



Door**Sure**: The Garage Door Status Detector

University of Central Florida
Department of Electrical and Computer Engineering
Advised by Dr. Chung Yong Chan

Final Report
EEL 4914 Senior Design II
Spring - Summer 2025

Reviewers

Dr. Suboh Suboh
Dr. George Atia
Dr. Mark Llewellyn

Group 22

Mina Ghandour
Felix Liu
Khang Nguyen
Andrea Rosa

Electrical Engineering
Computer Engineering
Electrical Engineering
Computer Engineering

Table of Contents

Chapter 1. Executive Summary.....	1
Chapter 2. Project Description.....	2
2.1 Motivations.....	2
2.2 Existing Products.....	3
2.3 Goals and Objectives.....	5
2.4 System Description.....	7
2.5.1 System Specifications.....	7
Table 2.5.1: System Requirements.....	8
2.5.2 Component Specifications.....	9
Table 2.5.2: Component Specifications.....	9
2.5.3 House of Quality.....	9
Figure 2.5.3: House of Quality.....	10
2.6.1 Hardware Block Diagram.....	11
Figure 2.6.1: Hardware Block Diagram.....	11
2.6.2 System Flowchart Diagram.....	12
Figure 2.6.2.1: Primary Device System Flowchart.....	12
Figure 2.6.2.2: Secondary Device System Flowchart.....	13
Chapter 3. Research and Investigation.....	13
Hardware Comparison.....	14
3.1.1 MCU Technology Comparison.....	14
Table 3.1.1: Comparison between MCU and FPGA (IBM, 2024).....	14
3.1.2 MCU Part Selection.....	15
ESP8266.....	15
ESP32.....	15
ESP32-S3.....	15
BCM2835.....	15
RP2040.....	15
RP2350.....	16
Table 3.1.2: MCU Part Selection.....	16
3.2.1 Door Status Technology.....	17
Reed Switch Sensor.....	17
Hall Effect Sensor.....	17
Limit Switch Sensor.....	17
Ultrasonic Sensor.....	18
Infrared Sensor.....	18
LiDAR Sensor.....	18
Motion Sensor.....	19
Camera Object Detection Sensor.....	19

Table 3.2.1: Comparison of Door Detection Sensor.....	19
3.2.2 Door Status Parts Selection.....	20
MC-38.....	20
SR6-004.....	20
MKA-14103.....	20
Table 3.2.2: Reed Switch Parts Selection.....	20
3.3.1 Obstacle Detection Technology.....	21
Ultrasonic Sensor.....	21
Infrared Sensor.....	22
Lidar Sensor.....	23
Table 3.3.1 Obstacle Detection Technology.....	24
3.3.2 Obstacle Detection Selection.....	25
Ultrasonic Part Selection.....	25
HC-SR04.....	26
RCWL-1601.....	26
HY-SRF05.....	26
Table 3.3.2.1 Ultrasonic Part Selection.....	26
Camera development boards.....	26
ESP32-CAM.....	27
ESP-EYE V2.1.....	27
ESP-S3-EYE.....	27
T-Camera plus.....	28
AI Vision WonderCam.....	28
XIAO ESP32 S3 Sense.....	28
Table 3.3.2.2: Camera Development Board Comparison for the Prototype...	29
3.3.3 Camera modules.....	31
OV2640.....	31
OV3640.....	31
OV5640.....	31
OV7670.....	32
OV7725.....	32
NT99141.....	32
Table 3.3.3: Camera module.....	32
3.4.1 Temperature and Humidity Selection.....	33
DHT11.....	33
DHT22.....	33
SHT7.....	33
Table 3.4.1: Temperature and Humidity sensors.....	34
3.5.1 Motor Controller Technology.....	34
DC Controllers.....	34
AC Controllers.....	35

Servo Controllers.....	35
Stepper Controllers.....	35
Table 3.5.1: Comparison of Motor Controllers.....	35
3.5.2 Motor Controller Servo Selection.....	35
SG90.....	36
MG90S.....	36
MG90D.....	37
MG996R.....	37
RDS3225.....	37
Table 3.4.2: Servo Motor Comparison.....	38
3.6.1 Voltage Regulation Technology.....	38
Linear Regulators.....	39
Figure 3.6.1.1: Series Linear Voltage Regulator (Subedi, 2024).....	39
Figure 3.6.1.2: Shunt Linear Voltage Regulator (Subedi, 2024).....	40
Switching Regulators.....	40
Table 3.6.1: Switching Regulator part selection.....	40
3.6.2 Switching Regulator Parts Selection.....	40
LM2576.....	40
AP6320X.....	41
TI TPS54202.....	41
Table 3.6.2: Switching Regulators part selection.....	41
3.7.1 Batteries Technology.....	41
Table 3.7.1: Battery Technology Comparison.....	42
AA Batteries.....	43
AAA Batteries.....	43
9V Batteries.....	43
123 Batteries.....	43
CR2 Batteries.....	44
18650 Batteries.....	44
Li-Po Batteries.....	44
Table 3.7.2: Lithium Battery Size Comparison (Renogy, 2024).....	44
3.7.2 Li-Po Battery Selection.....	45
Table 3.7.2: Li-Po Battery Size Comparison.....	45
Communication Protocols.....	46
3.8.1 Wired Communication Protocols.....	46
UART.....	46
I2C.....	46
SPI.....	46
USB.....	46
Ethernet.....	47
Table 3.8.1: Wired Communication Protocols.....	47

3.8.2 Wireless Communication Protocols.....	47
Wi-Fi.....	47
Bluetooth.....	47
Table 3.8.2: Wired Communication Protocols.....	48
Software Comparison.....	48
3.9.1 Remote Networking Technology.....	48
Table 3.9.1: Remote Networking Comparison.....	49
3.10.1 Cloud Server Service Providers.....	49
Amazon Web Services (AWS).....	50
Google Cloud Platform.....	52
Microsoft Azure.....	53
Table 3.10.1: Cloud Server Provider Comparison.....	54
3.11.1 Database Types.....	55
Relational SQL.....	55
Non-Relational NoSQL.....	55
Table 3.11.1 Relational Vs Non-Relational DB.....	56
3.11.2 Database Implementation.....	57
Cloud Bigtable.....	57
Cloud Datastore.....	57
Firestore.....	57
Firebase Realtime Database (RTDB).....	58
Table 3.11.2 Google Cloud Platform Database Implementations.....	58
3.12.1 Platform Technology.....	59
Table 3.12.1 Platform Technology Comparison.....	60
3.13.1 Communication between Microcontrollers.....	61
Table 3.13.1: Local Communication Comparison.....	62
3.14.1 AI-Based Object Detection.....	63
Table 3.14.1: AI-Based Object Detection.....	64
3.15.1 Embedded Firmware.....	65
Table 3.15.1: Embedded Firmware.....	66
3.16.1 Notification System.....	66
Chapter 4. Standards and Design Constraints.....	66
4.1 Standards.....	67
4.1.1 PCB Design.....	67
Figure 4.1.1 Hierarchy of IPC Design specification (2220 series).....	68
4.1.2 Wi-Fi.....	69
Figure 4.1.2 Evolution of IEEE Standards For WI-FI Technology.....	70
4.1.3 Bluetooth.....	70
4.2 Design Constraints.....	71
4.2.1 Time.....	71
4.2.2 Economic Constraint.....	73

4.2.3 Safety Constraint.....	74
4.2.4 User friendly Constraint.....	75
4.2.5 Manufacturability Constraint.....	76
4.2.6 Sustainability Constraint.....	77
Chapter 5. Comparison of ChatGPT with other Similar Platforms.....	78
Case Study 1.....	78
Case Study 2.....	79
Case Study 3.....	80
Case Study 4.....	80
Chapter 6. Hardware Design.....	82
6.1 Subsystems.....	82
Garage Status System.....	82
Figure 6.1.1: Garage Status Subsystem.....	82
Object Detection System.....	83
Figure 6.1.2: Object Detection Subsystem.....	83
Camera Feed System.....	84
Figure 6.1.3: Camera Feed Subsystem.....	84
Temperature and Humidity System.....	85
Figure 6.1.4: Temperature Subsystem.....	85
Buzzer System.....	86
Figure 6.1.6: Buzzer Subsystem.....	86
Servo Motor System.....	87
Figure 6.1.7: Servo Motor Subsystem.....	87
Communication System.....	88
Figure 6.1.8: Communication Subsystem.....	88
Power Management System.....	89
Figure 6.1.9: Power Subsystem (general).....	89
Battery Choice.....	90
Power Regulation.....	90
Table 6.1.10 Switching Regulators.....	91
6.2 Power Distribution.....	91
Table 6.2.1: Primary Device Component Power Specifications.....	91
Figure 6.2.2: Primary Device Component Power Formula.....	92
Table 6.2.3: Primary Device Component Power Specifications.....	92
Figure 6.2.4: Primary Device Component Power Formula.....	93
6.3 USB to UART bridge.....	93
6.4 JTAG and UART.....	93
Table 6.3.1: UART Connections.....	94
Table 6.4.2: JTAG Connections.....	94
6.5 Schematic Design.....	95
Primary Device Schematic.....	95

Figure 6.4.1: Primary device Schematic.....	96
Secondary Device Schematic.....	96
Figure 6.5.2: Secondary Device Schematic.....	98
USB to UART converter Schematic.....	98
Temperature and Humidity sensor Schematic.....	99
Figure 6.5.4: Temperature and Humidity sensor Schematic.....	99
Power Systems Schematic.....	99
Figure 6.5.5: 3.3V regulator Schematic.....	100
Figure 6.5.6: 5V regulator Schematic.....	100
Chapter 7. Software Design.....	101
7.1 Use Case Diagram.....	101
Figure 7.1: Use Case Diagram.....	102
7.2 Entity-Relationship Diagram.....	102
Figure 7.2: Entity-Relationship Diagram.....	103
7.3 Class Diagram.....	103
Figure 7.3: Class Diagram.....	105
7.4 State Diagram.....	105
Figure 7.4: State Diagram.....	107
7.5 Sequence Diagram.....	108
Figure 7.5: Sequence Diagram.....	109
7.6 User Interface.....	109
Figure 7.6.1 Prototype Login Page (left), Final Login Page (right).....	110
Figure 7.6.2 Prototype Settings Page (left), Final Settings page (right). ..	111
Figure 7.6.3 Prototype Home Page (left), Final Home Page (right).....	112
Figure 7.6.4 Prototype Live Feed Page (left), Final Live Feed Page (right). ..	113
7.7 Development Environments and Tools.....	113
7.7.1 Programming Languages.....	113
7.7.2 IDEs.....	113
7.7.3 Libraries and Dependencies.....	114
7.7.4 Web Application Deployment and GitHub.....	115
Chapter 8. System Fabrication/ Prototype Construction.....	116
8.1 Prototype Construction.....	116
Figure 8.1.1 Primary System Initial Prototype.....	116
Figure 8.1.2 Secondary System Initial Prototype.....	117
Figure 8.1.3 Object Detection System Initial Prototype.....	117
Figure 8.1.4 Primary Device Final Prototype.....	118
Figure 8.1.4 Secondary Device Final Prototype.....	119
8.2 PCB Layout.....	119
8.2.1 Primary board PCB.....	120
Figure 8.2.1 Primary board PCB layout.....	121

8.2.2 Secondary board PCB.....	121
Figure 8.2.2 Servo system PCB layout.....	122
8.2.3 Voltage Regulator PCB (3.3V and 5V).....	122
Figure 8.2.3.1 3.3V Voltage Regulator PCB layout.....	123
Figure 8.2.3.2 5V Voltage Regulator PCB layout.....	124
8.2.4 Temperature and Humidity sensor.....	124
Figure 8.2.4 Temperature and humidity sensor PCB layout.....	125
8.2.5 USB to UART converter.....	125
Figure 8.2.5 USB to UART converter PCB layout.....	126
8.3 Housing Design.....	126
Figure 8.3.1 Front view of Primary Device with Housing.....	127
Figure 8.3.2 Side View of Primary Device with Housing.....	128
Figure 8.3.3 Mounting of the Primary Device with Housing.....	128
Figure 8.3.4 Front view of Secondary Device with Housing.....	129
Figure 8.3.5 Side view of Secondary Device with Housing.....	130
Figure 8.3.5 Mounting of the Secondary Device with Housing.....	130
Chapter 9. System Testing and Evaluation.....	130
9.1 Alter State of Garage Testing.....	130
9.2 Object Detection Testing.....	131
9.3 Livestream Testing.....	132
9.4 Object Identifier Testing.....	133
9.5 Power Testing.....	134
9.6 Senior Design II Final Testing.....	135
Chapter 10. Administrative Content.....	136
10.1 Budget.....	136
10.2 Bill of materials.....	136
Table 10.2: Bill of Materials.....	136
10.3 Work Distribution.....	137
Table 10.3: Work Distribution.....	138
10.4 Milestones.....	139
Table 10.4.1: Milestones for Senior Design I.....	140
Table 10.4.2: Milestones for Senior Design II.....	140
Chapter 11. Conclusion.....	141
APPENDIX A - References.....	143
General Statistics and Market Research.....	143
Technical Articles & Cloud Services.....	143
Microcontroller & Hardware Datasheets.....	143
Technical Education & Sensor References.....	144
APPENDIX B - Parts.....	145
APPENDIX C - Data sheet.....	149
APPENDIX D - LLM Prompts.....	155

Key Considerations:..... 157

Battery Options:..... 157

Best Choice for Low Power & Long Battery Life:..... 158

Tables

Table 2.5.1: System Requirements.....	8
Table 2.5.2: Component Specifications.....	9
Table 3.1.1: Comparison between MCU and FPGA (IBM, 2024).....	14
Table 3.1.2: MCU Part Selection.....	16
Table 3.2.1: Comparison of Door Detection Sensor.....	19
Table 3.3.1 Obstacle Detection Technology.....	24
Table 3.3.2.1 Ultrasonic Part Selection.....	26
Table 3.3.2.2: Camera Development Board Comparison for the Prototype...	29
Table 3.3.3: Camera module.....	32
Table 3.4.1: Temperature and Humidity sensors.....	34
Table 3.5.1: Comparison of Motor Controllers.....	35
Table 3.4.2: Servo Motor Comparison.....	38
Table 3.6.1: Switching Regulator part selection.....	40
Table 3.6.2: Switching Regulators part selection.....	41
Table 3.7.1: Battery Technology Comparison.....	42
Table 3.7.2: Lithium Battery Size Comparison (Renogy, 2024).....	44
Table 3.7.2: Li-Po Battery Size Comparison.....	45
Table 3.8.1: Wired Communication Protocols.....	47
Table 3.8.2: Wired Communication Protocols.....	48
Table 3.9.1: Remote Networking Comparison.....	49
Table 3.10.1: Cloud Server Provider Comparison.....	54
Table 3.11.1 Relational Vs Non-Relational DB.....	56
Table 3.11.2 Google Cloud Platform Database Implementations.....	58
Table 3.12.1 Platform Technology Comparison.....	60
Table 3.13.1: Local Communication Comparison.....	62
Table 3.14.1: AI-Based Object Detection.....	64
Table 3.15.1: Embedded Firmware.....	66
Table 6.1.10 Switching Regulators.....	91
Table 6.3.1: UART Connections.....	94
Table 6.4.2: JTAG Connections.....	94
Table 10.2: Bill of Materials.....	136
Table 10.3: Work Distribution.....	138
Table 10.4.1: Milestones for Senior Design I.....	140
Table 10.4.2: Milestones for Senior Design II.....	140

Figures

Figure 2.5.3: House of Quality.....	10
Figure 2.6.1: Hardware Block Diagram.....	11
Figure 2.6.2.1: Primary Device System Flowchart.....	12
Figure 2.6.2.2: Secondary Device System Flowchart.....	13
Table 3.2.2: Reed Switch Parts Selection.....	20
Figure 3.6.1.1: Series Linear Voltage Regulator (Subedi, 2024).....	39
Figure 3.6.1.2: Shunt Linear Voltage Regulator (Subedi, 2024).....	40
Figure 4.1.1 Hierarchy of IPC Design specification (2220 series).....	68
Figure 4.1.2 Evolution of IEEE Standards For WI-FI Technology.....	70
Figure 6.1.1: Garage Status Subsystem.....	82
Figure 6.1.2: Object Detection Subsystem.....	83
Figure 6.1.3: Camera Feed Subsystem.....	84
Figure 6.1.4: Temperature Subsystem.....	85
Figure 6.1.7: Servo Motor Subsystem.....	87
Figure 6.1.8: Communication Subsystem.....	88
Figure 6.1.9: Power Subsystem (general).....	89
Table 6.2.1: Primary Device Component Power Specifications.....	91
Figure 6.2.2: Primary Device Component Power Formula.....	92
Table 6.2.3: Primary Device Component Power Specifications.....	92
Figure 6.2.4: Primary Device Component Power Formula.....	93
Figure 6.4.1: Primary device Schematic.....	96
Figure 6.5.2: Secondary Device Schematic.....	98
Figure 6.5.4: Temperature and Humidity sensor Schematic.....	99
Figure 6.5.5: 3.3V regulator Schematic.....	100
Figure 6.5.6: 5V regulator Schematic.....	100
Figure 7.1: Use Case Diagram.....	102
Figure 7.2: Entity-Relationship Diagram.....	103
Figure 7.3: Class Diagram.....	105
Figure 7.4: State Diagram.....	107
Figure 7.5: Sequence Diagram.....	109
Figure 7.6.1 Prototype Login Page (left), Final Login Page (right).....	110
Figure 7.6.2 Prototype Settings Page (left), Final Settings page (right).	111
Figure 7.6.3 Prototype Home Page (left), Final Home Page (right).....	112
Figure 7.6.4 Prototype Live Feed Page (left), Final Live Feed Page (right).	113
Figure 8.1.1 Primary System Initial Prototype.....	116
Figure 8.1.2 Secondary System Initial Prototype.....	117
Figure 8.1.3 Object Detection System Initial Prototype.....	117
Figure 8.1.4 Primary Device Final Prototype.....	118

Figure 8.1.4 Secondary Device Final Prototype.....	119
Figure 8.2.1 Primary board PCB layout.....	121
Figure 8.2.2 Servo system PCB layout.....	122
Figure 8.2.3.1 3.3V Voltage Regulator PCB layout.....	123
Figure 8.2.3.2 5V Voltage Regulator PCB layout.....	124
Figure 8.2.4 Temperature and humidity sensor PCB layout.....	125
Figure 8.2.5 USB to UART converter PCB layout.....	126
Figure 8.3.1 Front view of Primary Device with Housing.....	127
Figure 8.3.2 Side View of Primary Device with Housing.....	128
Figure 8.3.3 Mounting of the Primary Device with Housing.....	128
Figure 8.3.4 Front view of Secondary Device with Housing.....	129
Figure 8.3.5 Side view of Secondary Device with Housing.....	130
Figure 8.3.5 Mounting of the Secondary Device with Housing.....	130

Chapter 1. Executive Summary

DoorSure is a smart garage door monitoring and control system designed to enhance home security, improve user convenience, and provide peace of mind. Developed as a capstone project by Group 22 at the University of Central Florida, DoorSure addresses a common homeowner concern: forgetting to close the garage door, a mistake that can lead to security risks and potential property damage. The system empowers users with real-time awareness and remote control over their garage door, anytime and anywhere.

