the same underlying self-created service handler protocol, which streamlines the process of making changes or extending functionalities in the future. Due to the shared protocol, swapping or modifying server functionality would require minimal effort (refer to Appendix C for additional details on the functionality provided).

The CLI is preferably used via a program like PuTTY, which allows users to interact with the Master Node through a convenient terminal interface. For the correct setup of PuTTY, please refer to Appendix D. This functionality allows for actions such as remotely locking or unlocking all devices, which can be particularly useful for teachers who need to manage access from various locations.

The second part of the non-essential functionality focuses on logging. The Master Node records any user interaction with the Slave Nodes or Master Node itself. Similar to the access levels, logs are stored locally using SPIFFS [9]. As a result, even if a temporary Wi-Fi disruption occurs, logs will not be lost. Logs are uploaded to the database using a REST API once per day (at midnight) and are only deleted from local storage upon successful upload. Due to the potentially large number of logs (bytes) to be uploaded at once and the limited RAM on the ESP, an efficient batch system is employed. This system determines the available heap memory and creates appropriately sized batches of logs to avoid overloading memory, uploading all logs in manageable batches. This ensures a smooth and efficient log uploading process without causing memory issues on the device.

3.2.8 Slave Node (Finite State Machine)

1. Idle State: The initial state where all nodes are inactive, waiting for an iButton to be read by OneWire. In this state, the drawers are closed, power tools are off, and the web application is ready for interaction.

2. iButton Read State: Triggered when an iButton is read by the OneWire in either the Relay slave or Hall sensor slave. The ID from the iButton is transmitted to the Master node.

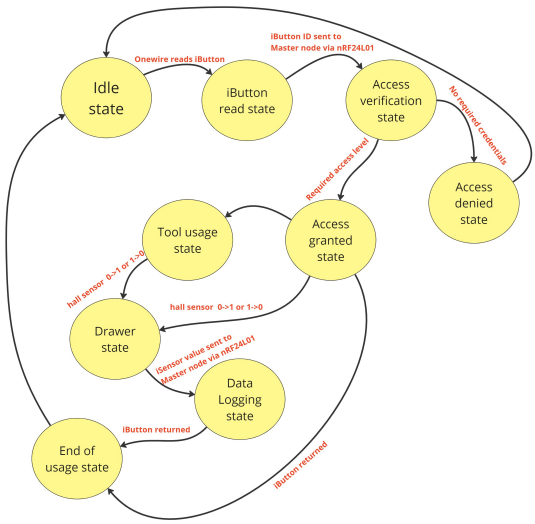

Figure 3.7: FSM of the system

3. Access Verification State: The Master node verifies the ID's access level against the database.

4. Access Granted State: If the ID is verified and the user has the necessary access level, the Master node sends a signal to the corresponding slave node to grant access.

5. Access Denied State: If the ID is not verified or the user lacks the necessary access level, the Master node sends a signal to deny access, and the system returns to the Idle State.

6. Tool Usage State: Once access is granted, the corresponding action takes place (tool is used, drawer is opened).

7. Drawer Opened State: This state is entered when the Hall sensor in the drawer detects a change in its state (closed or opened). The Hall sensor then sends this data to the Master node.

8. Data Logging State: In this state, the Master node logs the Hall sensor data and the timestamp in the database, recording which drawer was opened, when, and by whom.

9. End of Usage State: Once the user stops using power tools or drawers, the system returns to the Idle State.

4 EVALUATION AND VALIDATION

The evaluation and testing is an essential part of the project to make sure that both the separate modules and the whole prototype work successfully. Therefore, a great

emphasis was placed on conducting the necessary evaluation and testing of the code.

4.1 Testing and Test Framework

To ensure thorough testing of the master node's functionality, a dedicated test framework was created. This framework includes both unit testing and feature testing, covering various aspects of the master node's operations.

To provide flexibility and ease of use, the test framework was integrated with kconfigs in ESP-IDF. This allows developers to selectively compile, flash, and run specific tests by simply configuring the desired tests in Menuconfig. The use of kconfigs simplifies the test selection process and streamlines the testing workflow.

