GROUP 07

MARK FERNANDO, VISAL SAMARAKOON, SHAMILA THENNAKOON

Garage Door

PROJECT REPORT

School of IT

Metropolia University of Applied Sciences

16 March 2025

Abstract

This project report presents the design, implementation, and testing of a Garage Door Opening system that seamlessly integrates local and remote-control functionalities. The system employs a stepper motor to drive the door, a rotary encoder to track its position and direction, and limit switches to define the fully open and closed positions. Users can operate the system locally via physical buttons or remotely through MQTT communication, enabling real-time monitoring and control from a distance.

Key features of the system include an automatic calibration process to determine the door's movement limits, status updates (door state, error state, and calibration state) via MQTT, and error handling to detect and respond to issues such as obstructions or malfunctions. The system also provides local status feedback using LED indicators, with blinking LEDs signaling error conditions. The implementation leverages object-oriented programming principles, ensuring modularity, ease of maintenance, and scalability.

Comprehensive testing confirmed that the system meets all specified requirements, including precise calibration, seamless local and remote operation, and accurate status reporting. The project successfully integrates hardware and software components to deliver a reliable and user-friendly garage door automation solution. Future enhancements could include expanded remote-control capabilities and integration with smart home platforms, further enhancing the system's versatility and usability.

Contents

Introduction

The Garage Door Opener project aims to design and implement a functional, secure, and user-friendly system for automating garage door operations. The system integrates local and remote-control functionalities, allowing users to operate the door both manually via physical buttons and remotely through an MQTT-based communication network. This dual-mode operation ensures flexibility and convenience, enabling real-time monitoring and control from any location.

At the core of the system are key hardware components, including a stepper motor for precise door movement, a rotary encoder to track the door's position and direction, and limit switches to define the fully open and closed positions. These components work in tandem to ensure smooth and reliable door operation. Additionally, the system incorporates status reporting and error handling mechanisms, providing users with real-time updates on the door's state and detecting issues such as obstructions or malfunctions.

The project is structured using object-oriented programming principles, ensuring modularity, scalability, and ease of maintenance. By encapsulating hardware functionalities into well-defined software modules, the system achieves a high level of reliability and adaptability. The implementation also includes local status indicators using LEDs, with blinking patterns signaling error conditions, enhancing the system's usability and safety.

This report outlines the project specifications, design considerations, implementation details, and testing results, demonstrating how the system meets both functional and advanced requirements. The successful integration of hardware and software components results in a robust and efficient garage door automation solution, with potential for future enhancements such as expanded remote features and integration with smart home platforms.

Methods and Material

In this section, the methodology and components utilized in the project will be outlined.

Hardware Components:

Garage Door Opener

The garage door opener consists of the following parts. a stepper motor to move the door up and down, a rotary encoder detect door movement and movement direction, two limit switches to detect when the door has reached the end of the movement range, and a door. The door in our test setup is a plastic block that is driven back and forth by the motor.

Stepper motor

A stepper motor is a type of electric motor designed to rotate its shaft in precise, fixed increments, or steps. This characteristic is achieved through the motor's internal design, enabling the exact angular position of the shaft to be determined by counting the number of steps it takes, without requiring a sensor. This capability makes stepper motors well suited for a variety of applications.[1]

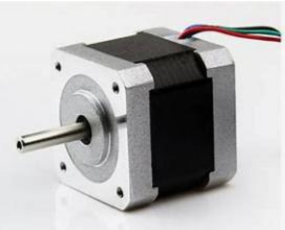


Figure 1: Stepper motor [2]

Rotary Encoder

A rotary encoder is an electromechanical device that converts the angular position or motion of a rotating shaft into analog or digital output signals. This conversion enables precise monitoring and control of parameters such as position, speed, and acceleration in various applications [3]

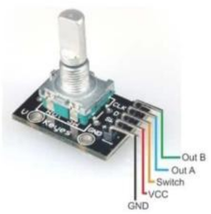


Figure 2: Rotary Encoder [3]

Limit Switches

A limit switch is an electromechanical device used to detect the presence or absence of an object or to monitor the position of a mechanical component through physical contact. When an object engages the switch's actuator, it activates internal contacts, opening or closing an electrical circuit. In the context of a garage door opener, limit switches are used to determine when the door has reached the end of its movement range.[4]