Designed with universal compatibility and cost-efficiency in mind, DoorSure stands apart from existing systems that are often expensive or brand-specific. It supports nearly all standard garage doors without requiring significant hardware modifications, combining key features of status detection, object awareness, live streaming, and remote operation into a unified, affordable solution.

DoorSure utilizes an IoT-based approach, integrating sensors, a servo motor, and microcontroller units to detect and control the garage door's position. Users interact with the system through a custom application, which supports live camera feeds and real-time notifications. Safety is further enhanced through object detection, which prevents door closure if an obstruction is detected. As a stretch goal, the system also explores AI-based object classification to identify common garage items that are causing an obstruction.

This report provides a comprehensive overview of the DoorSure project. It begins with the project motivation, objectives, and a high-level system description. It then explores existing market solutions and highlights DoorSure's improvements in affordability, functionality, and flexibility. The research and investigation chapter outlines the rationale behind component and technology selections. Later sections present the hardware and software design, including system schematics, communication protocols, and user interface development. Additional chapters address design constraints, engineering standards, and system fabrication. The prototype construction and testing chapter discusses real world evaluation, followed by a summary of the project budget, materials, team milestones, and plans for Senior Design II.

From component integration to software design and power management, DoorSure is built with a strong emphasis on safety, user-friendliness, and universalness. It is a practical IoT solution tailored for any home with a garage. Ultimately, DoorSure is about more than automation; it's about giving homeowners confidence in their security, no matter where they are.

Chapter 2. Project Description

In this section of the report, our team properly introduces our Senior Design project idea. This section discusses project motivations, goals and objectives, and proposed system descriptions. This section further analyzes our proposal by discussing engineering and marketing requirements and specifications, as well as detailing the hardware and software aspects for our proposed device.

2.1 Motivations

Imagine this: you are rushing out of your home, running late for work. In the chaos of the morning— getting dressed, making breakfast, grabbing your keys— you pull out of the driveway without a second thought. But by the time you make the drive and arrive at your workplace, a nagging thought crosses your mind— did you remember to close the garage door? In today's fast-paced world, this scenario has more than likely been a reality for the average person. Preoccupied with more pressing tasks, it becomes easy for one to forget something as important as closing the garage door before leaving home.

Leaving your garage door open is not just an inconvenience, it poses a serious security risk. An open garage door provides easy access for burglars, putting your home, valuables, and personal safety at risk. According to the 2019 FBI UCR: Crime in the United States, approximately 6.9 million property crimes were reported. Of these property crimes, 1.1 million were burglaries, resulting in losses of an estimated \$3 billion in stolen valuables and direct property damage. More recently, 2024 reports from Securiteam indicate that burglary remains a major concern, with approximately 2.5 million burglaries occurring in the United States, resulting in an estimated \$3.4 billion in losses. However, advancements in smart security technology and neighborhood surveillance systems have contributed to an overall declining trend in burglary rates over the years. Enhanced security measures— such as smart home devices, motion-activated cameras, and automated alerts—are making it harder for intruders to go undetected, reducing the likelihood of break-ins.

Beyond theft, however, an open door exposes your garage to the elements - rain, wind, or even wandering animals that could cause property damage. If you're away for hours, or even the entire day, these risks only increase. In response to this problem, modern technology can be used to create smart home solutions that provide not only property security but also peace of mind. As engineers, we created a solution by integrating sensors and real-time monitoring systems with software technology that allowed homeowners to remotely check the status of and receive alerts about their garage door. This system also gave

users the ability to interact with their garage door, ensuring their home remained secure even when they were miles away.

This project served as our first real immersion into the world of engineering design, allowing us to take an idea from concept to complete prototype. Unlike structured classroom labs, where solutions are often known before experimentation begins, this project required us to identify and design a device that would resolve a real-world issue, research and iterate on our designs to create a functional and reliable system, and integrate a software application that would directly interact with the hardware. We gained hands-on experience with key engineering principles such as circuit design, sensor integration, and software development, all while addressing a real-world problem that affects millions of people. This project was not just about building a functioning product—it was about developing and applying the concepts and technical skills we had learned throughout our academic career here at University of Central Florida. By combining hardware and software components, we explored the intersection of electrical and computer engineering as a team, preparing us for more complex projects and team dynamics in our future careers. Ultimately, this project was an opportunity to gain valuable hands-on design experience while working with teammates of differing levels of skill sets to achieve a common goal: to transform an everyday inconvenience into an innovative, user-friendly solution that enhances security and convenience for homeowners everywhere.

2.2 Existing Products

In our research, our team noticed existing garage door status detectors tended to be directly linked to their corresponding garage door brand. One of the brands found in our research, the Chamberlain MyQ Smart Garage Door Sensor, is cost-effective but is only compatible with Chamberlain and LiftMaster garage doors. Moreover, Chamberlain offers an indoor security camera with motion detection that connects to their corresponding application. However, these two devices are sold separately.

Expanding on this idea, our device combined both features of a garage door status detector and a security camera, while also offering open/close door features via a servo motor with an MCU and having a live streaming camera module that could be placed anywhere the client wanted and connected seamlessly together with the web application.

Chamberlain works with most garage door brands that use photoelectric sensors that do not shut off and it is not intended for use with openers in which the photoelectric sensors are located near the bottom of the garage door, all

these requirements for the device to work properly which is a major inconvenience to customers.

Our project aimed to combine the garage door status detector with the camera while aiming for a lower cost compared to what Chamberlain offers; for approximately 60 dollars combining both devices into one for our project has lower costs. In addition, our project aims to offer full compatibility with garage doors regardless of the year of manufacture, offering a peace of mind when buying our device.

Another device that has similar properties to our proposed device is the Genie's Aladdin Connect. The Aladdin connect can enable the use of smartphones to open and close any garage door after 1993, but this is a costly item, being over \$50 to purchase a set of 2 garage control keys. Although this product offers control keys, it does not offer the camera add-on, which would be a safety concern as there wouldn't be a live feed of the garage. Our proposed device offers the live feed of the camera viewable at any and all times on the accompanying application.

An honorable inspiration is SwitchBot Smart Switch Button Pusher, an automated device designed to press physical buttons and switches. It attaches to the surface of the button, and a small arm presses the button when activated. The device primarily operates via Bluetooth, and the mobile app is available as well. The remote operation is achieved by pairing with the SwitchBot Hub, which connects to Wi-Fi. The SwitchBot is a popular choice for light switches, coffee makers, and other appliances, and this application makes it an easy solution for smart home devices without modifying existing hardware.

Unlike the SwitchBot, our garage status detecting device was specifically designed for garage doors, featuring real-time monitoring of the door's open/close state, which SwitchBot lacked. Also, while SwitchBot relied on Bluetooth or hub control, our solution operated on Wi-Fi and Bluetooth, accessing without the need for additional hardware. Another key advantage of our project was object awareness, preventing the garage door from closing if an object was detected in the path of the garage door while open. Moreover, our device used a replaceable battery instead of a rechargeable battery, which reduced downtime with frequent recharging. DoorSure ensured excellent compatibility and reliability concerning garage doors and home security compared to the more generalized approach of SwitchBot.

Our application aimed to expand on the existing functions of the Chamberlain's device and the Genie's Aladdin, as well as incorporating the SwitchBot's actions

to create a universally compatible device with existing garage doors, camera integration showing a live feed within our application, object detection and identification, and real-time notification updates. While recognizing that each product had their strengths and weaknesses—for example, Chamberlain’s product offered the garage door status detector with a camera but with a trade-off on cost and compatibility with all garage doors, and the Genie’s Aladdin Connect offered control keys with mobile application connectivity but with a trade-off of high cost and no camera integration—we aimed to find a happy medium between cost and extra components such as camera integration. Using these devices as a baseline, our proposed device expanded on these existing ideas to create a new cost-effective, user-friendly, and efficient device that combined different aspects of the existing products into one clean-cut alternative specific to garage doors.

2.3 Goals and Objectives

Basic Goals:

1. The user has constant and accurate access to their garage door’s current status.
2. The user is able to alter the status of the garage door regardless of their location.
3. The user is notified if the garage door status is altered from its current state based on preferences.
4. The system blocks a status change if there is an object in the way of the garage door.

Advance Goals:

1. The user can view live camera feed from the primary device on the web application.

Stretch Goals:

1. The device utilizes AI to accurately identify objects that are causing any potential blockage to the garage sensors, therefore not allowing the garage door status to be changed.

Objectives:

To ensure constant and accurate access to the garage door’s status, our device utilized a microcontroller to process sensor inputs and manage communications. A reed switch was used to detect whether the garage door was fully open or fully closed, while a sensor component measured the garage door’s position in real time. This improved our device’s accuracy rate concerning the real-time

status of the door. Furthermore, the microcontroller's wireless capabilities transmitted collected status data to a database, allowing the user access to the data through our secure application at any and all times.

In order to enable remote control functionality, our application sends information to and from the user's device via a remote networking service. After testing with bluetooth connection, our team finalized the design to have both individual devices connect to the Wi-Fi to improve latency times. Security features such as encrypted data transmission and user authentication were implemented to prevent unauthorized access to the data of the garage door. Each web page redirect first checks if the user is logged in to ensure this security feature.

The system notifies users whenever the garage door status changes, according to preferred notification configurations. The reed switch and object detecting sensor send updates of the door status to the database, triggering real-time notifications whenever movement changes are detected. Notifications are sent as emails.

To ensure safety and property protection, the system prevents the garage door from operating if an obstruction is detected. The object detection sensor continuously scans for objects in the path of the garage door when the garage door status is open. If an object is found to be in the path of the door, a function call disables the servo motor from operating.

The camera component is used to capture images to simulate a live stream at a press of a button on the live feed page. The camera communicates with the primary device's MCU, allowing the user access to this footage at all times through the application. This allows users to view a live video feed of their garage at any time through the web application.

As a stretch goal, the system incorporates AI-based object recognition to identify the objects that are preventing the garage door from closing. The camera component captures an image from the press of a button which is then fed into an AI model. This AI then analyzes the image and updates the corresponding area on the application with the objects in the image.

Our device connects directly to the user's home Wi-Fi network and utilizes wireless communication. The system further requires user authentication to increase security. It is important to add this level of security to not allow just anyone with access to the Wi-Fi control of the garage. This ensures that if an unauthorized user connects to the home Wi-Fi, they will not simply be able to enter the application and open the user's garage door.

2.4 System Description

The hardware components for our device consist of the following bullet points. Please note that specific component selections are further explored in Chapter 3 of this report.

- Microcontrollers
- Wireless Modules
- Camera Module
- Sensor
 - Reed Switch
 - Ultrasonic sensor
- Power Source
 - Rechargeable battery
- Power LED Indicator
- Enclosure

The software aspect of this project consists of an application, designed for direct user interactivity with the hardware. From the accompanying application, users are able to:

- Receive garage status notifications:
 - The garage door has been opened
 - The garage door has been closed
 - Garage door left open after x amount of time elapsed
- Directly interact with the garage:
 - Fully close the garage door
 - Fully open garage door
- View live camera feed
- View AI identification of object causing an obstruction

2.5.1 System Specifications

The system specifications indicate what the smart garage door system must achieve to meet customer needs and project goals. They ensure the system can detect, process, and communicate the user's garage door status in various environmental and operational conditions. We have tested the highlighted requirements in our live demo and determined our proposed device is successful as these needs have been met.

Table 2.5.1: System Requirements

Parameter	Target Value	Description
Garage Door Status Detection	$\geq 95\%$ Accuracy	Ensures sensor precision for detecting door position.
Servo Movement Delay	≤ 2 seconds	Ensures quick responsiveness to user input
Obstacle Detection accuracy	$\geq 90\%$ Accuracy	System blocks close door action if an object is obstructing the path
Obstacle object identification accuracy	$\geq 75\%$ Accuracy	Identify the object in the path of the garage door
Door Opening time duration	≤ 20 seconds	Ensures the garage door opens within a reasonable timeframe for user convenience while maintaining smooth and safe operation.
Door Closure time duration	≤ 20 seconds	Ensures the garage door closes efficiently without unnecessary delays, enhancing security and usability.
Operating Voltage	3.3V - 5V	Ensures compatibility with embedded systems.
Overall Power Consumption	~2W - 6W	Total energy usage of the system
Unit Cost	\$40-\$70	Affordable, competitive cost
Battery Life	≥ 3 months	Extends operation without frequent replacements.
Installation Time	≤ 20 minutes	Ensures a user-friendly setup.

Parameter	Target Value	Description
Operating Temp	-20°C to 60°C	Ensures reliable operation in various environments.
Humidity Tolerance	10% - 90% RH	Prevents failures due to humidity exposure.

2.5.2 Component Specifications

The component specifications define the specific hardware modules and capabilities, explaining the role of each component in the overall system. These specifications ensure the smart garage system is compatible, reliable, and meets the technical demands of the project.

Table 2.5.2: Component Specifications

Component	Parameter	Value
Microcontroller	Clock rate	80 MHZ - 240 MHZ
Camera Module	Resolution	At least 720P
Power Source	Battery life	Last 3 month
Wi-Fi Bands	Bandwidth	2.4GHz, 5GHz
Wi-Fi Range	Coverage	≥ 30 meters indoors
Bluetooth Range	Coverage	≥ 10 meters indoors

2.5.3 House of Quality

The House of Quality diagram showcases our component and system specifications in terms of engineering and marketing requirements. Through the ABET lectures, our team selected specifications from the previous tables 2.5.1 and 2.5.2 to detail the most important specifications that will measure how successful our project is at the end of our Senior Design.

The House of Quality is constructed by first getting and prioritizing customer requirements such as the compatibility with all garage doors and then we would translate those requirements to corresponding engineering requirements which would be more technical related to the project, it can be seen as the way to meet those customer requirements, additionally, we can set product target related to the engineering requirement so that we can actually have a numerical target to measure how successful our project is.

The customer requirements and engineering requirements are the sets of data that define the house of quality and our project in general, we define the relationship between each customer requirement and each engineering requirement based on their correlation and whether it is positive or negative using the legend in the upper left corner based on the direction of desirability of each of the requirements. Finally the roof shape defines the correlation between the engineering requirements themselves.

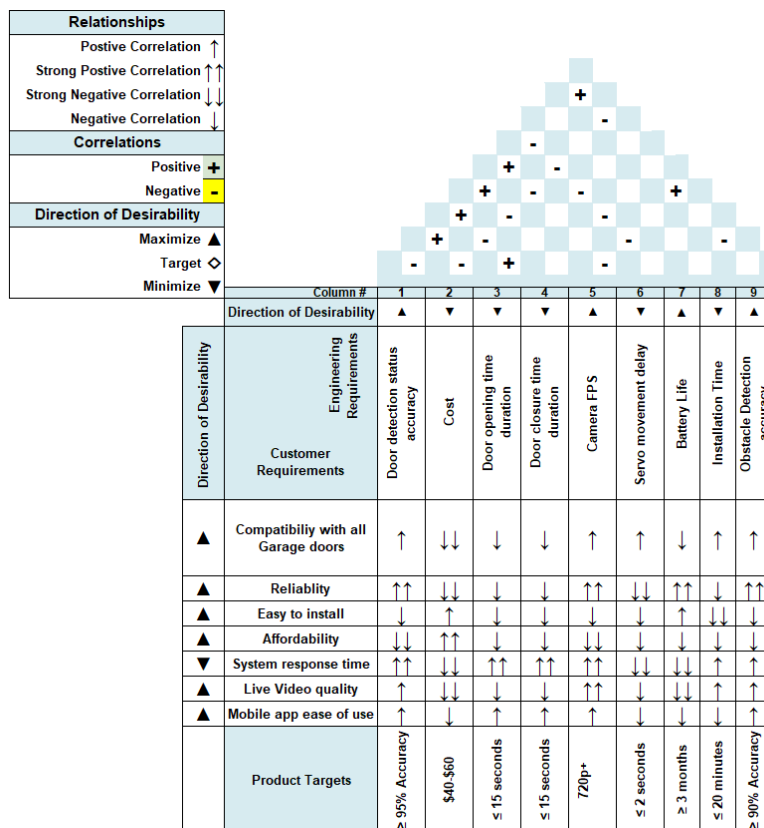


Figure 2.5.3: House of Quality

2.6.1 Hardware Block Diagram

Our device hardware required several peripherals to meet the goals of a successful PCB fabrication in our course. Following our proposed system specifications table, in terms of operating temperature and humidity, our hardware includes a built-in temperature sensor. This helped monitor the device in the garage, especially in Florida, where humidity and temperature were constantly fluctuating and often present in high concentrations. Our device also featured a buzzer system which produces a buzzing pattern based on the target status of the door.