By leveraging the test framework, developers can effectively validate the performance and reliability of the master node. It provides a structured and systematic approach to test the master node's functions independently, ensuring that each component operates as intended and meets the required specifications. We have provided unit and feature tests for all parts of the master node code as it currently exists. This framework allows for the easy addition of more tests if extra functionality is added to the complete system.

4.2 Master Node limitations

1. Memory Constraints: The ESP32 microcontroller used in the Master Node has limited memory, which may affect the number of logs that can be stored locally. This could result in older logs being deleted to make room for new logs if the device is unable to connect to a Wi-Fi network for an extended period.
2. Wireless Communication Range: The Master Node's wireless communication with Slave Nodes using the NRF protocol is limited by the transmission range of the NRF module. Physical barriers, such as walls or other obstacles may reduce the effective range and require additional consideration during system deployment. This would not pose an issue in module 4 at Group T, but it is important to be aware of this limitation when deploying the system in a different location.
3. Dependency on Wi-Fi Connectivity: While the Master Node can perform its base functionality without a Wi-Fi connection, certain non-essential features, such as remote control and logging, rely on an active Wi-Fi connection. Temporary loss of connectivity may limit the availability of these features.
4. Unencrypted Communications: The TCP and NRF connections used by the Master Node are not encrypted. This could expose the system to potential security risks, such as eavesdropping or man-in-the-middle attacks. To mitigate these risks, additional security measures, such as implementing encryption or secure communication protocols, should be considered. Implementation of these features was out of the scope of this course but should be considered when a real deployment is needed.

4.3 Power consumption

Power consumption is an essential part of every electronic system as it directly impacts efficiency and operational costs. Therefore, the power consumption of each PCB was tested separately under different circumstances. It is worth noting that the power consumption when the NRF is receiving data (Figure 4.3) is lower than in the idle state (Figure 4.1) . This is caused by the iButton reader continuously sending a presence pulse to the iButton reader to detect when a key-FOB is presented. Assuming the power is delivered by an 18V 4Ah battery, we can expect a battery life of 2000 hours in the idle state. Even when many messages are sent over the NRF24 wireless module, the battery life should realistically not drop below 1800 hours. If we assume the system is enabled 16 hours per day and is used every single day, without exceptions, we expect a battery life of 112 days. Factoring in holidays and a more realistic workload a battery life of 5 months is not unreasonable.

	I (mA)	V (V)	P (mW)
Relay Slave	21	18.04	379
Hall sensor Slave	19	18.01	342

Figure 4.1: Power consumption when nRF24L01 is not in deployment

	I (mA)	V (V)	P (mW)
Relay Slave	18	18.04	342
Hall sensor Slave	19	18.01	325

Figure 4.2: Power consumption when nRF24L01 is sending data

	I (mA)	V (V)	P (mW)
Relay Slave	21	18.04	379
Hall sensor Slave	19	18.01	342

Figure 4.3: Power consumption when nRF24L01 is receiving data

5 DISCUSSION

This project's primary emphasis is creating a solution to the problem of access management in tool rooms on campuses, specifically in KU Leuven Campus Group T. The designed system demands to be user-friendly and reliable, but also cost-effective in terms of initial setup and maintenance. According to the system evaluation and testing, the slave nodes work reliably and efficiently because of the focus on efficient power delivery and clean, debuggable code. Moreover, the master node consumes very little energy when inactive, thanks to a strict no-polling rule followed during development. The robust, Wi-Fi-independent logging and access system is a significant contributor to the reliability of the full system, and its importance cannot be understated when delivering this system to the customer. The expected battery life of 5 months fits well within the academic setting of Group T, where a change of batteries once per semester for each slave node can be combined with a full system check.