Figure 3: Limit switch

Controller PCB

For the project, the PCB controller was chosen as the microcontroller board. It is an electronic system designed to manage and regulate the functions of a printed circuit board (PCB). It includes a microcontroller (RP2040-based system) to controls the system and processes commands, LEDs to Indicate the status of the door locally, Buttons used for local control (SW0, SW1, SW2), power circuitry, and input/output (I/O) interfaces. By programming the controller, it can oversee tasks like data acquisition, signal processing, and power management. Its primary role is to handle the timing, sequencing, and overall operation of the PCB. [5]

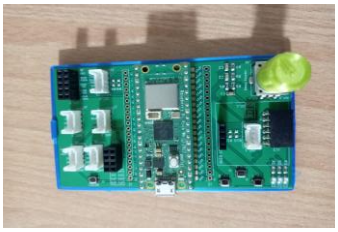


Figure 4: Controller PCB

Stepper motor driver

Stepper motor drivers are devices engineered to operate stepper motors, enabling precise position control and continuous rotation without the need for a feedback system. So, it can connect the JST connector to the 6-pin connector on the stepper driver board.[6]

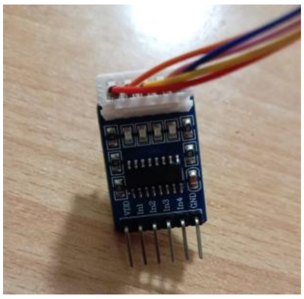
 

Figure 5: Stepper motor driver

I^{2}C EEPROM

EEPROM stands for Electrically Erasable Programmable Read-Only Memory. It is a type of memory that retains data even when power is off, making it non-volatile. It can be erased and rewritten electrically, allowing for multiple updates to the stored data. This makes it ideal for applications requiring flexible and persistent storage of small data sets, like configuration settings or system parameters. Unlike traditional ROM, which is static and unchangeable after manufacture, EEPROM provides the advantage of reprogramming as needed, enhancing its utility in modern electronic devices.[7]

Figure 6: I^{2}C EEPROM

Pico Probe Debugger

The Raspberry Pi Debug Probe is a USB device offering a UART serial port and a standard Arm Serial Wire Debug (SWD) interface. It is designed for effortless, solder-free, plug-and-play debugging.[8]

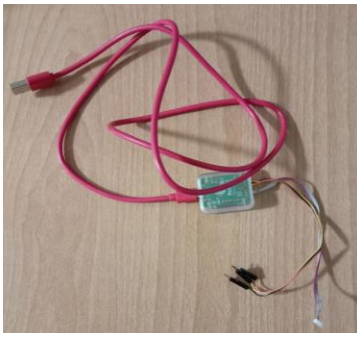
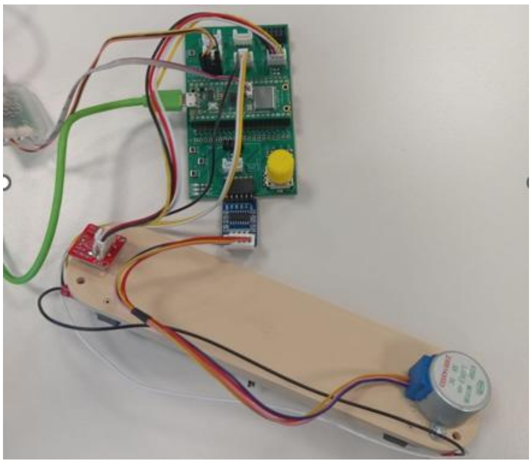
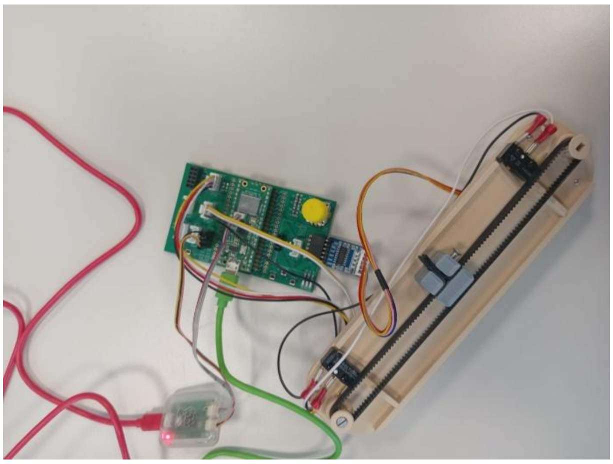
 

Figure 7: Pico Probe Debugger [8]

Here is the experimental setup we have designed for the Garage Door Opener,

Software Components

MQTT Communication

The Garage Door Opener project, MQTT is used to enable remote control and status reporting. The system publishes status updates (e.g., door state, error state) to the topic "garage/door/status" and subscribes to the topic "garage/door/command" to receive remote commands. This allows users to monitor and control the garage door from a remote location, ensuring seamless integration with IoT ecosystems.