Note that our completed hardware system is split into two devices, the primary device containing the sensors for garage status and object detection, and the secondary device with the servo housing, buzzer component, and temperature and humidity sensor. The specifics are detailed below accordingly:

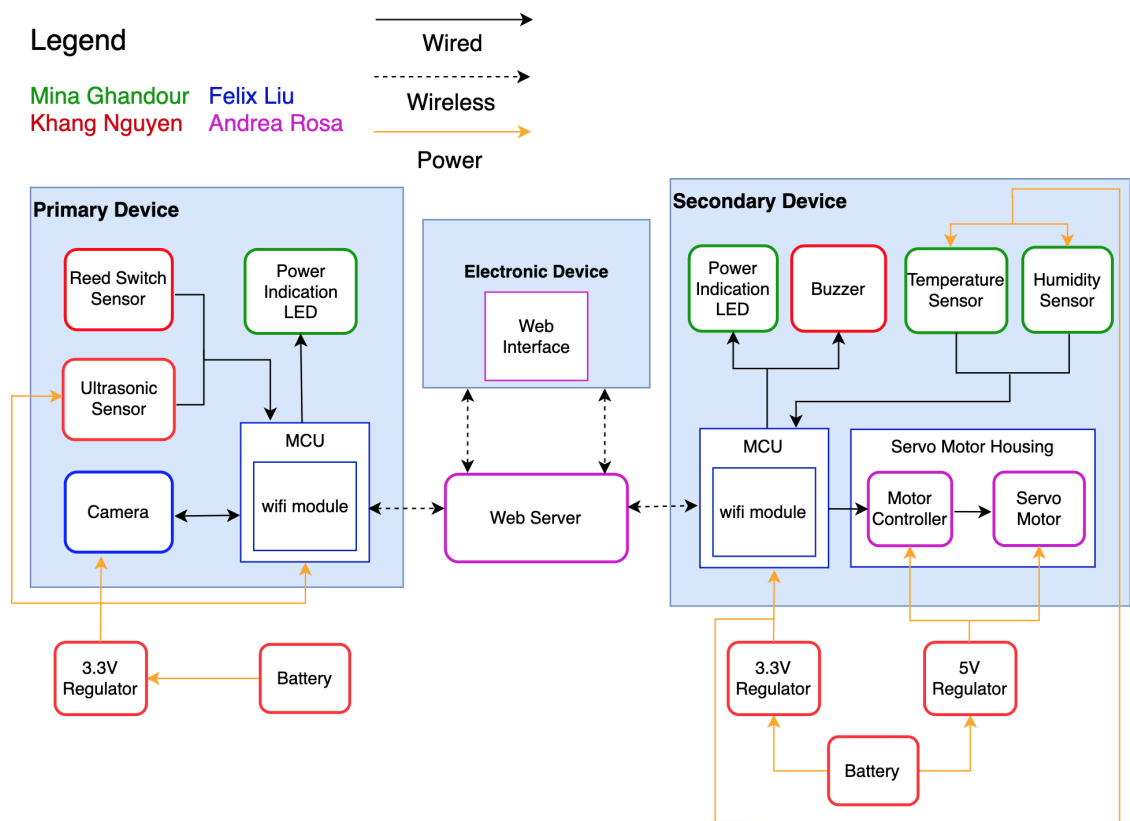


Figure 2.6.1: Hardware Block Diagram

2.6.2 System Flowchart Diagram

The system flowchart diagrams showcase the software flow. The software is unique to each device based on their corresponding main function. The primary device's main function is to detect object obstructions, detect door position status, and stream camera feed. The software diagram is shown in Figure 2.6.2.1 below.

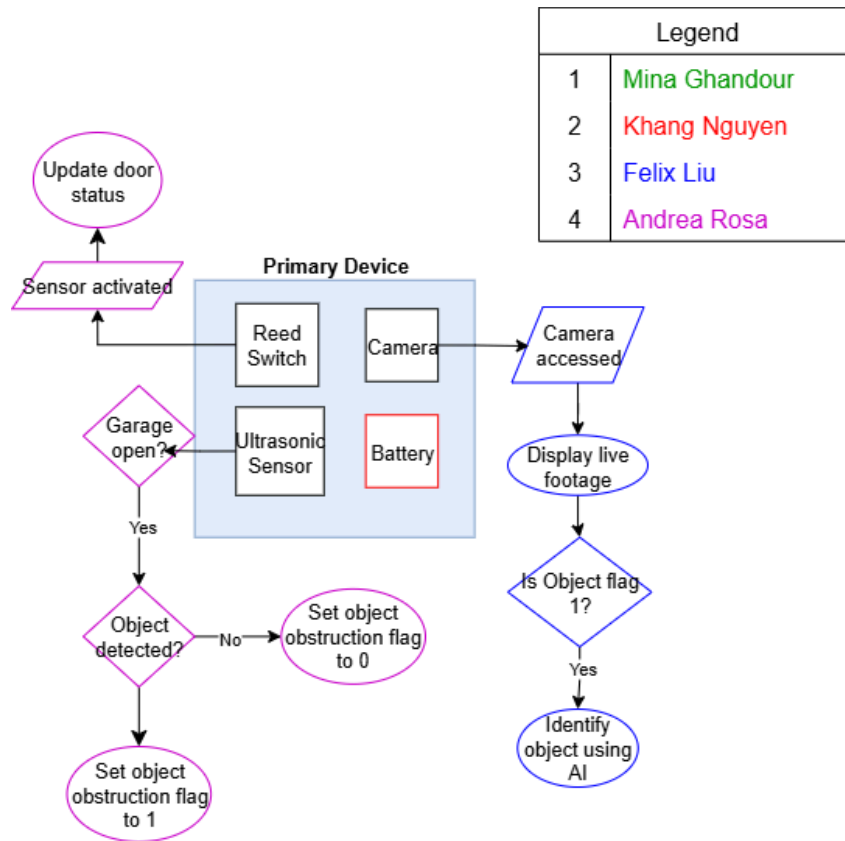


Figure 2.6.2.1: Primary Device System Flowchart

The software flowchart in Figure 2.6.2.2 showcases the main software functions of the secondary device. The secondary device is responsible for the actual movement of the servo, which presses the garage door button. The corresponding buzzer component sounds a pattern based on the target status of the door position. A temperature and humidity sensor also reads the garage environment to ensure the system is within operating conditions and to provide users with environmental conditions of their garage right on the application.

The buzzer component is called by the servo code according to the target door position. For example, if the servo is making a movement to “open” the garage

door, then the buzzer will sound out a two beep pattern. If the servo is making a movement to “close” the garage door, then the buzzer will sound out a four beep pattern. This pattern buzzing helps to recognize the desired movement of the servo during debugging and also alerts the user that an action has been sent to and received by the servo motor.

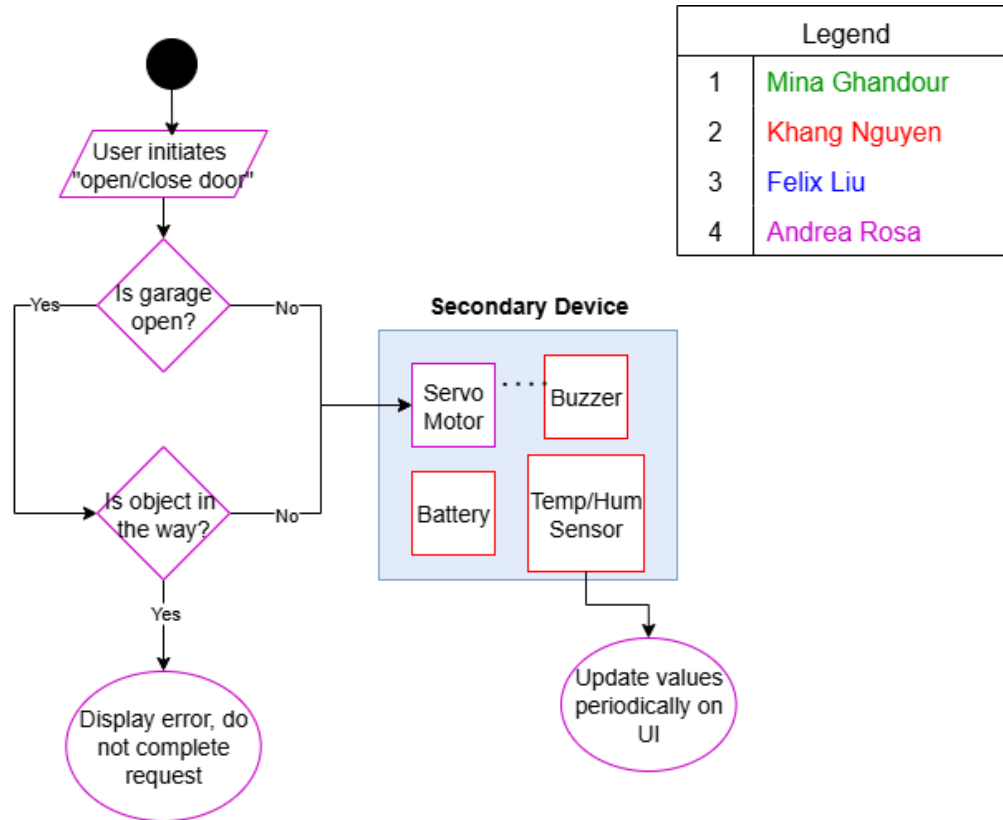


Figure 2.6.2.2: Secondary Device System Flowchart

Chapter 3. Research and Investigation

Extensive research was one of the most important aspects of project design. As discussed in Senior Design I ABET lectures, it was important for a project to be relevantly researched to avoid failure. In this section of the report, our team delved into different technological aspects of our project proposal, including both hardware and software comparisons. These comparisons were absolutely essential to our designing process as they not only allowed the team to make the most informed decisions on hardware parts and software technology that would minimize overall costs and communication, but also maximize efficiency, performance, and reliability. By thoroughly evaluating different hardware and software options, our team aimed to select the most optimal solutions.

For simplicity's sake, our selections are highlighted on the table of comparisons throughout this chapter.

Hardware Comparison

In this section of the report, we compared differing hardware parts and technological components that were needed to build our device. We explored our options with microcontrollers, batteries, voltage regulators, sensors, cameras, motor controllers, and servo motors. We took into account their cost, power consumptions, and overall capabilities to ensure our device was as cost and software effective as possible to achieve our outlined engineering requirements.

3.1.1 MCU Technology Comparison

For our garage detector, we required the use of a microcontroller unit (MCU) as the devices themselves did not require any high complexity compared to using a field-programmable grid array. As our devices were battery powered, using the microcontrollers was advantageous due to their low power consumption. Programming for MCUs utilized simple software development languages like C++, Python, and other low and high-level programming languages, allowing the time to configure each device to be manageable. (IBM) For the choices to compare which MCU we could potentially use, we decided to pick components that had a development kit with the respective MCU integrated already. This allowed the process of prototyping the devices to be easy to configure and troubleshoot.

Table 3.1.1: Comparison between MCU and FPGA (IBM, 2024)

	MCU	FPGA
Hardware Structure	Highly Modular	Fixed
Processing	Parallel	Sequential
Power Consumption	Optimized for low power	High power consumption
Programming	Software Language (C++, Python, Java)	Hardware Language (HDL, Verilog)
Cost	Expensive	Cheap

Versatility	Very flexible; customization on hardware level	Applicable to broad range of applications; not as flexible compared to FPGA
-------------	--	--

3.1.2 MCU Part Selection

ESP8266

The ESP8266 is a low-cost microcontroller that integrates Wi-Fi capabilities and a large peripheral interface that provide many external connections for our devices. The MCU contains a 32-bit CPU processor that can run on 80 to 120 MHz depending on clock speed needed.

ESP32

The successor to the ESP8266, the ESP32 is a dual core MCU containing Wi-Fi and Bluetooth features, unlike the ESP8266. All of the features of the 8266 are the same in the ESP32, but contain upgraded specifications such as having a maximum of 240 MHz clock speed and increased memory of 520 KB SRAM. These improved features still make this chip low-cost to purchase.

ESP32-S3

The ESP32-S3 contains much of the same hardware as the ESP32 but has improved hardware capabilities such as increased PSRAM and flash memory. Although the MCU has a slightly higher cost compared to the ESP32, this is necessary to supply the appropriate processing power for devices like a camera.

BCM2835

The BCM2835 processors can be found in the Raspberry Pi Zero series of microcontrollers. Due to its configuration as an ARM processor, the BCM2835 achieves good performance at a low cost.

RP2040

The RP2040 is an ARM processor that is mainly used in the Raspberry Pi Pico series of microcontrollers. The Pico W microcontroller allows usability of Wifi and Bluetooth features and performance is comparable to the ESP32 and 8266 processor chips.

RP2350

The RP2350 is an dual-core ARM processor that is mainly used in the Raspberry Pi Pico 2 series of microcontrollers. This is an improved version of the RP2040, giving higher performance due to the increased memory limitations.

Table 3.1.2: MCU Part Selection

MCU and Devkit	ESP8266 (NodeMCU)	ESP32 (ESP32 WROOM)	ESP32-S3 (ESP32-S3 WROOM)	BCM2835 (Raspberry Pi Zero W)	RP2040 (Raspberry Pi Pico W)	RP2350 (Raspberry Pi Pico 2 W)
Price	\$8	\$9	\$15	\$15	\$4	\$5
Wi-Fi	Yes	Yes	Yes	Yes	Yes	Yes
Bluetooth	No	Yes	Yes	Yes	Yes	Yes
Wireless connectivity built in MCU?	Yes	Yes	Yes	No	No	No
Clock Speed	160 MHz	240 MHz	240 MHz	1 GHz	133 MHz	150 MHz
Flash Memory	4 MB	4 MB	8MB	None	2 MB	4 MB
RAM	64 KB	520 KB	520KB	512 KB	264 KB	520 KB
Operating Voltage (V)	3.6V	3.6V	3.6V	5V	5V	5V
Power consumption(Peak Active)(W)	0.23	1.65	1.65	2.5	1.5	0.5

3.2.1 Door Status Technology

Door position detection provides critical information about the status of doors in different environments. In several applications, these sensors detected whether a door was open, closed, or partially open. Our focus was on achieving excellent accuracy, low power consumption, and strong reliability. The goal was to build a door detection system that would work efficiently and dependably in a home setting.

As the device was required to detect the status of the garage door and identify any interferences, sensors were needed for the system to be fully operational. The devices required the implementation of multiple sensors, so it was important to evaluate which sensors would be feasible for the project. We analyzed sensors that had proximity capabilities as well as sensors that activated when objects interacted with them.

Reed Switch Sensor

Reed Switches are electromechanical devices that contain two ferromagnetic plates that interact with each other when a magnet is applied in their area. When a magnetic field is present the plates close on each other, inducing electricity as a result. Depending on the situation applied, the behavior of reed switches can trigger certain events. In our case, when the magnet attached to the garage door goes into the range of the reed switch, the device communicates to the electronic device saying that the door is closed; the door status would be open if the magnet is out of the device's range. Due to there being no physical contact between the magnet and the reed switch, the lifespan of the switch is very long.

Hall Effect Sensor

Like reed switches, hall effect sensors use electromagnetism to provide electrical signals to a circuit. Hall effects rely on the existence of magnetic fields to induce an output, meaning that depending on the polarity of the magnet the behavior will vary on the hall effect switch. They can either be unipolar, which only uses one polarity or bipolar, which uses both polarities of the magnetic field. Hall effects have a long lifespan like reed switches due to having no physical contact to produce their output. In order for the hall effects to detect the magnetic field there must be a current applied to the system, which can be detrimental for systems that require low power usage.

Limit Switch Sensor

Limit switches are electromechanical switches that activate when a physical force is applied to an actuator. In the case of using limit switches as an

application for our garage door status, when the garage door closes a physical part of the door comes into contact with the actuator, allowing the circuit to activate and notify the user that the garage door is closed. Although this is simple and easy to install, due to the physical contact the switch gradually wears down, compared to devices that use only light or magnets to activate, which virtually does not degrade.

Ultrasonic Sensor

Ultrasonic sensors are a proximity sensor that uses ultrasonic vibrations to detect the presence of objects and calculate the distance from them. Because ultrasonic waves are a high frequency sound, they do not operate with color in mind. Their independence of color allows their measurement accuracy to be unaffected by transparent and translucent materials. Furthermore, they can be adapted to detect objects from a wide range. Performance depends on conditions of the location as temperature and humidity can affect sound; if the interior of the garage is either too cold or hot, sound might travel differently compared to normal conditions.

Infrared Sensor

Infrared (IR) sensors are proximity sensors that use infrared light to detect infrared radiation. Their high frequency does not allow IR to travel far distances unlike ultrasonic and they are very accurate when applied on a straight line. They are effectively the best if there is no visible light in the area, meaning that they should be applied in dark areas. Even if the main detecting device is improperly installed by a few degrees this can heavily affect the performance of the ir sensor.