However, it is essential to acknowledge the limitations of the master node, as outlined in the previous section. The memory constraints, wireless communication range, dependency on Wi-Fi connectivity for certain features, and unencrypted communications pose potential challenges in certain deployment scenarios or when considering future expansion of the system. Nevertheless, through system evaluation and testing, it can be shown that the implemented problem solution meets its initial expectations and provides a reliable and efficient access management system for KU Leuven Campus Group T.

6 CONCLUSION

In conclusion, our team has implemented an access control system for Group T's module 4, allowing students to use the tools provided in the room in a safe and controlled environment without constant supervision by a teacher. The implemented system uses different access levels that can be assigned to each student separately based on their current education level and received training. The student receives a key-FOB when entering the room which can be scanned at every drawer and power tool in the room. The system grants the student access based on their access level and the tool's or drawer's required access level. All events are logged by the system and can be controlled through CLI or a user-friendly web application, enabling the staff of group T to easily check all recent activity in the tool room and control the system. The implementation of this system focuses on efficient power delivery and works even without a stable Wi-Fi connection. All slave nodes communicate to the master node which in turn can access a database that contains the necessary access levels and a table of logs. When the Wi-Fi is down, the system remains functional and holds on to all logs that should be uploaded to the database until the Wi-Fi connection is re-established. The energy efficiency is also a key point of the design, using a standard commercial battery, all slave devices can be powered for around 5 months before requiring a battery switch.

ACKNOWLEDGEMENTS

We want to thank the teaching staff of Group T for their continuous technical support when developing this system, Mr. Stevens was invaluable in keeping us up to date with the most relevant information and supplied us with answers to any questions we had regarding this project. We also want to thank Mr. Spaepen for giving us the opportunity to work on a system that is not only interesting but has a real use case.

LIST OF SYMBOLS

Acronyms

ACID	Atomicity Consistency Isolation Durability
CLI	Command Line Interface
CRC	Cyclical Redundancy Check
DC	Direct Current
ERD	Entity Relationship Diagram
FSM	Finite state machine
GPIO	General Purpose Input - Output
HTTPS	Hypertext Transfer Protocol Secure
IDE	Integrated Development Environment
MAX_RT	Maximum Retransmit
PCB	Printed Circuit Board
PHP	PHP: Hypertext Preprocessor
RX	Receive
RX_DR	Receive Data Ready
SMD	Surface Mount Device
SPI	Serial Peripheral Interface
SPIFFS	Serial Peripheral Interface Flash File System
STVD	ST Visual Develop
TCP	Transmission Control Protocol

TX	Transmit
TX_DS	Transmit Data Sent
UART	Universal Asynchronous Receiver-Transmitter

BIBLIOGRAPHY

[1] A. Devices, “DS9092 iButton Probe | Analog Devices.” [Online]. Available: https://www.analog.com/en/products/ds9092.html#product-overview

[2] N. Semiconductor, “NRF24L01 Datasheet(PDF) - Nordic Semiconductor.” [Online]. Available: https://www.alldatasheet.com/datasheet-pdf/pdf/1489848/NORDIC/NRF24L01.html

[3] A. Devices, “Reading and Writing 1-Wire Devices Through Serial Interfaces | Analog Devices.” [Online]. Available: https://www.analog.com/en/app-notes/reading-and-writing-1wirereg-devices-through-serial-interfaces.html

[4] S. Microelectronics, “Stm8s105k4.” [Online]. Available: https://www.st.com/en/microcontrollers-microprocessors/stm8s105k4.html

[5] Espressif, “ESP32 Wi-Fi & Bluetooth MCU I Espressif Systems.” [Online]. Available: https://www.espressif.com/en/products/socs/esp32

[6] S. Microelectronics, “STVD-STM8 - ST Visual develop IDE.” [Online]. Available: https://www.st.com/en/development-tools/stvd-stm8.html

[7] Google, “Flutter GitHub.” [Online]. Available: https://github.com/flutter/flutter

[8] ——, “Dart documentation.” [Online]. Available: https://dart.dev/guides/index

[9] Espressif, “SPIFFS Filesystem - ESP32 - ESP-IDF Programming Guide latest documentation.” [Online]. Available: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/storage/spiffs.html