Interrupt handling

Interrupt handling is a critical mechanism in embedded systems that allows the microcontroller to respond immediately to specific events, such as changes in input signals, without continuously polling the input pins. This approach is highly efficient, as it reduces CPU overhead and ensures timely responses to external events.[9] In the context of the Garage Door Opener project, interrupt handling is used to detect rotary encoder movements and sw1 button press to stop while moving, enabling precise control and monitoring of the garage door's movement and button press.

Object-Oriented Programming (OOP)

Object-Oriented Programming (OOP) is a way of designing software by organizing it into objects instead of just functions and logic. In the Garage Door Opener project, OOP is used to represent hardware components and their functions as objects.

- GPIOPin class declare pins and offer read and write options if needed.
- Button class is checking button press event with debounce and avoiding multiple events while pressing.

- ButtonManager class responsible for handling single and double press events.
- EEPROM class handles writing and reading data for state management.
- LED class handles LEDs functionality by using turns all on, turns all off and custom led handling.

Docker platform

Docker is an open-source platform that simplifies application deployment, scaling, and management through containerization. Containers are lightweight, portable units that include an application along with their dependencies, libraries, and configurations. Unlike virtual machines, Docker containers share the host operating system's kernel, making them more efficient in resource usage and startup time. [11]

Project Specifications

1. **Calibration**

- The system was required to calibrate the door's movement by running it up and down to determine the number of steps between the fully open and fully closed positions.
- Calibration was initiated by pressing SW0 and SW2 simultaneously.
- The system used limit switches and the rotary encoder to detect when the door reached the end of its movement range.
- After calibration, the door was not allowed to hit the limit switch bodies during normal operation.

2. **Status Reporting**
    - The system was required to report the following statuses via MQTT.
        - Door State: Open, Closed, or In Between.
        - Error State: Normal or Door Stuck.
        - Calibration State: Calibrated or Not Calibrated.
    - Status updates were published to the MQTT topic "garage/door/status".
3. **Local Operation**

The system was required to allow local control via physical buttons:

- SW1: Toggled the door's movement:
- If the door was closed, it started to open.

- If the door was open, it started to close.
- If the door was in motion, it stopped.
- If the door had been previously stopped, it resumed movement in the opposite direction.

The system provided local status indication using LEDs:

- Door open: Indicated by LED 1
- Door in between: Indicated by LED 2
- Door close: Indicated by LED 3
- Door calibrated: indicate by turn on all LEDs
- Door stuck: Indicated by a blinking all LEDs

4. **Remote Control**

The system was required to support remote control via MQTT:

- It subscribed to the topic "garage/door/command" to receive remote commands.
- It executed commands such as Open, Close, and Stop.
- Command responses were sent to the topic "garage/door/response".

5. **Error Handling**

If the door got stuck during movement, the system was required to:

- Stop the motor.
- Report an error state via MQTT.
- Revert to a Not Calibrated state.