LiDAR Sensor

Standing for Light Detection And Ranging, LiDAR uses laser beams to detect the distance and position of an object. The use of lasers allows greater accuracy and precision compared to using infrared and sound waves. The light that is being reflected off of surfaces returns to the lidar sensor for it to make proper time of flight measurements. These exact measurements require internal electronics to be implemented to calculate the real time systems. As such implementing a LiDAR sensor would be very costly to install due to how advanced the technology is.

Motion Sensor

Motion sensors, as mentioned in their name, detect whether or not some kind of motion is in range to activate a response. There are three kinds of motion sensors that are used: Passive Infrared Sensors (PIR) sense a change in temperature from an oncoming object compared to the background temperature. Microwave sensors send out frequency waves and detect if there is a shift in frequency depending on the object that is moving. The third kind is combining both passive IR and microwave sensors to accommodate both strengths and also reduce the chance for any false alarms.

Camera Object Detection Sensor

Alternatively, instead of using external sensors to detect the presence of an object, the camera can be used to detect objects via the use of computer vision. There are different methods that can be applied to allow the camera lens to analyze and process the existence of physical objects, but they all do require an instance of deep learning to keep a database of what can be classified as an object.

Table 3.2.1: Comparison of Door Detection Sensor

Sensor	Type	Cost	Accuracy	Distance Range	Detection Activation type
Reed Switch	Status Detection	Cheap	High	N/A	Magnet
Hall Effect	Status Detection	Expensive	High	N/A	Magnetic Field
Limit Switch	Status Detection	Cheap	High	N/A	Mechanical
Ultrasonic	Object Presence	Cheap	High	Up to 10m	N/A
IR	Object Presence	Cheap	Low	Up to 5m	N/A
LiDAR	Object Presence	Expensive	High	Up to 200m	N/A

Motion	Object Presence	Cheap	Medium	Up to 20m (PIR)	N/A
Computer Vision	Object Presence	Expensive (time consuming)	Medium	Determinant	N/A

3.2.2 Door Status Parts Selection

Reed Switches, although very effective and cheap for our implementation in DoorSure, also needs to be specifically selected in terms of size and durability.

MC-38

The MC-38 is a reed switch that is contained in a plastic housing, making interaction with a magnet in the context of the garage door closing will have more durability compared to glass.

SR6-004

The SR6-004 is a small size reed switch with a metal housing containment, which, like the MC-38, will be very durable when the magnet attached to the garage door comes into contact with the SR6-004. They are however expensive due their use in an industrial application.

MKA-14103

The MKA-14103 is a reed switch that has a compact size, but due to their glass housing, the risk of shattering is drastically increased, compared to the other reed switches for comparison.

Table 3.2.2: Reed Switch Parts Selection

Reed Switch	Type	Size & Construction	Price
MC-38	NC	Small, Plastic housing	Cheap

SR6-004	NO	Small, Metal housing	Expensive
MKA-14103	NO	Small, Glass housing	Cheap

3.3.1 Obstacle Detection Technology

The obstacle detection sensor is typical in automation, robotics, and safety systems. It helps detect objects in the path and measures distances using technologies like ultrasound, infrared red, laser, or camera AI. Each type has its own advantages and disadvantages when applied to different environments and scenes.

One of our goals is to feature the ability to monitor the garage door's clearance of the environment. Obstacle detection sensors warn the user when an object is detected within the garage door's path, letting them know there's something in the way. We need something that detects in real time, has good accuracy, and has low power consumption.

When selecting an obstacle detection sensor, several parameters must be considered. One key requirement is reliability under different environmental conditions, such as light changes or environmental noise. Additionally, the sensor should have a detection range that is good enough for our needs while maintaining a balance between accuracy and power consumption. Low power consumption is important since the device is running on batteries, so there is no need to replace it frequently.

Ultrasonic Sensor

An ultrasonic sensor measures distance using sound wave technology. It uses a transducer to send and receive ultrasonic pulses that relay back information about an object's proximity. High-frequency sound waves reflect across boundaries to produce distinct echo patterns (MaxBotix Inc.).

Ultrasonic sensors work by echolocation, similar to how bats and dolphins navigate their surroundings. They have two main components: First, the transmitter emits ultrasonic waves, which travel through the air until they hit an object. Second, the receiver picks up when the waves bounce off the object. Then, the sensor calculates the distance based on how long it takes for the

wave to return. This process keeps happening in real time, which means it is constantly measuring all the time.

Ultrasonic sensors have several advantages. One of the most significant is that they can operate under almost any lighting condition. This is due to their reliance on sound waves, unlike optical sensors, which are affected by bright or dark lights. Additionally, they do not require physical contact with the object. This reduces any wear and tear over time, which makes them reliable. Another advantage is that they can detect different materials and surfaces. Last, they have good accuracy, making them a reliable choice for this project.

Despite their advantages, ultrasonic sensors have some limitations. First, environmental factors like temperature, humidity, and air pressure can affect the accuracy of distance measurements. Second, they are limited to short or medium distances and are not suitable for long-distance detection. Third, multiple sensors may cause inaccurate readings. Fourth, they have a minimum detection distance. The result may not be accurate when the object is too close.

Ultrasonic sensors are widely used in modern life. They are often used in parking sensors to help drivers detect obstacles while backing up in automotive applications. They also help robots navigate their surroundings and avoid obstacles, such as vacuum robots at home.

Infrared Sensor

An IR sensor is an electronic device that detects IR radiation falling on it. Proximity sensors (used in touchscreen phones and edge-avoiding robots), contrast sensors (used in line following robots), and obstruction counters/sensors (used for counting goods and in burglar alarms) are some applications involving IR sensors. (CD-Team)

Infrared sensors work by emitting or detecting infrared radiation. There are two main types: active and passive infrared sensors. Active infrared sensors emit infrared light and measure an object's reflection to determine its presence or distance. Passive infrared sensors detect infrared radiation emitted by objects and are used for motion detection. The sensor triggers a response when an object moves in front of the sensor due to its infrared level changes.

There are many advantages of infrared sensors. First, they are low cost and easy to implement. They are inexpensive compared to other types of detection sensors, making them a popular choice for simple applications. Second, they have low power consumption, which is ideal for battery-powered devices. Third,

they can provide fast object detection in short-range applications. Fourth, passive infrared sensors can detect motion without physical contact.

Despite their advantages, infrared sensors have some limitations. First, they are sensitive to environmental conditions. The result can be inaccurate when the environment is not ideal. Second, they can only detect short ranges, which limits their capability when it comes to further distances. Third, they can be affected by the heat source. High temperature can reduce their accuracy and may cause a false trigger.

Infrared sensors are widely used in modern life. They measure body temperature by infrared radiation from the skin in the healthcare system. Passive infrared sensors trigger an alarm when movement is detected in security systems. They are commonly found in remote controls, where the remote transmits signals to TVs.

Lidar Sensor

Light Detection And Ranging (LiDAR) is a laser-based remote sensing technology. The idea behind LiDAR is quite simple: point a small laser at a surface and measure the time it takes the laser to return to its source. (BEAUD)

Lidar sensors use laser beams to scan areas and measure distances. They emit rapid laser pulses, reflecting the surface and returning to the sensor. The system calculates how long each pulse can return into distance measurement. The lidar can build a detailed 3D map of its surroundings. Some are 360-degree fields of view, and others are fixed to specific areas.

There are several advantages to lidar sensors. They are highly accurate and work in both bright and dark environments. They provide precise distance measurements, such as detailed mapping and object detection. In addition, they detect objects at a distance, which is helpful for self-driving cars and drones. They can scan large areas quickly.

Although Lidar sensors have many advantages, they also have some limitations. First, they cost much more than other sensors. Second, they require more power, making battery-powered devices suffer. Third, detecting certain surfaces, such as reflective or transparent materials, can be difficult. Fourth, some weather conditions can interfere with laser pulses.

Lidar sensors are used in many industries due to their high accuracy and 3D mapping. They help detect objects and measure distances in self-driving cars.

They create high-precision maps for a number of things. Robots can use LiDAR sensors for obstacle detection and navigation.

Camera with AI

The camera with AI is an advanced vision system that combines image capture with artificial intelligence to analyze and interpret visual data. Unlike traditional cameras, which only record images, it can recognize objects, track movement, and make real-time decisions based on what it sees.

AI cameras work by using machine learning algorithms to process the images or videos they capture. They are equipped with advanced software that can detect objects, recognize faces, recognize patterns, and even predict movements. Some AI cameras use deep learning models to improve their accuracy, learning from past data to improve their detection and classification capabilities. Depending on the application, these cameras can operate independently or in conjunction with cloud-based AI systems for greater processing power.

AI cameras have many advantages. One of the most significant is that they can analyze detailed visual data in real time and recognize different objects. They can provide more than just distance measurement. Also, they can be updated and trained through software, which can be improved over time.

Although AI cameras have many advantages, they also have some limitations. First, they require processing power, which can lead to higher power consumption and may require additional hardware such as GPUs. Second, they require images to analyze and train. Privacy could be a concern. Third, they may struggle in low or bright lighting conditions in which the image clarity is reduced.

AI cameras have been widely used in recent years. They have been used in security access, facial recognition, and so on. Also, they can help vehicles recognize traffic lights, obstacles and more. Even in healthcare, they assist in diagnosing medical treatments through image analysis.

Table 3.3.1 Obstacle Detection Technology

Sensor Type	Description	Pros	Cons
Ultrasonic Sensor	Measuring distance using sound waves	-Good accuracy -Works in many lighting	-Affected by environmental noise

		conditions	-Power consuming
Infrared Sensor	Detects based on infrared reflection	-Low cost -Simple to implement	-Poor performance in bright light -Short range
Lidar Sensor	Laser to measure the distance	-High accuracy -Works in all lighting conditions	-Cost -Higher power consumption
Camera with AI	Processes images to detect object	-Versatile -More visual information	-Processing power -Privacy concerns

3.3.2 Obstacle Detection Selection

Since we chose the camera with artificial intelligence capability as our object detection mechanism after carefully considering other options, the camera has become a central component in our garage door project. It actively detects objects in the path of the garage door—possibly a human or an animal— so a fast response has been crucial to ensure the safety of users and to prevent any accidents or damage to property. Additionally, it has allowed the user to view a live feed to confirm that the garage door is closed, which serves as an additional verification step alongside the reed switch. Therefore, the camera has served as the object detection mechanism, stopping the door if it detects an object in the way, ensuring both safety and peace of mind for the user.

As safety has been a key priority in this project, we have ensured the implementation of all applicable standards and safety regulations to guarantee reliability and security. Since the camera triggers an interrupt upon detecting an obstacle in the path of the garage door, we have conducted intensive testing to minimize false triggers. Balancing reliability and cost has been key in selecting our camera development board with the appropriate camera module, along with choosing the ultrasonic sensor that best fits the needs of the project.

Ultrasonic Part Selection

The ultrasonic sensor needs to be small, affordable, accurate, and cover a long range.

HC-SR04

The HC-SR04 can reach up to 13ft in ideal conditions, mainly areas that do not produce much electronic signals that can interfere with the sensor's operations. The HC-SR04 has an operating voltage of 5V and can be purchased at a low cost.

RCWL-1601

The RCWL-1601 is similar in specifications to the HC-SR04, except that the sensor can operate under 3.3V. This is useful for devices that only want to operate under 3.3V of power.

HY-SRF05

The HY-SRF05 is similar in construction to the HC-SR04 only that it has a fifth pin for configuring different modes: if the pin is connected to ground then it behaves like the HC-SR04 but when the pin is unconnected the trigger and echo pins will combine into one wire. This also operates under 5V.

Table 3.3.2.1 Ultrasonic Part Selection

Sensor Type	Voltage	Current	Maximum Range	Cost	Power Draw
HC-SR04	5V	2mA	13ft	Cheap	0.01W
RCWL-1601	3.3V - 5V	2mA	13ft	Cheap	0.006W (3.3V), 0.01W (5V)
HY-SRF05	5V	2mA	13ft	Cheap	0.01W

Camera development boards

For Senior Design 1, we chose camera development boards that are equipped with the ESP32 chip. This was so we can easily transition the camera code for the dedicated camera in Senior Design 2. The choice of camera board was important as the PSRAM needs to be high enough for image processing to be consistent, along with the included camera module to be easy to program and flash with.

ESP32-CAM

The ESP32-CAM option stood out as the perfect choice for our project having the perfect balance between cost and capabilities that we need, as they are the cheapest option and are easy to code and flash as the built-in OV2640 camera is easily compatible with the ESP programming. In addition with its 4MB of PSRAM this is enough to view a live camera feed for consistency. A version with the OV7670 camera module exists, but we chose the OV2640 so that we would lower the cost as the OV2640 already getting shipped with the development board which would give us an extra opportunity to save more money. Moreover, this board would support multiple low power modes which is highly desirable since we would be operating on a battery and low frequency maintenance would be one of our goals. The ESP32-CAM would require a micro USB to Serial port CH340 Board but since it is already shipped with the development board and still maintaining being the cheapest option, therefore, we chose the ESP32-CAM with the OV2640 camera module as it is the cheapest, most available and library supported camera module. The downside of ESP32-CAM is it's not a dedicated camera instead a development boards. That's the reason we can't use it as the final product.

ESP-EYE V2.1

The ESP-EYE would be a miniature ESP32-based development board that has an ESP32 dual core Tensilica LX6 processor with 8 MB PSRAM, coming equipped with 2 LEDs, a MEMS Microphone and an OV2640 camera module which would be 2MB camera, the ESP-EYE generally is optimized for face and speech recognition application making it easy and simple for new electronics users to experiment with these features and possibly configure it, although this version would meet our criteria for the project but due to its high price and unavailability, we didn't choose the ESP-EYE V2.1.

ESP-S3-EYE

The ESP-S3-EYE represents the next generation in the ESP32 camera boards, this would make this version a miniature AI development board optimized for image recognition and audio processing, this version would be based on the ESP32-S3 and ESP-WHO framework, ESP-WHO is an image processing development platform based on Espressif chips with many examples online human detection, face recognition and many other applications, The ESP-S3-Eye is perfect for heavy image processing application with plenty of memory of 8 MB octal PSRAM, 8 MB flash and a 2 MB camera to develop a wide range of AIOT applications, this board comes pre-equipped with an LCD display and a microphone which would be needed for image processing and audio processing, it also comes with default firmware which would be

automatically loaded upon connecting the board to a power supply via a micro USB cable, then the LCD display will show the live video stream recognizing the human face of the user. The inclusion of the LCD display along with the usage of the S3 capabilities make this product expensive to acquire.

T-Camera plus

The T-Camera plus based on the ESP-32-S3 microcontroller with Tensilica Xtensa dual core lx7 microprocessor with OV2640 camera module and also supporting OV5640, one key advantage this board has is the 1.3 LCD capacitive touch screen and a microphone, this would allow immediate visual feedback allowing users to monitor live camera feed without the need to integrate with a web application or an external screen, this feature would be incredibly useful in a variety of applications such as smart surveillance doorbells, although this development has more capabilities and offer more advanced features than the ESP32-CAM such as the touch screen and the microphone, it would also bring some trade off in the software as it adds more complexity. Like the S3-EYE, because the T-Camera plus uses an LCD screen with S3 capabilities this also ends up to be an expensive component.

AI Vision WonderCam

The AI vision camera module would be an unconventional choice boasting 8 built in functions including color, face, number and road sign recognition, vision line following, image classification and feature learning, this module can be used for people without extensive programming background since you can implement multiple AI features with one click training using a built in display and control key, moreover WonderCam's I2C connector makes it compatible with multiple MCUs including Arduino, Raspberry Pi and ESP32, also, the AI vision WonderCam supports programming using Arduino IDE while providing Arduino routines and source code so users can easily develop projects by themselves.

Despite it's highly sophisticated capabilities, LewanSoul (the brand behind WonderCam) focused on making their device as user friendly as possible by incorporating two lego compatible holes, this board would be perfect to empower hobbyists to integrate AI Vision functionalities to various projects including lego based projects, although this board is for beginners but it would work for our project, but since it's cost of nearly \$60, we chose the ESP32-CAM as it has a much lower price.

XIAO ESP32 S3 Sense

The Xiao ESP32 S3 known for it's thumb sized compact design of 21 x 17.5 mm with light weight which makes it perfect for space limited projects such as a

smart watch, equipped with a built in USB for ease of programming and debugging, moreover, it is provided with a detachable OV2640 and a digital microphone enhancing its small size design, another key advantage of the Xiao ESP32 S3 board is the outstanding RF performance supporting 100 meters or more of remote communication when connected to U.FL antenna, the Xiao ESP32 S3 leverages a dual core ESP32 S3 chip supporting bluetooth and wifi connectivity, it also contains an external SD card slot for an external 32 GB of memory which make it suitable for AI driven applications by supplying massive amount of data storage and processing capabilities of 8 GB PSRAM, although, this board has numerous advantages, the increased capabilities while maintaining the small size results in high cost compared to the ESP32-CAM.

Final Choice: ESP32-S3 N8R8

After evaluating multiple camera development boards throughout the project, we selected the ESP32-S3 N8R8 for our final design. It includes 8MB of PSRAM, which is essential for handling high-resolution image capture and efficient communication with AI series like Google Gemini. Unlike other ESP32 based boards, the ESP32-S3 N8R8 also supports external dedicated camera modules, giving us the flexibility to upgrade from the built-in OV2640 if needed. Its dual-core processor, USB programming support, and built-in Wi-Fi and Bluetooth capabilities made it a perfect choice for our project.