APPENDICES

APPENDIX A : SCHEMATIC

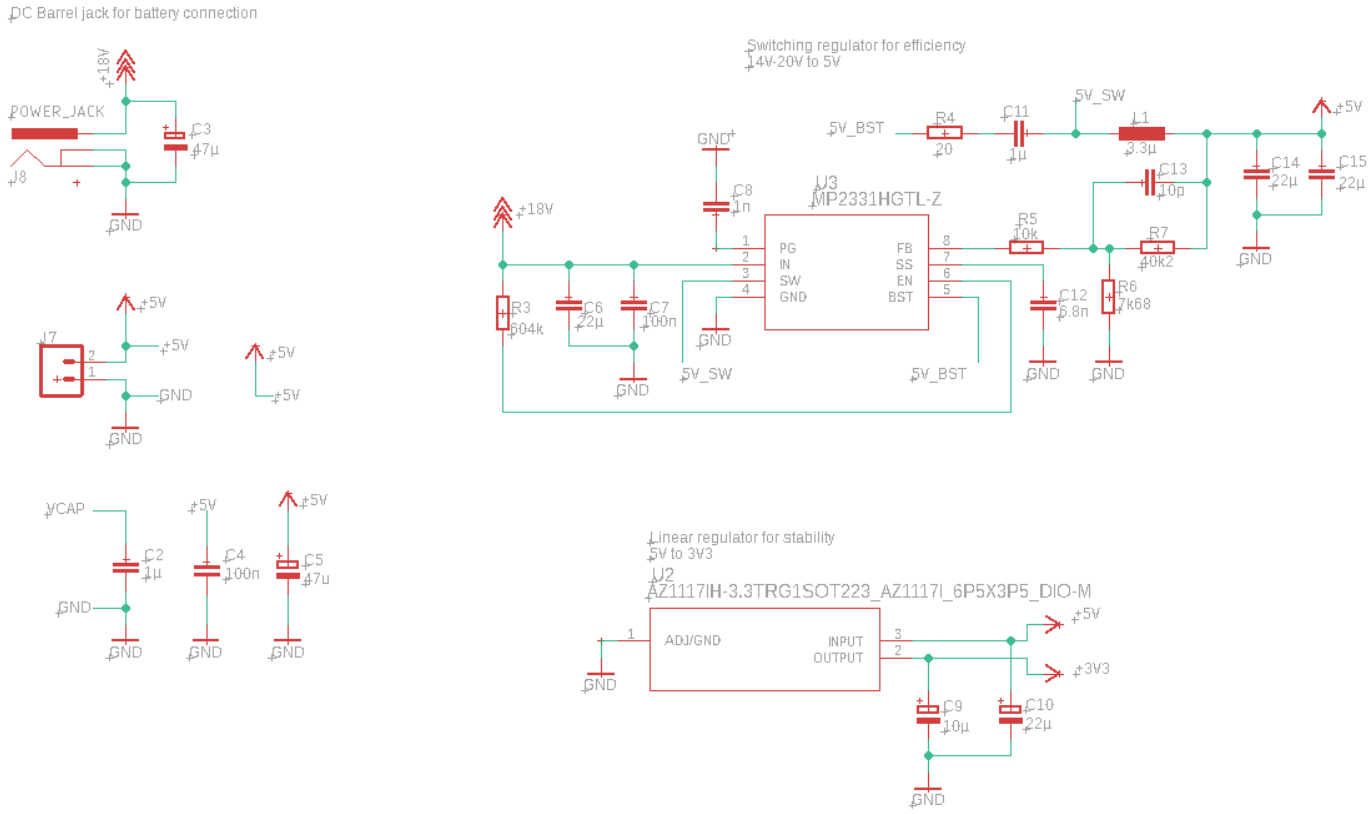

Figure A.1: Schematic sheet 1

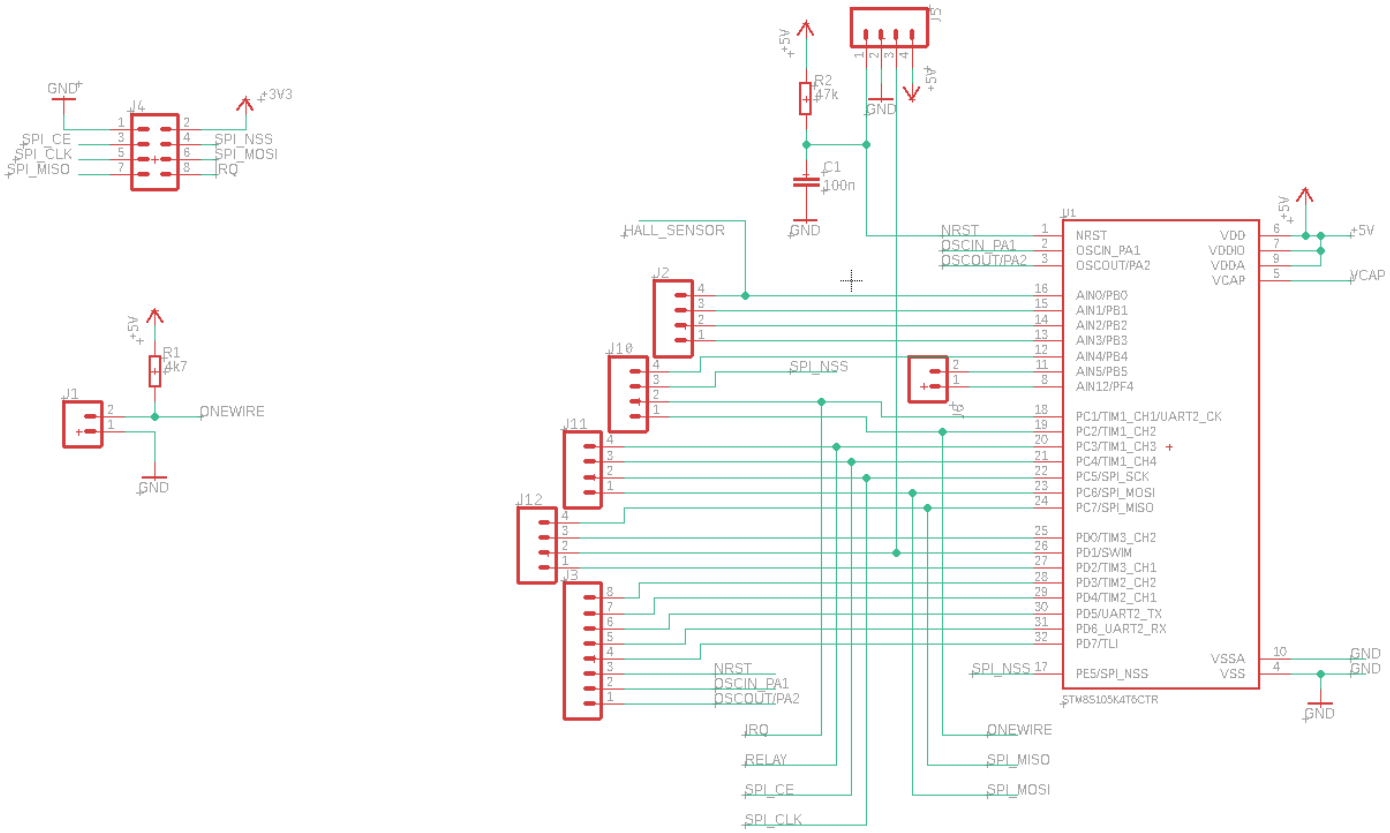

Figure A.2: Schematic sheet 2

APPENDIX B : MASTER-SLAVE MESSAGE PROTOCOL

This appendix briefly describes the lightweight messaging protocol for communication between master and slave nodes. This protocol sits on top of the standard nRF message protocol employed by the nRF modules to facilitate communication.