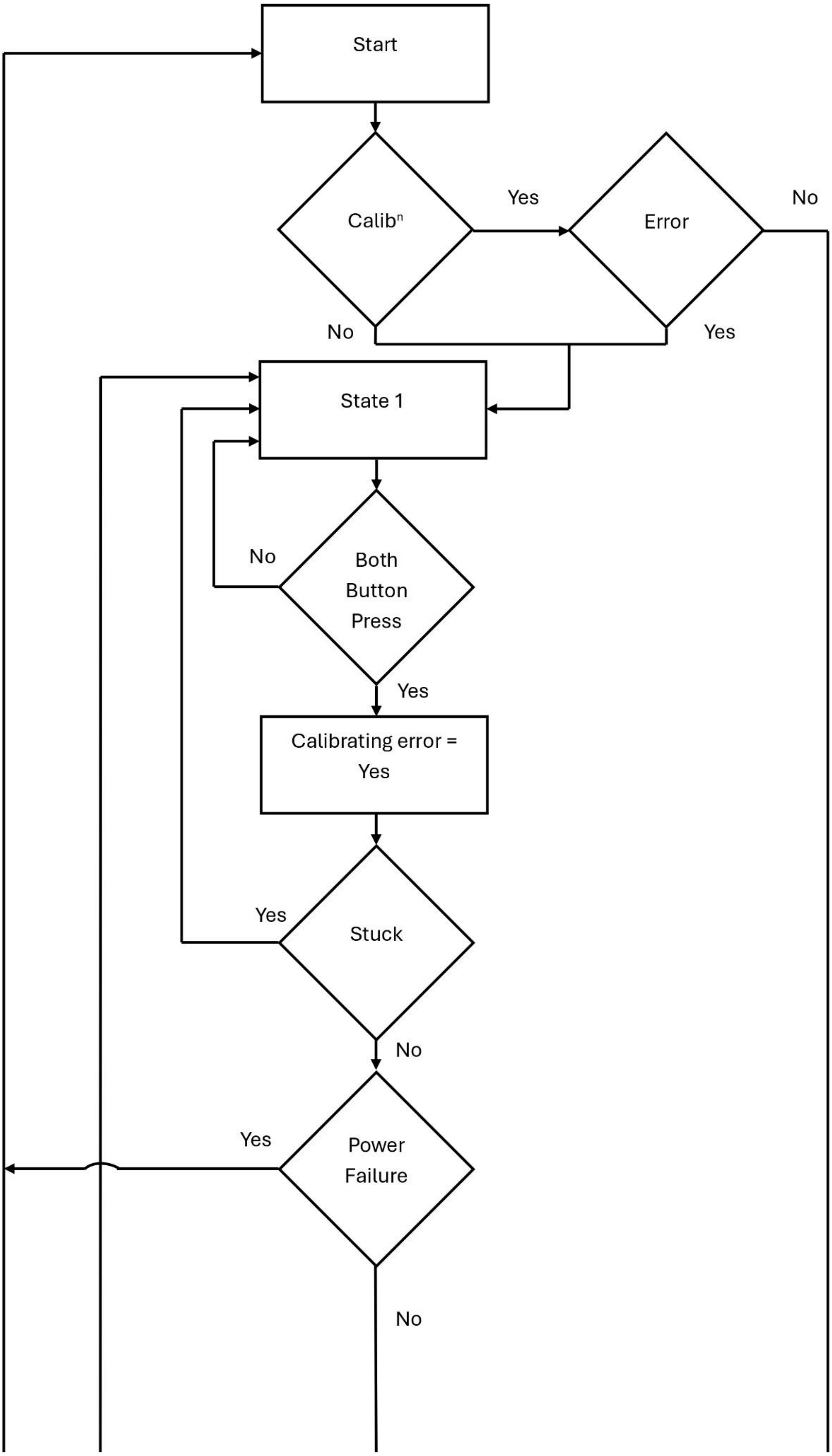
The system prevented local and remote operation in the Not Calibrated state.