Table 3.3.2.2: Camera Development Board Comparison for the Prototype

Camera development boards	Short description	PSRAM Size (MB)	Pre-installed module and compatible modules	USB-to-UART bridge	Cost
ESP32-CAM	WiFi+Bluetooth dual-mode development board uses PCB on-board antennas and cores based on ESP 32 chip	4	OV2640 (OV7670)	Needs a USB to Serial port adapter CH340	\$ 9.25
ESP-Eye V2.1	WISOC	8	OV2640	CP2102	\$24.95

	ESP32 dual core Tensilica LX6 processor with WIFI and Bluetooth				
ESP-S3-EYE	Miniature AI development board with an LCD display	8	OV2640	CH340	\$45.00
T-Camera Plus S3	T-Camera-Plus-S3 ESP32 with 1.3 inch touch LCD	8	OV2640 (OV5640)	CP2104	\$51.09
AI vision WonderCam	AI vision boasting 8 built in functions including image classification	8	OV2640	CP2102	\$59.99
XIAO ESP 32 S3 Sense	ESP32 S3 Sense dual core Xtensa processor chip operating up to 240 MHZ	8	OV2640 (OV5640)	Built in USB	\$23.99
ESP32 S3 N8R8	Dual core, 8MB PSRAM, 8MB flash	8	OV5640	Built-in USB	\$15.99

Based on our choices, we chose the ESP32 S3 N8R8 development board as it achieves the perfect balance between low cost and capabilities needed for the project which are a high quality, highly library supported camera to ensure reliability.

3.3.3 Camera modules

As using development boards are not allowed for significant PCB design, we transitioned to dedicated camera modules that can be easily flashed from the microcontroller, along with being easy to flash and program on. Image resolutions, lens size, library support, power consumption, and cost are important to choosing our choice of camera.

OV2640

The OV2640 is the most used and widely supported camera module since all the development boards above have the OV2640 pre-installed, as it achieves the perfect balance between image quality, power consumption and cost, having high image quality of 2 MP that it can be used for many AI application such as image recognition and in the context of our project object detection, moreover, having a balanced image resolution reflected in the power consumption, since the system operates using a battery, power consumption is significantly important when choosing every component in our project to lower overall power consumption and as a result of that reduce maintenance needs, which would enhance system reliability, the OV2640 popularity would reflect in strong library support and resources that would help us integrate it into our project and a low cost compared to other camera modules.

OV3640

The OV3640 is another camera module we considered, offering a higher resolution of 3 MP compared to the OV2640 which would result in better live feed and increased capabilities in terms of object detection. This would lead to increased power consumption, therefore lowering the overall battery life. The OV3640 has a moderate library support which would impact community support for this camera module.

OV5640

The OV5640 has better performance specifications compared to the OV2640, having a resolution of 5MP, which allows in higher image quality at the cost of increased power consumption. Although the OV5640 has moderate support with the ESP library compared to the OV2640, transitioning code designed for the 2640 to the 5640 is fairly simple to do without any issues. In addition, the dedicated breakout boards contain the necessary pins for the ESP that can communicate and initialize with.

OV7670

The OV7670 has a 0.3MP resolution quality, which would not be sufficient in applications that require image processing such as AI detection. Although they are sold at a low price, the support for the ESP coding library is very poor, making development around the 7670 difficult to program with.

OV7725

Like the OV7670, The OV7725 provides a resolution of 0.3 MP, which also would not apply well with image processing applications. The limited support for the OV7725 will make programming for this camera module difficult to program with.

NT99141

The NT99141, while having a 1MP resolution, which is feasible for image processing, does not have a dedicated breakout board to implement with. Designing a PCB around the NT99141 is difficult, as creating PCBs for camera interfaces are known to be really difficult to troubleshoot with and considering time as a main constraint, much of our time cannot be spent to getting a custom camera pcb properly working for a very long duration.

Table 3.3.3: Camera module

Camera Module	Resolution (MP)	Lens size	Library support	Power requirement	Cost
OV2640	2	1/4"	High (Most used and supported)	Moderate	\$8.99
OV3640	3	1/4"	Moderate	Moderate-High	\$13.63
OV5640	5	1/4"	Moderate	High	\$24.99
OV7670	0.3	1/6"	Low	Low	\$2.10
OV7725	0.3	1/4"	Moderate	Low	\$4.72
NT99141	1	1/4"	Low	Moderate	\$21.58

Although continuing to use the OV2640 from the development from Senior Design 1 would be the ideal choice, the main problem is the limited amount of

dedicated breakout boards that exist. These breakout boards also do not contain the pins that the ESP chip needs to supply the proper data to send information to and initialize the camera.

Ultimately, we decided to use the OV5640 for our final project since it would be compatible with the ESP32-S3 and the existing code we have developed for the ESP32-CAM is an easy transition to due to the OV5640's ESP library support.

3.4.1 Temperature and Humidity Selection

In situations where the garage room reaches temperatures that could cause the DoorSure devices to become inoperable, the devices needed to alert the user that they were about to power down due to operating in conditions that could potentially damage the system. Although including temperature monitoring was an obvious choice, humidity was also incorporated into the system to accommodate areas with tropical and humid climates. To reduce complexity in building our devices, we decided to select components capable of detecting both temperature and humidity within a single package.

DHT11

The DHT11 is a ultra low cost sensor that can sense both temperature and humidity. Many devices that require low power usage can utilize this component. The ranges for temperature and humidity are suitable for most situations, unless they are used in very extreme conditions, but for home applications they are sufficient to be used.

DHT22

The DHT22, like the DHT11 can also sense temperature and humidity. The main difference is the upgraded specifications in the ranges as the DHT22 can capture temperature and humidity ranges beyond the values of the DHT11. Due to its prioritization on high precision the component is more accurate than the DHT11.

SHT7

Created by Sensiron, the SHT7 is a humidity and temperature sensor that uses internal signal processing to provide a fully calibrated and fair measurement. The high quality build of the SHT7 and advanced innovations used to record the values to sense the environment does make this component significantly expensive and over-capable in a household environment.

Table 3.4.1: Temperature and Humidity sensors

Sensor	DHT11	DHT22	SHT7
Temperature range (°C)	-20 - 60	-40 - 80	-40 - 123.8
Temperature Deviation (°C)	+/-5	+/-0.5	+/-0.5
Humidity Range (RH)	5-95	0-100	0-100
Humidity Deviation (RH)	+/-5%	+/-2%	+/-3%
Power Requirement (V)	Min 3.3V, Max 5V	5V	Min 2.4V, Max 5V
Current Draw (A)	2mA	2mA	2mA
Cost	\$5	\$6.25	\$30

3.5.1 Motor Controller Technology

With the implementation of our servo motor, we have required the use of a motor controller. Motor controllers have been responsible for regulating the operation of motors, ensuring that the correct amount of speed, torque, direction, and power is applied. Without the use of a motor controller, the motor would have no control over its output, leading to major inefficiencies within the system and potentially damaging the device's power systems. To properly manage motor operation, motor controllers have used voltage control to adjust speed, pulse width modulation to apply precise speed control, polarity switching to manage motor direction, and current regulation to control the torque applied. There have been four main types of motor controllers considered for implementation in our project.

DC Controllers

DC motor controllers mainly operate and control motors using direct current. Because of their ease of control with current they have great precision for controlling the amount of speed a motor should exert. They are best used when electronics require precise speed control over motor performance.

AC Controllers

AC on the other hand uses alternating current. The use of alternating current allows the motors to operate efficiently with minimal power consumption. AC controllers are usually equipped with variable frequency drives to control the speed and torque of a motor. They are usually found in large motors or contraptions that would consume energy if a DC motor controller was used.

Servo Controllers

Servo controllers are used in situations where accurate positioning is a priority, like in manufacturing. By sending pulse width modulation signals to the motor, the motor shaft can apply a specified angle, based on the information the user gives to the controller.

Stepper Controllers

Using incremental movements, stepper controllers can provide precise positioning. They are similar to servo controllers, but use a specific sequence of electrical signals to make incremental positioning, which is useful for applications that require accurate positioning.

Table 3.5.1: Comparison of Motor Controllers

Controller Type	DC	AC	Servo	Stepper
Cost	Expensive	Cheap	Expensive	Expensive
Efficiency	Poor	Accurate	Accurate	Accurate
Power delivery	Current	Current	Pulse Width Modulation	Pulse Width Modulation

3.5.2 Motor Controller Servo Selection

In our system, the servo motor is responsible for the movement that physically presses the user's garage button to change the state of the door—either from open to closed or vice versa. Because of this, the primary factors we have considered when selecting a servo motor have included torque, weight, and size. The servo needed to be lightweight and compact enough to attach securely to the user's garage button without adding unnecessary strain or

obstructing other components. At the same time, it required sufficient torque to reliably press the button without stalling, which could result in delays or unsuccessful execution—especially if the button required significant force to activate. A servo in the 5–10 kg/cm torque range has been considered ideal, depending on the resistance of the button. Based on the button’s resistance, we have increased voltage to the servo using a voltage regulator. Additionally, the mounting system needed to ensure stability, preventing the servo from shifting or misaligning after repeated use. Given that the servo was expected to be used frequently, power efficiency and durability have also been key factors—metal gears have improved longevity, while a low-power design has been beneficial for systems operating on limited energy sources. We have also factored in the speed of the servo motor, as we aimed to minimize delays as much as possible. Ultimately, we selected a high-torque, low-profile servo that balanced power, efficiency, and cost.

SG90

The SG90 Servo Motor by Beffkkip has an operating voltage range of 4.8 to 6.0V with an idle speed of .11 seconds per 60 degrees at 4.8V and .08 seconds at 6.0V. It rotates at an angle of 180 degrees which is more than ample for our action. Although being a super cost-effective option -- with a 4-pack costing \$7.99 – this servo motor uses a mix of plastic and copper metal gears, which is not highly reliable. Depending on how much torque is needed to activate the button, having a plastic and metal gear mix increases the chances of a stall or delay in the action due to too much pressure. Furthermore, with frequent movements as expected from opening and closing the door might cause fast wear and tear. The SG90 Servo Motor by Beffkkip has an operating voltage range of 4.8 to 6.0V with an idle speed of .11 seconds per 60 degrees at 4.8V and .08 seconds at 6.0V. It rotates at an angle of 180 degrees which is more than ample for our action. Although being a super cost-effective option -- with a 4-pack costing \$7.99 – this servo motor uses a mix of plastic and copper metal gears, which is not highly reliable. Depending on how much torque is needed to activate the button, having a plastic and metal gear mix increases the chances of a stall or delay in the action due to too much pressure. Furthermore, with frequent movements as expected from opening and closing the door might cause fast wear and tear.

MG90S

The MG90S Servo Motor by Miuzei has an operating voltage range of 4.8 to 6.0V with an idle speed of .1 seconds per 60 degrees at 4.8V and .08 seconds at 6.0V. According to the internal structure diagram provided on their product page, the servo also includes a CNC-process for high precision, making sure of no

angular deviation. This is crucial for making sure the motor is able to hit the garage button precisely. Additionally, its full-metal gear construction enhances durability, reducing the risk of gear stripping over time. However, with a stall torque of only 2.2 kg/cm at 6.0V, it may struggle if the garage button requires more force to activate.

MG90D

The MG90D Servo Motor by HOOYIJ is an upgraded version of the previous MG90s. It has an operating voltage range of 4.8V to 6.0V, with a speed of .1 seconds per 60 degrees and .08 seconds per 60 degrees, respectively. The stall torque at 4.8V is 2.1kg/cm and at 6.0V is 2.4 kg/cm. It also has metal gears similar to its predecessor, providing increased durability and resistance to wear. Compared to the MG90S, the MG90D offers slightly higher torque, which can improve reliability when pressing the garage button. Its compact size and precise movement make it a viable option if the required button force is low.

MG996R

The MG996R Servo Motor by Deegoo has an operating voltage range of 4.8V to 6.0V. It features a stall torque of 9.4 kg/cm at 4.8V and 11 kg/cm at 6.0V, making it significantly stronger than smaller servos like the MG90S or SG90. This increased torque ensures that even stiff or resistant garage door buttons can be pressed reliably without the risk of stalling. However, this servo is noticeably larger and heavier than its predecessors, weighing 55g with dimensions of 40.7 x 19.7 x 42.9 mm. While the added weight may pose mounting challenges, its full metal gears contribute to long-term durability, reducing wear over time from the expected operation frequencies. The operating speed is 0.17 seconds per 60 degrees at 4.8V and 0.14 seconds at 6.0V, which, while slower than smaller servos, is still fast enough for practical use for our garage door button system.

RDS3225

The RDS3225 Servo Motor by ANNIMOS has an operating voltage range of 5V to 6.8V with an operating speed of .16 seconds per 60 degrees at 5V and .14 seconds per 60 degrees at 6.8V. This servo motor has a lot of tradeoffs when compared to the previously mentioned servo motors. It requires higher voltage, is heavier and slightly slower, but exerts a significantly higher torque, not stalling until almost three times as much kg/cm as other motors. The metal gears place it on par with the MG90s in terms of durability but its heavy weight due to being fully metal gears might cause the device to be more difficult to place over the garage door button. The higher torque does, however, significantly decrease risk

of not being able to exert enough pressure to press the button, therefore increasing input accuracy.

Table 3.4.2: Servo Motor Comparison

	SG90	MG90s	MG90D	MG996R	RDS3225
Stall Torque (kg/cm)	1.4 at 4.8V	1.8 at 4.8V	2.1 at 4.8V	9.4 at 4.8V	24 at 5.0V
	1.9 at 6.0V	2.2 at 6.0V	2.4 at 6.0V	11 at 6.0V	28 at 6.8V
Speed (seconds/60 degrees)	0.11 at 4.8V	0.10 at 4.8V	0.10 at 4.8V	0.17 at 4.8V	0.19 at 5V
	0.08 at 6.0V	0.08 at 6.0V	0.08 at 6.0V	0.14 at 6.0V	0.14 at 6.8V
Gear Material	Plastic	Metal	Metal	Metal	Full metal
Weight (g)	9	13.4	13.5	55	60
Dimensions (mm)	22.2 x 11.8 x 31	22.4 x 12.1 x 28.5	22.8 x 12.2 x 28.5	40.7 x 19.7 x 42.9	40 x 20 x 40.5

3.6.1 Voltage Regulation Technology

The DoorSure devices required a voltage regulator to be implemented to provide the proper voltage to the components that require power and prevent any potential short circuit overloading. There are mainly two types of regulators, linear and switching regulators. Linear regulators maintain a desired voltage by stepping down the input voltage to the target voltage. Despite their lower price point and simple construction these regulators have low efficiency and produce a lot of heat dissipation, making them inefficient to use in high level applications. Switching regulators have a complex design but allows them to be efficient in converting voltages. They can not only be step-down, but also boost low voltage sources into the targeted voltage.

Linear Regulators

Because of their simplistic design linear regulators only have the capability to step down voltages. Due to their low efficiency in terms of power dispersion, they can only be used for circuits that do not produce a significant amount of power to minimize the loss of power through heat. Two main types of linear regulators exist: Series regulators have the input load and transistor connected to series whereas shunt regulators have the two components connected in parallel. The difference comes from how each circuit manages the load. Series regulators work varying the transistor element from the output voltage to make sure they match the requirements of the reference voltage. Shunt transistors compare the feedback and reference voltage to control the specific output. Series regulators are more efficient out of the two as the amount of current drawn is effectively used by the load, therefore not using much power to regulate voltage.

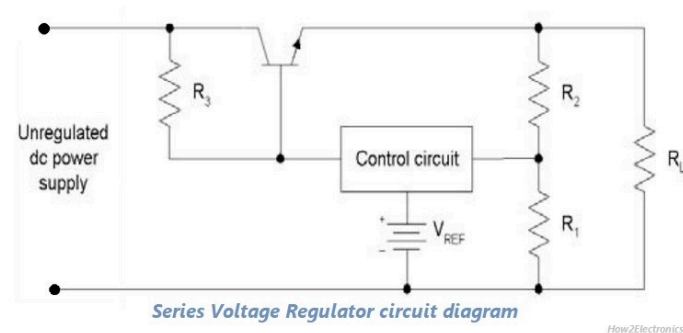


Figure 3.6.1.1: Series Linear Voltage Regulator (Subedi, 2024)

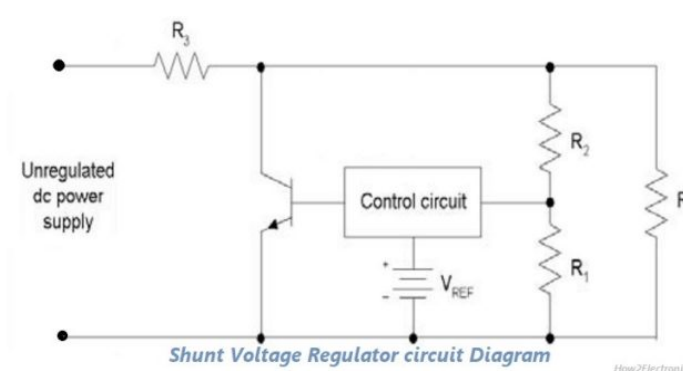


Figure 3.6.1.2: Shunt Linear Voltage Regulator (Subedi, 2024)

Switching Regulators

Switching regulators control the period of when the input voltage is active in supplying to the device by switching the element on and off via pulse width modulation to get the desired output voltage. The switching effect allows very minimal heat to be dissipated, which is why switching regulators are way more effective than linear regulators. Due to the nature of switching regulators, they can not only be used to step down voltages (buck converters) but also step up voltages (boost converters).