B.1 Protocol Structure

- 1st byte: Message type (ASCII encoded, 0: access check, 1: ping)
- 2nd-3rd bytes: Slave node device number (ASCII encoded)
- Data (up to 8 bytes): Depends on message type

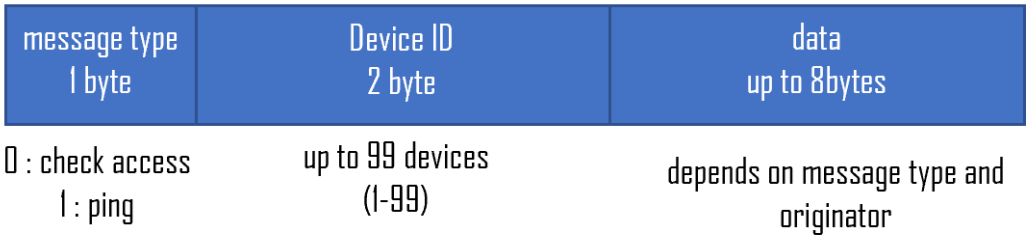

Figure B.1: Master-Slave protocol structure

B.2 Access Check Message

Data (8 bytes) contains the key value. The master node verifies this key against its internal database and responds with a 1-byte data indicating access status (1: access granted, 0: access denied). The master node can also initiate an access check message to control the slave node's equipment.

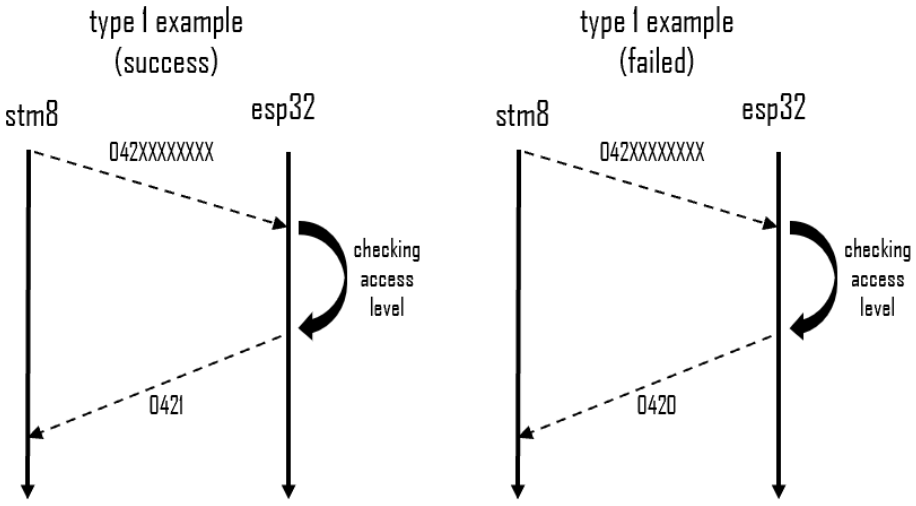

Figure B.2: access check message example

B.3 Ping Message

Data (ASCII encoded, 1 byte) represents the error state or specific request from the slave node or master node. Ping messages facilitate regular status updates and communication between nodes, allowing error monitoring and maintenance of the system's overall health.