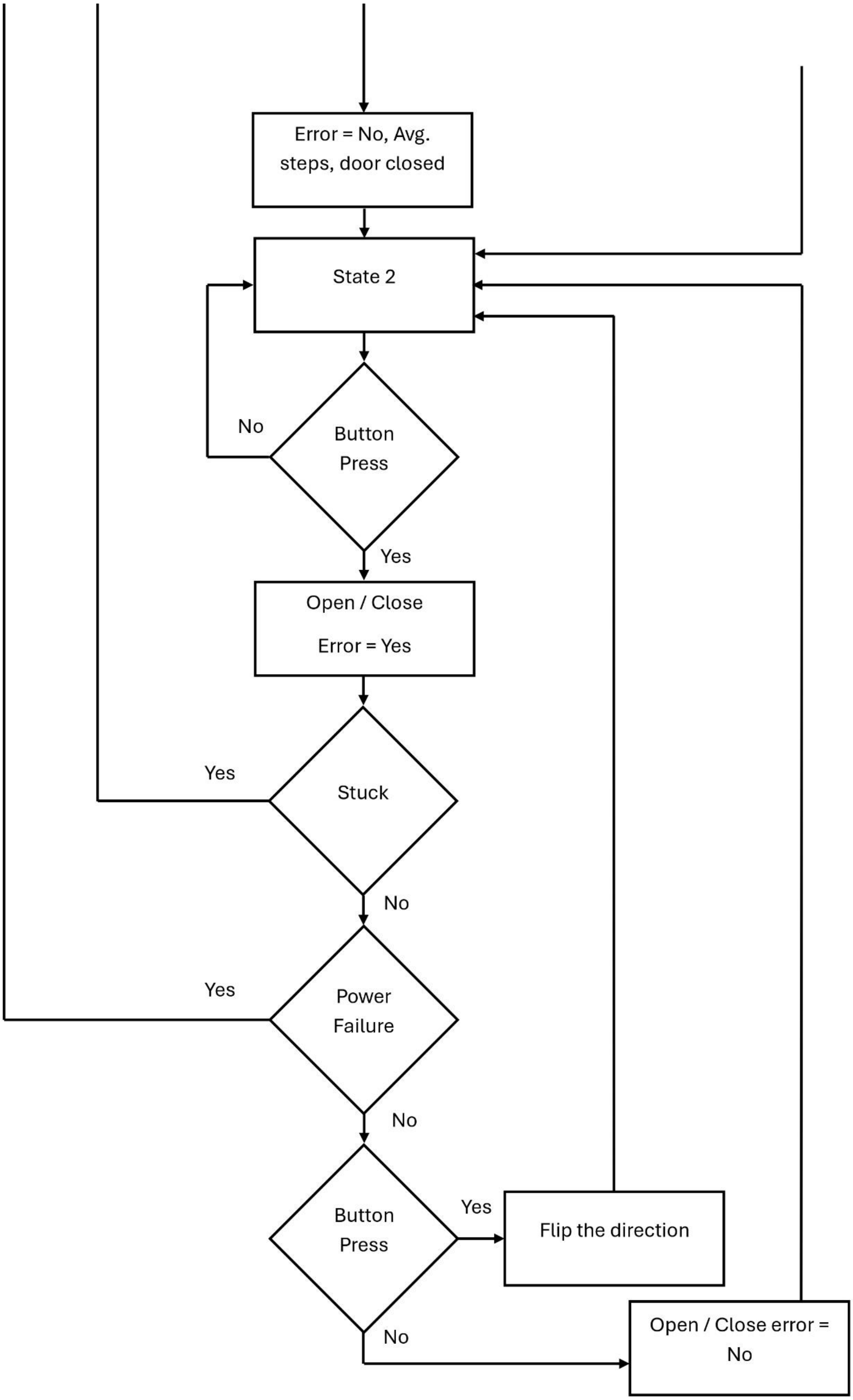
6. **State Management**

The system saves 5 states in EEPROM and performs specific tasks based on the current state.

- Door state: open, close, between
- Error state: error yes, error no
- Calibration state: calibration yes, calibration no
- Average steps: stores average steps
- Door direction: Store the door direction when it is in the in-between stage

Flow Chart


Start
Calibⁿ
Yes
Error
No
No
Yes
State 1
No
Both
Button
Press
Yes
Calibrating error =
Yes
Yes
Stuck
No
Yes
Power
Failure
No


Error = No, Avg. steps, door closed
State 2
Button Press
No
Yes
Open / Close
Error = Yes
Stuck
Yes
No
Power Failure
Yes
No
Button Press
Yes
Flip the direction
No
Open / Close error = No

Implementation

Wiring and Pin Assignments

The Garage Door Opener base is equipped with a 6-pin JST connector and a 4-pin Grove connector. The JST connector has been connected to the 6-pin connector on the stepper driver board, while the Grove connector is connected to a GPIO connector on the PCB.

- Stepper Motor Controller: GP2, GP3, GP6, GP13 (all outputs).
- LEDs: GP20 - GP22.
- Buttons: GP7 - GP9
- Rotary Encoder: GP 14 & GP 15 (Configured as a pull-up)
- Limit Switches: GP 27 & GP 28 (Configured as a pull-up)
- I2C: GP16 & 17 (Configured as a pull-up)

The software operates in a state machine consisting of two states. Case 1 is responsible for calibration, and Case 2 is responsible for door open/close operations. When the device powers on, it checks the EEPROM states, and based on the EEPROM states, it directs the system to the relevant case. In each case, a stuck detection is performed, and if the door gets stuck it immediately stops and recalibrates the door.

The stuck detection is implemented using both interrupt and time-checking methods. Each time the encoder detent is detected, a value is added to the queue. The difference between two values of the queue(interrupts) is continuously checked, and if it exceeds the tested time, a stuck condition is detected.

The stopping event during opening and closing is detected by the second interrupt. If a button interrupt is triggered, the button press flag is set to true. The direction is then toggled, and the system runs in the opposite direction if the button is pressed again.

The system monitors the door and button status, recalibrating if stuck. It toggles the door direction based on button press and uses interrupts for real-time direction and response.