Table 3.6.1: Switching Regulator part selection

	Linear	Switching
Efficiency	Low	High
Heat Loss	Large	Minimal
Voltage Conversion	Step-Down	Step-Up, Step-Down
Design	Simple	Complex
Cost	Cheap	Expensive

3.6.2 Switching Regulator Parts Selection

With switching regulators chosen due to their efficiency, which part being chosen all depends on their reliability and size. The circuitry surrounding the regulator can all differ in size and as the system devices are in a small form factor, the same can be applied with the choice of regulators.

LM2576

A well known switching regulator, the LM2576 is easy and simple to install due to their low component requirements. They can output from 3.3V, 5V, and even

versions that have adjustable output. When designing a pcb around the LM2576, the design footprint can be rather large, which can be detrimental for project designs that prioritize small footprints.

AP6320X

The AP6320X series of regulators offer a high efficiency rate at a quiescent current of 22 microamps, which leads to a smaller pcb design footprint. This can output either 3.3V or 5V, along with having good performance of resisting electromagnetic interference. The performance comes at a caveat, unless the pcb is designed correctly, the increased sensitivity may potentially cause unintentional power spikes.

TI TPS54202

The TPS54202 regulator has integrated switching FETs to prevent any potential power overloads from occurring. With the inclusions, this makes the pcb footprint creation small, but not as small as the ap6320x series. In addition, the EMI is reduced by their spread of frequencies, making the regulator less sensitive.

Table 3.6.2: Switching Regulators part selection

	LM2576	AP6320x	TI TPS54202
Efficiency	80%	88%	80.2%
Output current	3A	2A	2A
Quiescent current	50 μ A	22 μ A	45 μ A
PCB footprint	Large	Small	Medium
Cost	Expensive	Cheap	Moderate

3.7.1 Batteries Technology

Choosing the appropriate battery for the garage door status detector has been critical to ensuring long-term operation and reliable performance. For our detector system, we carefully considered the type of battery technology to

implement in our devices. Before selecting a specific battery, we analyzed the differences between primary and secondary batteries. Primary batteries, which are single-use and non-rechargeable, were deemed unsuitable for our needs. In contrast, secondary batteries are rechargeable and offer greater long-term value. After evaluating these options, we concluded that secondary batteries would be the most suitable choice for our system. This decision was made to maximize device uptime and reduce the need for frequent battery replacements. Additionally, rechargeable batteries would minimize long-term costs and enhance user convenience by eliminating the need for constant battery purchases.

Table 3.7.1: Battery Technology Comparison

Battery Type	Description	Pros	Cons
Alkaline (AA/AAA)	Common Disposable Batteries	-Low cost -Availability	-Lower capacity -Less lifespan
Carbon-Zinc	Primary Batteries	-Good for low drain systems	-Low shelf life -Less capacity
Lithium	Secondary Batteries	-High Capacity -Rechargeable	-Higher cost

Although any primary battery could technically have worked with the system, we evaluated which type would be the most effective and long-lasting for installation in our devices. Alkaline batteries are commonly used in household products due to their widespread availability and low cost. However, they are prone to leakage over time and have a limited lifespan. Zinc-carbon batteries, while inexpensive, are best suited for low-power devices and suffer from a short shelf life, making them suboptimal for continuous operation. Lithium batteries, on the other hand, offer a significantly higher capacity and longer lifespan compared to alkaline and zinc-carbon options. While they are slightly more expensive, their superior performance and resistance to leakage made them the most suitable option for users who do not want to frequently replace batteries.

In addition to selecting the appropriate battery type, we also considered the size and capacity of the battery. The battery must be compact enough to fit within the constraints of both the primary and secondary devices, while still supplying sufficient power to meet the system's energy demands. Balancing physical

dimensions with power capacity has been essential to ensuring optimal performance without compromising the device form factor.

3.6.2 Batteries Selection

AA Batteries

AA batteries are one of the most common sizes of batteries the general consumer uses and while they would be a good use for the devices, they would be insufficient to use in the long run. As AA batteries only hold a voltage of 1.5 V, this would require two AA batteries connected in series to properly function, which would contradict the design language in the small form factor. Since we are also using lithium batteries in our project buying a pack of 4 would be expensive to purchase. For these reasons the AA batteries are not a good choice to use.

AAA Batteries

AAA batteries are similar to AA batteries but only with a slightly smaller length and depth. Despite the slightly smaller size they still have the same issues as mentioned in AA batteries; only 1.5V of voltage and requiring at least 2 AAA batteries for each of the devices making the overall enclosure of the design slightly bulky. The AAA batteries would also not be good to use for the project.

9V Batteries

9V batteries are the only kind of battery with a square shape, and 1 battery can be used per device. While 9 volts is enough to supply the ESP32, the large value of the voltage can cause problems even if a step down regulator is used. (need to find sources other than bs and chatgpt) In addition, because we are using lithium batteries a lithium 9V battery would be expensive, on average around 5-7 dollars per 9V battery is not ideal for the project.

123 Batteries

Compared to AA batteries, 123 batteries have a small and compact size that carry 3V of voltage. They have been applied to many electronic systems (like smart door locks) and because they are mainly lithium batteries their battery life is long. The disadvantage of using 123 batteries is the cost of each battery due to the advanced technology in lithium batteries.

CR2 Batteries

CR2 batteries have a similar size to 123 batteries while having the same voltage of 3V. As one battery can suffice (with a voltage regulator), the only disadvantage like the 123 batteries are the high price point for a pack of 2.

18650 Batteries

18650 Batteries are cylindrical batteries similar but are slightly larger than AA batteries and have a higher voltage and capacity. Their small form factor makes them ideal to utilize for the primary and secondary devices but they do not have a standard in capacity, which can be very devastating if the batteries chosen contain the wrong capacity that the project needs.

Li-Po Batteries

Lithium Polymer batteries do not have a dedicated size standard, which means that they can provide higher capacities in a small form factor. One major issue regarding them is the potential swelling if improperly charged. This is mostly mitigated with many having built in battery protection circuits to prevent occurrences like overcharging to happen.

Table 3.7.2: Lithium Battery Size Comparison (Renogy, 2024)

Battery Name	Shape	Dimensions	Voltage	Capacity	Current Draw
AA	Cylindrical	L 50 mm x D 14.2 mm	1.5V	3000 mAH	3A
AAA	Cylindrical	L 44.5 mm x D 10.5 mm	1.5V	850–1200 mAh	.85-1.2A
9V	Rectangular	H 48.5 mm x L 26.5 mm x W 17.5mm	9V	500–600 mAh	.5-.6A
123	Cylindrical	L 34.5 mm x D 17 mm	3V	1500 mAH	1.5A
CR2	Cylindrical	L 27.5 mm x	3V	800 mAh	.8A

		D 16 mm			
18650	Cylindrical	L 65mm x D 18mm	3.7V	Varies	Varies
Li-Po	Rectangular	Varies	3.7V	Varies	Varies

3.7.2 Li-Po Battery Selection

Although 123 batteries were ideal due to their availability in everyday shopping, the capacity of 1200mAh would not be enough to supply power to the devices for a prolonged period of time. Even though the 18650 batteries would be the most desirable, the risk of overcharging without a battery management circuit plus a low quality control of capacity listing made this not feasible. The Li-Po battery size variation along with the charging protection circuitry made this choice the best.

Ideally, at least 2500mAh would be more than sufficient to supply power to the primary and secondary devices, as each device needs to have a current load to be less than 2.5A, which has plenty of room in the case that the device power load suddenly spikes up.

As mentioned before, the size of the li-po battery needs to be within the sizes of both the primary and secondary devices respectively.

Table 3.7.2: Li-Po Battery Size Comparison

Name	Length	Width	Depth	Capacity	Current Draw
724957	57mm	49mm	7.2mm	2500mAh	2.5A
375678	78mm	56mm	3.7mm	2500mAh	2.5A
366890	92mm	68mm	3.6mm	2500mAh	2.5A
505573	73mm	55mm	5mm	2500mAh	2.5A
556065	65mm	60mm	5.5mm	2500mAh	2.5A

All batteries listed do have the sizes within both the primary and secondary devices, but the 556065 batteries were chosen mainly due to the group's previous experience with using this size, along with other projects in the past using them without any issue.

Communication Protocols

Each component connected to the microcontroller, hardware and software wise, needs to follow certain protocols to properly transmit and exchange the necessary data for these parts to be able to safely operate.

3.8.1 Wired Communication Protocols

UART

Standing for universal asynchronous receiver/transmitter, this protocol allows serial data to be exchanged between two different devices. This is initiated through the RX and TX pins of the devices. Communication can be transmitted either one-side (simplex), one side at a time (half-duplex), or both simultaneously (full-simplex). Due to their asynchronous communication both devices need to transmit under the same timing speed, or baud rates. UART is implemented in systems where applications do not require higher speed transfer rates and high throughput due to their low-cost and simple implementation.

I2C

Inter-Integrated Circuits allows synchronous communication between peripherals without any complex wiring. Devices that use I2C communicate using SCL (a clock pin for the main microcontroller to send clock signals to) and SDA (serial data being between the devices).

SPI

Serial peripheral interfaces are a type of synchronous communication that sends data between devices with no interruption. Despite using 4 wires for MISO, MOSI, SS/CS, and SCLK, because of the setup data can be sent and received at the same time and can transfer data at a higher rate compared to I2C.

USB

Universal Serial Bus devices are an interface that allows communication between peripheral devices and host devices. This allows peripherals to be physically swapped to other hosts without the host restarting. The ESP-WROOM-32 device contains a micro-USB port to allow data to be

transferred in order to flash new commands to the microcontroller.

Ethernet

Used in LAN (local area network) environments, ethernet allows data to be sent at fast rates through a local network. This is defined under the IEEE 802.3 and compared to wireless communications, wired networking offers more security and is less susceptible for network interference.

Table 3.8.1: Wired Communication Protocols

Protocol	Transmission Type	Communication type
UART	Serial	Asynchronous
I2C	Serial	Synchronous
SPI	Serial	Synchronous
USB	Serial	Synchronous
Ethernet	Ethernet (not serial or parallel)	Asynchronous

3.8.2 Wireless Communication Protocols

Wi-Fi

This wireless protocol is based on the IEEE 802.11 standards, which enables devices to connect to the internet wirelessly through the use of a router and a modem. Modern Wi-Fi standards have further range for connections and can handle situations where network traffic is congested.

Bluetooth

The bluetooth protocol can communicate with devices wirelessly using 2.4 GHz. Compared to Wi-Fi, the range that devices can connect through bluetooth is very short and consumes very less power. As such this is best used for devices that do not need a long term connection with each other.

Table 3.8.2: Wireless Communication Protocols

Protocol	Range	Bandwidth	Frequency (GHz)	Power Consumption
Wi-Fi	Far	High	2.4, 5	High (if transferring large amounts of data)
Bluetooth	Short	Low	2.4	Low

Software Comparison

Hardware alone is ineffective without the right software to support it. Software and hardware work in tandem to form a functional system, making it essential to dedicate as much thought to software selection as to hardware components. Given the vast array of software resources available today, it is crucial to conduct thorough research and carefully evaluate our options to determine the most practical and efficient technology stack. This section explores our team's software choices, from backend development to frontend implementation, ensuring a well-integrated and robust system.

3.9.1 Remote Networking Technology

One of the most critical components of our project was the remote networking and communication functionality integrated into our device. The system must allow users to control the garage door remotely, even when not connected to the local Wi-Fi network. Aside from the initial physical setup involving sensor installation, camera placement, and positioning of the primary and secondary device within the garage, most user interactions occur through the dedicated web application. Therefore, selecting the right software architecture to support secure, reliable, and remote interaction is essential.

During the early stages of development, it was necessary to consider how our software stack would interface with the physical hardware. The physical devices each connect to the user's home Wi-Fi network. However, for remote access—such as when the user is away from home—a separate, internet-accessible communication pathway is required.

Several potential solutions were evaluated to enable this remote interaction: port forwarding, using an onboard microcontroller web server, or leveraging a cloud server. While port forwarding is technically feasible, it introduces significant

usability and security challenges. Most users are unfamiliar with the steps required to set up port forwarding, and improper configuration may expose their network to security vulnerabilities. This approach would also conflict with our project's goal of offering a simple, user-friendly installation experience.

Cloud servers, by contrast, offer numerous advantages for both the user and the developer. They provide a scalable, secure, and cost-effective method for enabling remote access. Unlike port forwarding, cloud solutions eliminate the need for complex user-side setup and offer robust authentication and encryption mechanisms. Although cloud hosting may involve a small operational cost, it vastly improves the user experience and system reliability—making it the more suitable option, especially if the project is developed further than the course period for broader deployment.

Table 3.9.1: Remote Networking Comparison

	Advantages	Disadvantages
Port Forwarding	- Fast communication times - No cost	- Not user-friendly - Increased security risk - Increases installation time
Cloud Server	- Scalability - Decreased security risk - High storage capacity - User friendly	- Higher cost than port forwarding - Slower communication times

3.10.1 Cloud Server Service Providers

Cloud servers play a critical role in our system by enabling remote control and access to the garage door and its status when the user is away from the local home network to which the system is initially configured. As outlined in the previous section, cloud servers offer significantly greater scalability, enhanced security, and a more user-friendly setup experience compared to port forwarding.

In evaluating potential cloud service providers, we considered key criteria such as the availability of integrated services (real-time databases, user

authentication, and cloud functions like email notifications), pricing models based on monthly data usage, and ease of integration with ESP32 microcontrollers. These considerations are essential for ensuring that our system remains reliable, cost-effective, and simple to develop and maintain.

Amazon Web Services (AWS)

One of the most well-known and popular cloud platforms is AWS. Its extensive service offers a range of adaptability and makes developing significantly easier to handle, an important aspect with a limited time constraint such as for our project. Starting with their Elastic Compute Cloud, or EC2, service offers scalable virtual servers for a range of tasks, such as hosting basic applications.

Additionally, AWS provides Simple Storage Service, S3, an object storage solution suitable for storing backup files, logs, and photos as it supports data redundancy. For databases, their DynamoDB serves as a fully managed NoSQL option that delivers high-speed and low-latency, particularly for IoT. Moreover, AWS provides Lambda, a serverless compute service that executes code in response to events, allowing developers the ability to build applications without the difficulty of managing the servers themselves. AWS also provides AWS IoT Core, a service tailored for managing and securing IoT devices, making it suitable for smart home applications similar to our garage door status system for its ease of access of data between connected devices.

In terms of integration with our microcontrollers, we can utilize AWS IoT core to manage the devices such as allowing the ESP32 to securely connect, send, and receive data from the database. The core supports MQTT, HTTP, and WebSockets. We can also utilize AWS Lambda for serverless processing in response to events such as the esp32 sending an update on the garage status, like changing the current status to open for example, causing the IoT core to trigger a Lambda function that would update the DynamoDB or notify the user of the change. DynamoDB is designed for fast read and write with low latency, optimal for receiving and displaying real time door status on our web application for user viewability. In terms of accessing the camera component of our device, we can use AWS S3 for storage to store logs, images, or video clips. For example, if the ESP32 captures a photo of an object in the garage door's path, it can upload it to the S3 storage, where the URL can then be saved within DynamoDB. We can implement built in S3 lifecycle policies to automatically delete older images to reduce storage costs, which can be beneficial for our stretch goal dealing with object identification using AI.

Security is one of AWS's strong points, offering their own multi-layered security infrastructure to protect cloud applications, IoT devices, and sensitive data. They

follow a shared responsibility model in which AWS secures the cloud infrastructure with developers needing to configure security settings to protect the applications. The automatic AWS Shield for DDoS protection service would be highly beneficial for our IoT system, preventing disruptions caused by flooding AWS with fake requests. If an attacker tries to overload our system with false sensor updates, AWS Shield can mitigate the attack. Additionally, AWS Web Application Firewall (WAF) can be used to block malicious traffic before it reaches our web application. Another useful security measure provided by AWS is the Key Management Service for encrypted data storage. KMS provides automatic encryption for stored data, ensuring that even if an attacker gains access to the database, they cannot read or modify the information without the proper encryption keys. This service can also encrypt data at rest like the camera capture logs in the S3 storage or sensor data stored in the DynamoDB. Additionally, AWS automatically encrypts all data in transit between ESP32 devices and AWS IoT Core using TLS encryption, adding another extra padding of security to ensure user security and also ensuring compliance with security standards.

Storing API keys and authentication credentials in firmware and source code is a major security risk. AWS Secrets Manager provides secure storage for credentials, allowing the ESP32 devices to retrieve access tokens securely without exposing them in source code, as is needed. This helps to prevent unauthorized users from intercepting credentials and also allows the source code to be more manageable. Furthermore, Secrets Manager enables automatic credential rotation, ensuring that even if a security key is compromised, it can be replaced without requiring firmware updates on the MCU, which is an added user friendly perk while also ensuring security from attackers.

AWS offers a variety of pricing models, each designed to accommodate different workloads and usage patterns. The main pricing models include On-Demand, Spot, and Free Tier, just to name a few. Given the nature of our garage door status system, if we select AWS for our cloud hosting, it is important to also select the right model to optimize performance and cost. With AWS On-Demand Instances, we may pay for storage and computation resources as needed with no long-term obligations. This paradigm is flexible and could be optimal for the testing and development stages of our system. However, we would have to ensure we keep track of our usage, as the data can eventually become pricey if the system needs constant data logging from the microcontrollers and users updating and polling data from the database. With Spot Instances, costs are significantly lower than on-demand instances, but AWS is allowed to terminate these instances at any time if demand increases. This pricing model is not ideal since our system requires real-time availability for monitoring and control. If the

instance hosting our microcontrollers was terminated, the system would experience failures. In terms of the Free Tier, AWS provides limited services to support initial development and prototype such as AWS Lambda free for 1M requests/month, IoT core free for 2.25M messages/month, and S3 storage 5GB free for logging MCU events. Of course, once these free thresholds are met, costs would increase accordingly.