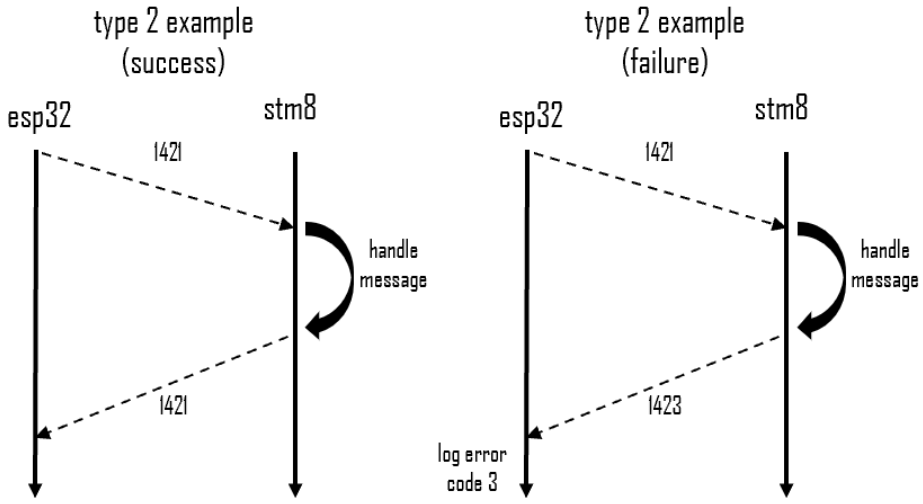

Figure B.3: ping message example

APPENDIX C : MASTER NODE SERVICE REQUEST FUNCTIONALITY

This appendix provides an overview of the master node's service request functionality, which is the underlying protocol used by both the Command Line Interface (CLI) and the web application. This functionality allows administrators to control and manage the master and slave nodes in the system.

C.1 Service Request Types

There are four types of service requests supported by the master node:

- TURNON
- TURNOFF
- SYNC
- ACCESSLEVEL

C.2 TURNON and TURNOFF

Both TURNON and TURNOFF service requests require an extra parameter, the device ID (1-99). When the device ID is set to 0, the request will apply to all devices. The TURNON request enables the specified device(s), while the TURNOFF request disables them. These requests allow administrators to control the operation of the master and slave nodes.

C.3 SYNC

The SYNC service request requires no extra parameters. When this request is issued, the master node will push local logs to the server, synchronizing the stored data. This allows administrators to maintain up-to-date records of the system's operation.

C.4 ACCESSLEVEL

The ACCESSLEVEL service request requires four extra parameters:

1. The action to perform: change (0), add (1), or delete (2) a device/key.
2. The target to adjust: device (0) or key (1).
3. The target's ID: for a device (1-99), or a key (8-byte hexadecimal).
4. The new access level (0-9).

This request allows administrators to modify the access level of devices and keys within the system, granting or revoking access as needed. Administrators can use this functionality to manage the overall security and operation of the master and slave nodes.

C.5 Web Application and CLI Implementation

For the web application, this protocol functionality is used through POST requests and JSON files, while for the CLI, the functionality is accessed through commands.

In the CLI, since it does not connect directly to the database, additional commands are available to pull logs, devices, and keys from the local database stored on the master node. This allows administrators to manage the system and retrieve relevant information without requiring direct access to the central database.

APPENDIX D : PUTTY CONFIGURATION

1. Download PuTTY from the official website: `https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html`.
2. Install PuTTY on your system.
3. Launch PuTTY.
4. In the PuTTY Configuration window:
    - In the ”Host Name (or IP address)” field, enter the static IP address assigned to the Master Node by your router.
    - Set the ”Connection Type” to ”Raw”.
    - Enter ”3333” as the port number.
5. Navigate to the ”Terminal” category in the left-hand menu:
    - Enable the ”Implicit CR in every LF” option.
    - Enable the ”Implicit LF in every CR” option.
6. Click the ”Open” button to initiate the connection.

Please note that the static IP address for the Master Node should be defined by your router. By configuring PuTTY with the provided settings, you will be able to establish a CLI connection with the Master Node

APPENDIX E : WEB APPLICATION FUNCTIONALITY

This appendix provides an overview of the admin web application functionality. This functionality allows administrators to control and manage the system.

E.1 Screens

The application in total provides the following functionality:

- Deleting tools
- Adding tools
- Change access levels for separate tools
- Add new access levels
- Remotely turn on or off the electrically powered tools
- See the logs of the system by date
- See the statistics to identify trends

E.2 Deleting tools

By deleting the tool from the system, it is deleted from both the MySQL database and the master node's local memory by sending the corresponding instructions through the HTTPS connection.

E.3 Adding tool

By adding the tool to the system, it is added both to the MySQL database and the master node's local memory by obtaining its ID in the schema table and sending the corresponding instruction with an ID. It is required to provide the name, access level, and the need for a power socket during addition.