Conclusion

The Garage Door Opener project achieved significant milestones, successfully implementing core functionalities such as local control, calibration, and basic status reporting. The system demonstrated reliable operation using stepper motor control, rotary encoder-based position tracking, and limit switches to ensure accurate door movement and position detection. However, due to unforeseen challenges and errors, the MQTT communication requirements were only partially fulfilled. While the system was able to publish basic status updates, some advanced remote-control features and command responses could not be fully implemented as initially planned.

Despite these limitations, the project showcased a strong foundation in object-oriented programming (OOP), ensuring a modular and scalable design. The use of OOP principles allowed for clear separation of responsibilities, making the system easier to maintain and extend in the future. Additionally, the integration of LED indicators provided effective local feedback, enhancing the system's usability.

The challenges encountered with MQTT communication highlighted areas for improvement and learning. Future work could focus on resolving these issues to fully implement remote control capabilities, including:

- Addressing the errors to enable complete remote command execution and response handling.
- Adding authentication and encryption to secure MQTT communication.
- Developing a user-friendly mobile application for remote monitoring and control.
- Expanding the system's integration with platforms like Home Assistant, Google Home, or Amazon Alexa.

In conclusion, while the project did not fully meet all MQTT-related requirements, it successfully delivered a functional and reliable garage door automation system with strong potential for future enhancements. The lessons learned from this project will serve as valuable insights for improving the system and achieving its full capabilities in subsequent iterations.

References

[1] Carmine Fiore ,**2020-stepper-motors-basics-types-uses-and-working-principles_r1.0_1,** PDF (media.monolithicpower.com). Accessed:16 March 2025

https://media.monolithicpower.com/mps_cms_document/2/0/2020-stepper-motors-basics-types-uses-and-working-principles_r1.0_1.pdf?_gl=1*1cg7t2q*_ga*MjAzMDkwNjU0NS4xNzM0MTk3MzQ3*_ga_XNRPF6L9DD*MTczNDE5NzM0NC4xLjEuMTczNDE5NzQwNi40LjAuMA..*_gcl_au*NDQwNDI5NTM2LjE3MzQxOTczNTE.&_ga=2.151768703.730536013.1734197347-2030906545.1734197347

[2] Nanotec, ST4118, Stepper Motor - NEMA 17(Data Sheet) . Accessed: 16 March 2025.

https://www.farnell.com/datasheets/2337767.pdf?_gl=1*jrwsxm*_gcl_au*MjA5ODk1MjY4Ni4xNzM0MTk4Mjg2

[3] What is Rotary Encoder? Construction & Working of Rotary Encoder- Mamtaz Alam, August 22, 2022. Accessed: 16 March 2025

https://how2electronics.com/construction-working-rotary-encoder/

[4] What is a Limit Switch? A Complete Guide. Accessed: 16 March 2025.

https://eshop.se.com/in/blog/post/limit-switch-guide.html

[5] Blog / PCB Controller: Everything You Need to Know – NextPCB. Accessed: 15 March 2025.

https://www.nextpcb.com/blog/pcb-controller

[6] Workspaces - OMA - Embedded Systems Programming TX00EX76-3005-document-project-pill-despenser (PDF).

https://oma.metropolia.fi/tyotilat?p_p_id=WorkspacePortlet_WAR_workspaceportlet&p_p_lifecycle=0&p_p_state=normal&p_p_mode=view&_WorkspacePortlet_WAR_workspaceportlet_struts.portlet.action=%2Fworkspace%2Fdocuments%2Findex&_WorkspacePortlet_WAR_workspaceportlet_struts.portlet.mode=view&workspace.id=340042128&workspace.name=Embedded%20Systems%20Programming%20TX00EX76-3005&currentFolder=10036746

[7] Blog admin, EEPROM (Electrically Erasable Programmable Read-Only Memory) – Definition & Detailed Explanation – Hardware Glossary Terms, June 6, 2024. Assessed: 16 March 2025.

https://pcpartsgeek.com/eeprom-electrically-erasable-programmable-read-only-memory/

[8] Raspberry Pi Debug Prob. Assessed: 16 March 2025.

https://www.raspberrypi.com/documentation/microcontrollers/debug-probe.html

[9] Barr, M. (1999). *Programming Embedded Systems in C and C++.* O'Reilly Media.

[10] Booch, G. (1994). *Object-Oriented Analysis and Design with Applications.* Addison-Wesley

[11] Docker Documentation. *Docker Overview.* Docker, Inc. Assessed: 16 March 2025.