Google Cloud Platform

Google Cloud Google Cloud is another major cloud provider known for its high-performance infrastructure, developer-friendly services, and integrations with AI tools like Gemini and Microsoft Copilot. Google Cloud provides scalable solutions for various applications including IoT and web hosting. One of the core services in Google Cloud allows developers to execute code in response to events, making them perfect for automating tasks such as updating Firestore when an ESP32 device sends a status update. Another key service, Google Cloud IoT Core, is designed for managing IoT devices, providing a secure way to connect, manage, and collect data from our ESP32 microcontrollers. IoT Core supports MQTT and HTTP, making it compatible with ESP32's built-in networking capabilities.

Security is managed through Firebase Authentication and database rules, which define access control policies to prevent unauthorized modifications. Additionally, Firebase provides offline synchronization, allowing the system to function even if the device temporarily loses internet access, which was previously one major perk solely attributed to local communications such as Bluetooth or ESP NOW. While RTDB is cost-effective for smaller projects, pricing is based on bandwidth usage and data transfer, which can increase with a high number of parallel running connections.

Google Cloud has security and access controls to ensure that IoT devices and cloud applications are safe. Identity and Access Management (IAM) enables developers to specify fine-grained access rights, guaranteeing that only approved ESP32 devices and users can make changes. Google Cloud IoT Core also requires secure device authentication, which uses public key cryptography (JWT tokens) to authenticate device identity. This keeps unauthorized devices from accessing the system, lowering the possibility of spoofing or tampering.

Another important security feature is VPC Service Controls, which secure sensitive data from Firestore and Cloud Storages from unwanted and unauthorized access. Firestore also supports role-based access control and rules-based security, which enable developers to limit read and write access depending on device identity or user authentication. This is a great measure to

take in securing security, and also ensuring that the microcontrollers have the proper read and write authorization in order to interact with the database as needed. Google Cloud offers DDoS protection with its Cloud Armor service, which is similar to AWS Shield. This prevents malicious actors from flooding the system with requests that impair garage door functionality. Cloud audit logs provide complete tracking of all access and change activity, assisting developers in detecting and responding to security issues.

For our garage door status system, we can use Google Cloud IoT Core to manage our devices connectivity, authentication, and messaging. The MCUs would securely connect to Google Cloud via JWT authentication and MQTT for real-time communication, allowing the MCUs to safely transmit and receive messages, resulting in seamless interaction between the garage door sensor, control logic, and web application. Firestore's real-time sync feature ensures that any changes to the garage door's state are reflected in the user interface in real time. This eliminates the need for polling, which saves power on the MCUs while increasing responsiveness.

Cloud Functions can be used to trigger notifications when the status changes, such as sending alerts to a mobile app or updating a dashboard. Similar to how AWS S3 stores captured camera images, Google Cloud Storage can be used. The ESP32 can upload images when an object is obstructing the garage door and therefore blocking the garage door from being closed. Using Cloud Storage lifecycle rules, old images can be automatically deleted after a set period to optimize storage costs.

Google Cloud also offers multiple pricing models. The Pay-As-You-Go model allows developers to pay only for the resources they use, making it ideal for IoT projects like ours. Unlike AWS, Firestore has a more predictable pricing structure since it charges based on a set document reads, writes, and storage, rather than a set capacity. Google Cloud also offers a free tier, consisting of up to 50,000 Firestore document reads per month and 5GB free Cloud Storage. This makes Google Cloud a cost-effective choice for prototyping and small-scale projects such as ours.

Microsoft Azure

Microsoft Azure is another leading cloud provider offering a vast array of services tailored for businesses, developers, and IoT applications. Azure provides scalable and secure solutions for hosting web applications, managing IoT devices, and processing real-time data efficiently. One of the core services in Azure is Azure Virtual Machines (VMs), which function similarly to AWS EC2 and

Google Compute Engine (GCE). These VMs provide scalable compute power, supporting various workloads, including web hosting, AI processing, and IoT applications. Azure also offers Azure Blob Storage, an object storage service comparable to AWS S3 and Google Cloud Storage. Blob Storage supports high-performance file storage with lifecycle management, making it suitable for managing logs, images, and backups.

For database management, Azure provides Azure Cosmos DB, a NoSQL database optimized for real-time applications. Cosmos DB is well-suited for IoT applications like our garage door system due to its low-latency data access, real-time updates, and automatic scaling. Unlike Firestore, which is optimized for hierarchical document storage, Cosmos DB offers multi-model support, allowing it to work with document, key-value, graph, and column-family data structures. It provides real-time data synchronization between IoT devices and the cloud, reducing the need for polling and saving energy on the ESP32 microcontrollers. This also ensures that any change in the garage door status is instantly reflected across all connected devices.

Another key service, Azure Functions, is Azure's serverless computing solution, similar to AWS Lambda and Google Cloud Functions. Azure Functions allow developers to execute event-driven code in response to triggers, making it ideal for tasks such as updating Cosmos DB when an ESP32 device sends a status update. Azure IoT Hub serves as the central service for managing IoT devices, enabling secure, two-way communication between ESP32 microcontrollers and Azure's cloud infrastructure. It supports MQTT, HTTP, and AMQP protocols, ensuring compatibility with ESP32's built-in networking capabilities.

In addition to Cosmos DB, Azure also provides Azure SQL Database, a fully managed relational database service similar to AWS RDS. If our garage door system requires structured data storage with advanced querying capabilities, Azure SQL could be a viable alternative. However, for real-time synchronization and low-latency performance, Cosmos DB remains the preferred option.

Table 3.10.1: Cloud Server Provider Comparison

	AWS	Google Cloud	Microsoft Azure
Security	Multi-Layered, AWS Shield	Cloud Armor	Pay as you go
Pricing	Tiered pricing: Pay as you go,	Tiered pricing: Per-second	Tiered pricing: Pay as you go,

	spot pricing, on-demand	billing, Free tiers	reserved instances
Databases	DynamoDB	Firestore, RTDB	Cosmos DB
Services	AWS IoT Core, Lambda	Google Cloud IoT Core, auto update Firestore	Google Compute Engine

3.11.1 Database Types

There are several types of databases, each useful in different contexts depending on the nature of the data being stored and how it is accessed. To select the most appropriate database type for our project, we considered factors such as data structure, query complexity, and the real-time system capabilities required by our application.

Relational SQL

Relational SQL databases are designed to handle more structured data within a well-defined schema. It organizes information into tables of rows and columns. This structured approach allows for complex relationships between data points through primary and foreign keys, allowing for advanced queries with high accuracy. The use of SQL as a query language provides powerful data write and read capabilities, making these databases particularly effective for applications where data is more complex – like address, employee id, salary — such as company records. These databases have vertical scaling and have limited support for real-time sync. Offline support requires additional configuring.

Although they are not optimized for real-time updates, it is possible to implement instant updates by polling in relational databases. However, polling increases power consumption and therefore temperature, as the devices will have to constantly be checking for updates to the database.

Non-Relational NoSQL

Non-Relational NoSQL Databases are more flexible in comparison to relational databases, having a varied data structure depending on the type of data being

used, such as documents, wide-column stores, and key-value, just to name a few. Instead of the fixed schema or predefined structure of a relational, these are more dynamic in terms of data organization. This characteristic makes NoSQL ideal for applications that need to handle large volumes of unstructured or rapidly changing information. Query languages vary between the type of database structure used, such as JSON or key-value lookups. This type of database is more suited to applications that implement real-time systems and IoT networks where performance is prioritized over strict relational integrity due to their horizontal scaling.

Due to event-driven updates, the need to poll for data changes was eliminated when using NoSQL databases. This resulted in lower power consumption, which helped increase the battery life of our system. Additionally, lower power consumption contributed to a more stable and cooler operating environment for the device.

It was possible to implement both relational and non-relational databases depending on the type of data stored by our web application. For example, more structured and less frequently changing data, such as usernames, passwords, and object detection logs, could be stored in relational databases, while real-time garage door status updates could be stored in a non-relational database. However, we chose to implement a NoSQL database due to its built-in real-time synchronization features, which served as the backbone and a critical component of our system.

Table 3.11.1 Relational Vs Non-Relational DB

	Relational	Non-Relational
Data Structure	Fixed schema	Flexible schema
Best Use	Structured data like user accounts or logs	Real-time updates, IoT network, event-based changes
Query	Complex queries	Optimized for speed
Scalability	Vertical	Horizontal
Real-Time Sync	Requires polling for instant updates	Built-in
Offline Support	Requires additional configurations	Offline syncing

Power Consumption	Higher	Lower
-------------------	--------	-------

3.11.2 Database Implementation

For database management, Google Cloud Platform has different databases implementing NoSQL. These include: Cloud Bigtable, Cloud Datastore, Firestore and Firebase Realtime Database.

Cloud Bigtable

Cloud Bigtable database is designed mainly for large-scale applications such as IoT data storage and time series analytics. Bigtable organizes data into rows and columns within tables, using unique keys to identify each row. Due to its data structure nature, it is extremely useful in storing massive amounts of structured data. This structure allows for super fast read and write operations, optimized for single-row lookups. Querying in Bigtable relies on primary key lookups and range scans, meaning it lacks the flexibility of SQL-like queries but excels in efficiently handling vast amounts of data. Bigtable does not support offline data synchronization or real-time updates. Instead, it is best suited for applications where large scale data is used over real-time synchronization.

Cloud Datastore

Cloud Datastore uses an entity-based structure where data is stored in key-value pairs within hierarchical schemas. Datastore supports more advanced querying such as indexing and SQL-like queries by implementing Google's GQL. It scales automatically, which helps to ensure data consistency across environments. Additionally, it features some offline support, although limited, in Datastore mode. This allows both mobile and web applications to cache data and sync it back once the connection is restored. Datastore does not include support for real-time updates.

Firestore

Firestore is a document and collections based database, great for IoT projects such as our own due to its low-latency fast reads/writes and automatic synchronization. This makes it ideal for applications where live and accurate updates are crucial. Firestore's built-in real-time data updates ensure that the door's status is instantly reflected across all connected devices and web applications, reducing the need for polling, and therefore the need for the devices to constantly be powered on even when not in use, resulting in a decrease of power consumption, extended battery life.

Firestore Realtime Database (RTDB)

RTDB is also a database that enables applications to store and sync data in real time. However, unlike Firestore, RTDB stores data as a JSON tree rather than a documents based data structure. One of RTDB's key advantages over Firestore is its lower latency. These two share the ability to push real-time updates to all connected clients/devices whenever data changes, eliminating the need for polling. This greatly reduces battery consumption as we can have it so that the MCUs only read and write data to and from the database when a change has been noted by the web application. RTDB is optimized for low-latency performance, ensuring that data is instantly available to users on the web browser. However, it lacks some of Firestore's advanced querying capabilities due to the difference in their data structures, as all data resides in a single location rather than being distributed across multiple areas.

For our project, the Realtime Database (RTDB) efficiently stored and synchronized garage door status updates in real time between the ESP32 devices and the web application. Since the main backbone of our data consisted of simple data variables such as "door-status" and "obj_obstruction" we ultimately used RTDB. We considered either focusing solely on RTDB for all data variables or using both Firestore and RTDB simultaneously for different functions. For instance, Firestore could be used to store values such as user login information, Wi-Fi SSID and password, and notification configurations, while real-time variables like the garage door status were stored in RTDB. Ultimately, we landed on keeping all variables in the RTDB for organization and simplicities sake, as that database met all our project needs.

Table 3.11.2 Google Cloud Platform Database Implementations

	Cloud Bigtable	Cloud Datastore	Firestore	Firestore RTDB
Data Synchronization	No real time	No real time	Real-time	Real-time
Offline Support	None	Limited	Stronger, automatic syncing	Basic
Data Structure	Key-value Wide column	Entities and Properties	Documents and collections	JSON tree
Query	Single-row	Simple,	Advanced,	Simple

		indexing	indexing	filters
--	--	----------	----------	---------

3.12.1 Platform Technology

When developing an application that interacts with a cloud server, effectiveness and cost actually vary significantly between iOS, Android, and web-based APK applications. iOS applications, due to Apple's fairly strict developing environment, often require a higher investment due to things like licensing fees and device specific tools like Xcode. Furthermore, actually publishing an app to the App Store requires the use of a Mac device. The App Store review process can also be time-consuming, delaying updates and increasing maintenance efforts, which could have caused a dent on our project timeline. Further, an additional \$99/year is required for an App Store developer account. Despite this additional cost, iOS apps are known to have great security and smooth performance. In terms of maintenance and app updates, iOS applications require periodic updates that must go through Apple's review process, slowing down deployment. Cloud integration is seamless within Apple's ecosystem but can be restrictive when integrating with third-party cloud services. Performance-wise, iOS apps tend to be highly optimized for Apple devices, ensuring a smooth and responsive user experience with their ecosystem. This could be extremely limiting to not only users – therefore hindering our universal and user friendly goal – but in terms of development, one of the four team members is an Android user.

Comparatively, Android applications generally have lower development costs, as they can be built on various platforms. Additionally, the Google Play Store is less restrictive and therefore more affordable when it comes to publishing an app, sitting at a one time \$25 fee for a publishing account. Android applications are relatively easier to maintain compared to iOS since updates can be rolled out more quickly without a lengthy and strict approval process. Android offers greater flexibility with their own cloud-based interactions, making it easier to integrate with third-party cloud services without restrictions. However, due to the variety of devices that run android, such as Google Pixel or Samsung to name a few, and operating system versions, ensuring consistent performance across different hardware configurations can be a challenge in a grander scope.

Web-based applications eliminate the need for company specific app store distributions and can be updated instantly. This makes them super cost effective when compared to the previous APK options for applications Furthermore, these

APKs can be accessed on both iOS and Android devices via a browser, making them more universally inclusive to our users. Additionally, due to their single codebase and cross-platform compatibility, web based apps provide ease of development. However, web apps can have limited access to device-specific features such as Bluetooth or push notifications, which could impact crucial overall functionality concerning real time updates for our users. The performance is also dependent on the browser connectivity, which could cause a delay in the app and real time access to data. Overall maintenance is the easiest among the three, as updates are deployed instantly on the server, making this perfect for development and update rollouts. Cloud integration is essential for web-based apps, as they rely entirely on these remote servers to function. In the case for our project, this is not a downside, as being able to use any third party cloud servers allows great flexibility with choosing a cloud server provider with lower costs and fast response times without device specific restrictions. However, performance may suffer in areas with poor network connectivity, as the application depends solely on the browser's ability to retrieve and process data for real time updates.

Table 3.12.1 Platform Technology Comparison

	iOS	Android	Web-Based
Development Cost	High, requires Mac, Xcode, fees	Lower	Lowest
Publishing Cost	\$99/year	\$25 flat fee	No app store distribution
Cloud Server Integration	Seamless with Apple only	Easy integration with third-party cloud services	Fully dependent on cloud-based interactions
App Store Restrictions	Strict and time-consuming	Faster approvals, less restrictive	N/A
Cross-Platform Compatibility	Limited	Limited	High
Performance	Optimized for Apple devices	Varies based on hardware	Dependent on browser performance

Security	High	Moderate	Moderate
Access to Device Features (Bluetooth, push notifications...)	Full access	Full access	Limited
Maintenance	Requires app store approval	Easier than iOS but still requires app store approval	Easiest with instant updates

3.13.1 Communication between Microcontrollers

Based on the communication protocols we chose, data transfer flow was affected, influencing how quickly data moved from one microcontroller to another. This introduced latency between devices and impacted response times. In addition to standard wired communication options, microcontrollers such as the ESP32 come equipped with built-in Wi-Fi and Bluetooth modules, enabling them to act as local servers without requiring additional external hardware. To achieve faster communication times, we utilized these built-in modules for local control, eliminating the need for cloud communication during local operations. This approach also reduced installation time since the user was present and connected to their home Wi-Fi network during the initial device setup.

To further minimize communication latency, we leveraged the Wi-Fi and Bluetooth capabilities of the microcontrollers. Our system consisted of two microcontrollers: the primary MCU responsible for processing input from the reed switch sensor and camera module, and a secondary MCU that, through a motor controller, managed the servo motor to open and close the garage door. Since the two MCUs were placed within the same room but separated by some distance, we tried to establish Bluetooth communication between them. This setup in theory should have produced low latency and allowed the garage door to be operated locally even without Wi-Fi connectivity.

While wireless communication provided flexibility and simplified installation, we also considered a standard wired communication approach for increased reliability and security. Wired connections such as UART or I2C between the Main MCU and Servo MCU ensured a stable, interference-free communication channel. However, wiring introduced physical constraints, limiting device placement due to the need to run cables between MCUs. This made installation

more complex, increased setup time, reduced user-friendliness, and restricted physical device placement within the garage. Given our design, it was crucial for the Servo Motor MCU to be located near the garage button, while the Main MCU monitored the garage door path to detect object obstruction. Although wired communication reduced latency and eliminated wireless connectivity concerns, the physical limitations imposed were unavoidable without altering how the device interacted with the garage.