E.4 Access Level Change

Changing the access level for separate tools notifies the master node and updates the 'equipment_types' and 'equipment_levels' tables in the database. Please see Figure E.1.

E.5 Adding access level

Adding a new access level updates the 'access_levels' table in the database. Please see Figure E.2.

E.6 Controlling electrically powered tools remotely

Turning the tool 'ON' or 'OFF' sends the corresponding instructions to the master node. Please see Figure E.3.

E.7 Logs

The app sends an instruction to the master node to upload the latest logs to the database when the user arrives at the 'History' page. To see the logs, the user should specify the date. The logs are then fetched from the 'logs' table in the database with the corresponding date.

E.8 Statistics

The user finds the statistics on the 'Dashboard' screen of the app. The app is scalable, therefore, new metrics can be added for statistics depending on the needs of the user.

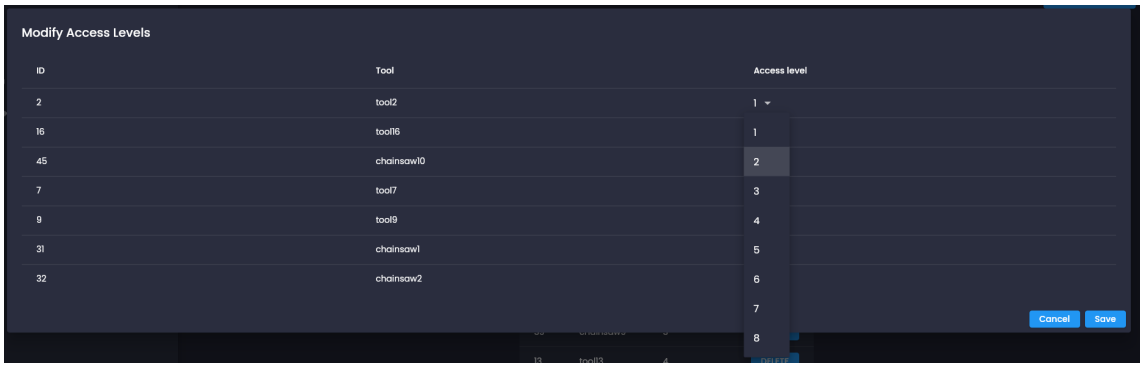

Figure E.1: Changing access level for tools

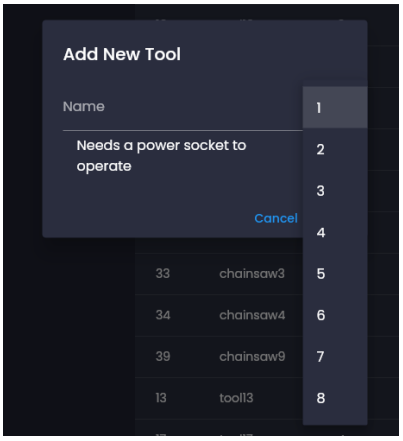

Figure E.2: Modal window for tool addition

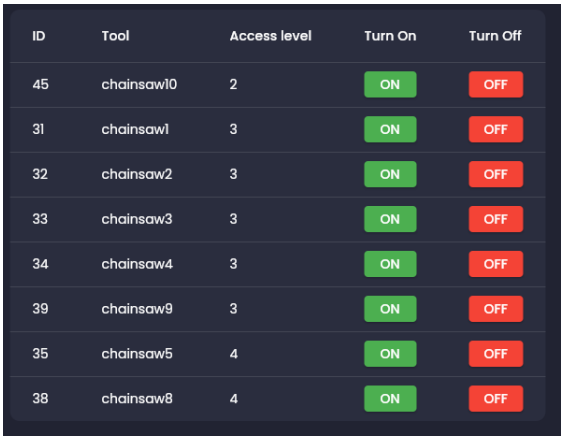

Figure E.3: List of tools with options to turn ON/OFF