Based on the results of our testing, we found that with the primary device connected to Firebase and the secondary device connected via Bluetooth, it was not possible for the primary device to maintain both communication channels simultaneously due to the limitations of the ESP32. Furthermore, when relying solely on Bluetooth for communication, the latency for servo movement exceeded 20 seconds, if any response occurred at all. To resolve these issues, we implemented a solution in which both microcontrollers connect to the cloud host Firebase, which acts as the mediator between the devices. All communication is handled by updating database variables in Firebase; the web application listens for these changes and then commands the corresponding device to react accordingly. This approach significantly improved responsiveness and reliability in coordinating actions between the devices. As such, despite its higher power consumption compared to the other local communication options, the lowest latency results aligned with our project specifications and goals along with high reliability and no extra user set up.

Table 3.13.1: Local Communication Comparison

Comm Method	Latency	Power Consumption	User Setup	Reliability
Bluetooth	Low, ideal for real-time communications	Lower than Wi-Fi	Pair the 2 MCUs for direct wireless communication	Reliable through Wi-Fi shortages;
Built-In Local Server MCU	Higher than Bluetooth	Higher than Bluetooth as it is always active	Connects to home network, no	Reliable through longer distances, dependent on Wi-Fi stability

			additional pairing	
Wired	Lower	Lowest due to direct electrical signals	Increase installation time due to wiring	Most reliable communications
Cloud Host Mediator	Lowest	High	None	Reliable

3.14.1 AI-Based Object Detection

For our DoorSure system, we needed a way to detect if something was in the way before closing the garage door. Since we're using the ESP32S3, we looked into a few lightweight object detection options that could work with our limited hardware. We found three leading platforms: TensorFlow Lite, OpenCV, Edge Impulse and Google Gemini. Each one has its own strengths and drawbacks, so we spent time looking at what they offer and how easy they are to set up.

TensorFlow Lite is a popular lightweight framework from Google that's made for running ML models on mobile and embedded devices. The advantage is that it supports offline inference, runs on lower-powered systems, and has access to a variety of pre-trained models like MobileNet. But once we dug deeper, we realized that the version made for microcontrollers, called TensorFlow Lite for Microcontrollers (TFLite Micro), is very limited. It can only handle very simple image classification models, and anything more complex - like very limited. It can only handle very simple image classification models, and anything more complex - like object detection with MobileNet SSD - does not fit the memory of the ESP32-CAM. On top of this, the setup and training process requires a lot of extra tools and know-how. In the end, we decided not to use TensorFlow Lite because it doesn't meet our needs in this project.

OpenCV is a really powerful computer vision library used for everything from face detection to advanced image processing. It has tons of features and can definitely handle object detection, but not on the ESP32-CAM. OpenCV is meant for devices that have an operating system, more RAM, and a lot more processing power. The ESP32-CAM just isn't built for that. We'd have to send

image data to a Raspberry Pi or another external device to process it, which adds extra cost and complexity of the project. So even though OpenCV is a solid tool in general, we decided against using it because it does not fit our low-power setup.

Edge Impulse is made specifically for embedded devices and edge AI applications, and it works really well with microcontrollers like the ESP32-CAM. What we liked most about it was how easy it is to use. One can collect data, train your model, and deploy it all through their web interface, without needing to know a ton about machine learning. It even optimizes models automatically to fit on smaller devices. It does require internet access during training, the final model runs completely offline on ESP32-CAM. During the testing, we found out that Edge Impulse's model has many limitations. It had difficulty detecting when the object is far from the camera, and the sunlight affected the performance as well.

Google Gemini is a cloud-based AI model that became our final choice. While it doesn't directly run on the ESP32S3, we integrated it by a hybrid approach: The ESP32S3 captures an image and uploads it to Firebase Storage. A separate script sends the image to Google Gemini via API for analysis and receives a natural language result. This bypassed all heavy computation and memory limitations of the ESP32S3. Gemini's ability to handle complex object detection and respond with a natural language made it a perfect fit for our use case.

Table 3.14.1: AI-Based Object Detection

Feature	TensorFlow Lite	OpenCV	Edge Impulse	Google Gemini
ESP32-CAM Compatibility	Limited (TE Lite Micro too constrained for object detection)	Not compatible (requires more RAM/OS than ESP32-CAM has)	Designed for microcontrollers like ESP32-CAM	Not directly run on ESP32-CAM; requires cloud processing
Offline Inference	Yes (for simple models)	No (requires external processing)	Yes (after training)	No (cloud-based inference only)
Model Complexity Support	Only very simple classification models	Not on ESP32-CAM	Automatically optimized for microcontrollers	Handles complex object detection and natural

				language queries
Ease of Use / Setup	Requires many tools and technical knowledge	Complex setup and external hardware needed	User-friendly web interface; minimal ML knowledge needed	Very easy (web API calls; no model training or embedded setup required)
Training Requirements	Requires external tools and expertise	External training required	Online training via browser	No training needed; uses pre-trained foundation models
Cost & Resource Overhead	Yes	No	Yes	Requires internet + API call (compute handled by Google)

3.15.1 Embedded Firmware

To bring our hardware to life, we needed the right development environment to program the MCUs. There are a couple of solid options for writing and deploying firmware. We focused on two main features: ESP-IDF and Arduino-based firmware. ESP-IDF is the official development framework from Espressif. On the other hand, Arduino-based firmware offers a very different experience.

ESP-IDF is the official SDK provided by Espressif, and it gives full control of the ESP32's features. It allows highly optimized applications with low-level functionality. It can be useful for advanced configurations and performance tuning. However, it also comes with a learning curve, embedded C and command-line knowledge is needed. It can take more time both setup and troubleshooting, it's not ideal for fast prototyping.

On the other hand, Arduino-based firmware uses the Arduino IDE, offering a more beginner-friendly environment. It supports a massive number of libraries and examples. The community support is also huge, making it an easier solution. Additionally, it allows us to test features without spending too much time on setup. While it may not provide the same level of low-level control or optimization like ESP-IDF, it has more than enough for projects like ours.

Table 3.15.1: Embedded Firmware

Feature	ESP-IDF	Arduino-Based Firmware
Official Status	Official SDK from Espressif	Community-supported platform
Control & Optimization	Full low-level control; great for performance tuning	Limited low-level access; less optimized
Ease of Use	Steeper learning curve; requires knowledge of C and CLI	Beginner-friendly; uses Arduino IDE
Library & Example Support	Decent, but less user-friendly	Extensive libraries and examples available
Community Support	Smaller, more technical community	Large and active community
Setup Time	Longer setup and configuration time	Quick setup and testing
Prototyping Speed	Slower; better for production-ready firmware	Faster iteration; ideal for prototyping

3.16.1 Notification System

As the chosen platform for DoorSure is web-based, email push alerts seemed to make the most sense to incorporate in our system as opposed to text messages or mobile banners. Additionally, as the user's email was already being captured during sign up to allow access into the Firebase RTDB, this was an intuitive decision.

To reliably notify users by email, we needed a backend that could send alerts quickly and flexibly. Two main options emerged: Firebase's native email capabilities and third-party services like SendGrid. We focused on two core priorities: ease of developer integration and flexibility with recipient email domains. Firebase's email system is tightly integrated with Firebase

Authentication. It handles verification and password reset emails out of the box, which is great for basic account workflows.. While it technically can send to any valid email address (not just Gmail), it lacks customization, doesn't support styled templates, and has no built-in analytics. More advanced usage like sending transactional emails or *alerts* requires Firebase Extensions or writing your own Cloud Functions.

SendGrid, on the other hand, is purpose-built for email. It supports sending to any address—Gmail, Yahoo, Outlook, and more—without limitations. Implementation is simple: just call their API with your message and recipient. SendGrid also provides a powerful dashboard with delivery tracking, open/click stats, suppression management, and customizable dynamic templates. For developers, it removes complexity and speeds up deployment.

While Firebase could handle some basic email use cases, SendGrid provides more power, fewer limitations, and better control over how, where, and to whom emails are sent. For a system that needs reliable notifications to any user email and easy integration, SendGrid is clearly the better fit. Its free tier is plenty enough for our system and is perfect for sending alerts to the user.

Table 3.16.1: Notification System Comparison

Feature	SendGrid	Firebase Cloud Email
Set up complexity	Simple with API key	Built-in support for auth. Emails; extensions need setup
Custom email templates	Full control	Limited with Firebase Auth, expand with extensions
Free Tier	100 emails per day and scalable	Limited to auth emails, no bulk or transactional emails

Chapter 4. Standards and Design Constraints

When starting a project, standards and design constraints should be kept in mind at all times to ensure that our project not only meets requirements and

standards but also functions within the constraints enforced on us such as cost and time duration. Standards document the characteristics of the product such as dimensions and most importantly safety.

4.1 Standards

Standards can be found in every aspect of our lives as simple as modern conveniences such as USB cables that allow devices to connect to each other, not only does these standards make devices work together but also share guidelines to make these technologies more accessible and reliable, throughout the next pages we explore relevant standards to our project, we have considered the latest guidelines to ensure optimal performance, by integrating the up to date standards we would lay the foundation for a solid project that we can innovate and advance to reach a market ready product capable of competing with the competition.

4.1.1 PCB Design

First of all, we need to define PCB, according to IEEE, PCB is a printed circuit board which is used to interconnect electronic components constructed of 1 layer or possibly multiple layers of dielectric and conductive materials that are designed to form conductive paths, in our project, we used JLC PCB, a trusted manufacturer with more than 9 million PCBs produced per year, to ensure our design is implemented up to the following standards.

For the PCB design, we are going to take in consideration both the professor's standards from the University of Central Florida and IEEE standards, by integrating both guidelines we ensure that our PCB design meets the global guidelines and ensure that our design would be approved locally by UCF, the university guidelines are incredibly useful as they consist of previous senior design experience which would make the implementation of our PCB in our project easier and robust.

For the professors' guidelines, we would need to use capacitors no smaller than the 0802 size and avoid using any IC with smaller pins spacing with 0.5 mm LM25xx or LM26xx regulators were highly advised as they are more robust and easier to use more than newer regulators from Texas Instruments, since they can be very sensitive to board layout citing previous senior design projects' experience. Despite the ideal regulator choice we chose the AP630xx regulators for lower quiescent current and smaller PCB layout footprint which is more convenient for us since we are implementing them as daughter boards. Having the regulators as daughter boards would make swapping them out easier in that case of the voltage regulator being shorted out. Through guidance, installing

LEDs on the voltage regulators were recommended in order of making testing and debugging easier, furthermore, in order for our design to be approved by the professors we would need to include at least one MCU, a power supply unit and several peripherals integrated with the MCU, also we would have to replace the development boards with customized boards.

We used the IPC-2221A which identifies generic physical design principles and is supplemented by various sectional documents including IPC-2222 and IPC-2223 which would be illustrated in the following figure:

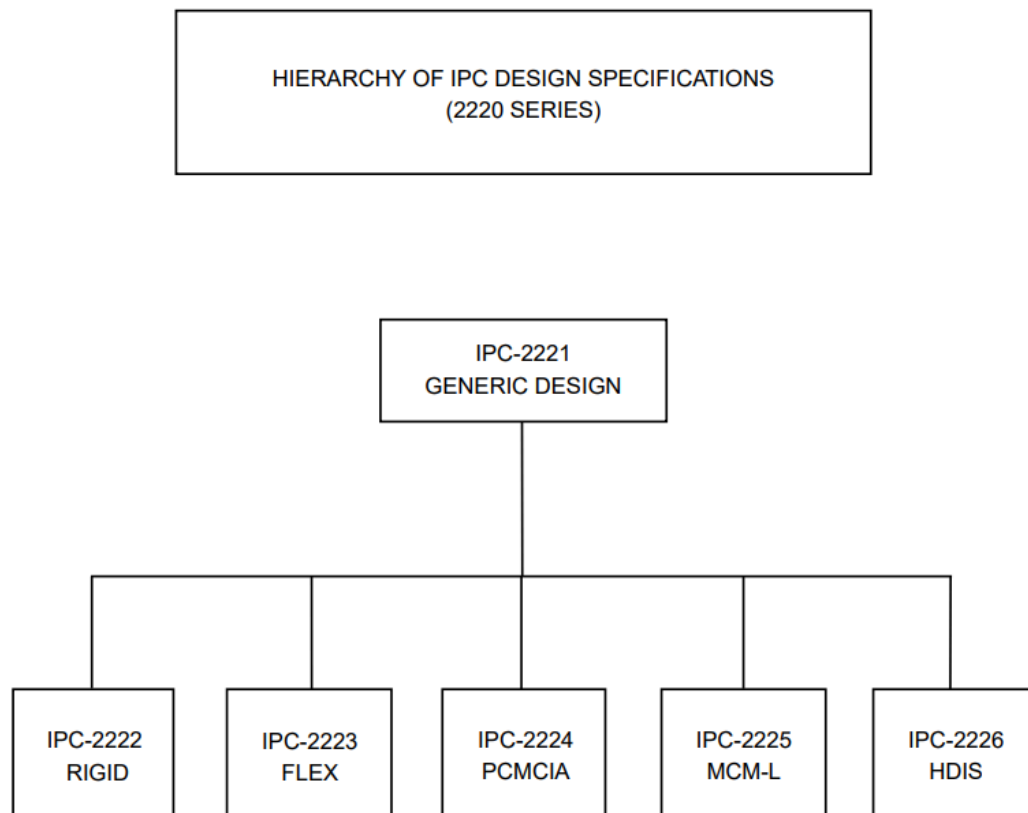


Figure 4.1.1 Hierarchy of IPC Design specification (2220 series)

This standard is intended to provide information on the generic requirements for organic printed board design.

More standards are provided by JLC PCB which we adhered in order to make our circuit realistic and doable, those standards include hole size and drill diameter specified below:

- Drill parameter for multilayer: 0.15-6.3 mm
- Min. via hole size/diameter for multilayer: 0.15 mm hole size/ 0.25 mm via diameter
- PCB thickness: 0.4 - 4.5 mm
- Via hole to hole spacing: 0.2 mm
- Via hole to track: 0.2 mm
- Copper clearance from routed board edges: ≥ 0.2 mm
- Minimum trace width: 0.16 mm

By implementing these guidelines by JLC PCB, the professors of the University of Central Florida and IPC- 2221A, we ensured that not only our design was doable and realistic but also followed all recent guidelines and up to standards.

4.1.2 Wi-Fi

As the DoorSure devices used wireless connectivity to interact with the web application, it was essential to ensure compliance with established wireless communication standards. One primary method for wireless interaction was Wi-Fi, which follows the IEEE 802.11 standard. Introduced in 1997, IEEE 802.11 was the pioneer of 2.4 GHz Wi-Fi and has since served as the foundation for future Wi-Fi infrastructures adopted by many international industries. The protocol has evolved over time and is currently at IEEE 802.11ax, or Wi-Fi 6, supporting data rates up to 9.6 Gbit/s and improved performance in dense data environments.

Both the ESP32 Development Board and the ESP32 Camera Development Board used in our project fully complied with the IEEE 802.11 b/g/n standards. To provide context, IEEE 802.11b (Wi-Fi 1), introduced by Apple in 1999, achieved higher data rates and reduced interference from devices such as cordless phones and baby monitors. It incorporated various modulation schemes, including direct-sequence spread spectrum and complementary code keying (DSSS/CCK), but was limited to indoor ranges of approximately 38 meters and outdoor ranges of 140 meters.

Following this, IEEE 802.11g (Wi-Fi 3) increased data rates up to 54 Mbit/s within the 2.4 GHz frequency band. Lastly, IEEE 802.11n (Wi-Fi 4) supported both 2.4 GHz and 5 GHz bands, enabling data rates up to 600 Mbit/s. This advancement allowed wireless local area networks (WLANs) to effectively replace wired networks, leading to the widespread adoption of Wi-Fi.

Since both development boards supported up to Wi-Fi 4 (IEEE 802.11n), they were able to reliably connect and update the database and, consequently, the web application with sufficient data rates. This standard provided a balanced

combination of power efficiency, speed, and range, making it widely used in IoT applications due to the broad compatibility of IEEE 802.11b/g/n protocols.

Below is the evolution of IEEE standards for WI-FI technology.

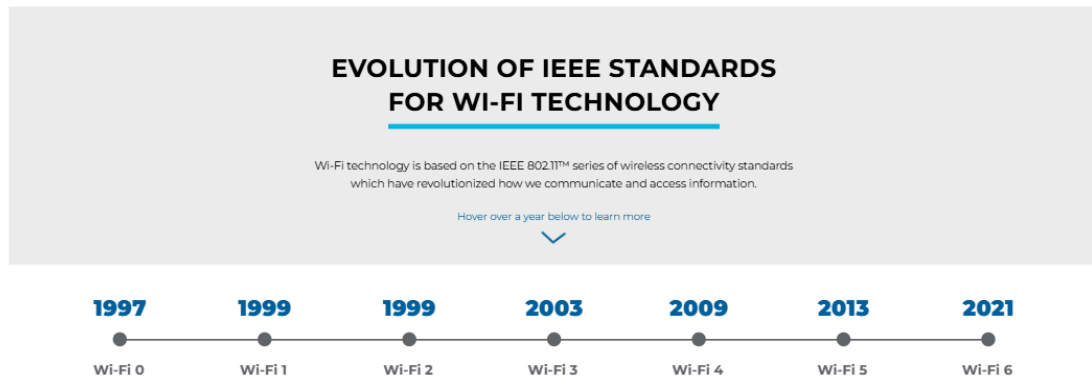


Figure 4.1.2 Evolution of IEEE Standards For WI-FI Technology

4.1.3 Bluetooth

Another way the DoorSure devices could connect with each other was through Bluetooth. This standard enables wireless communication over short ranges, operating in the 2.4 GHz unlicensed industrial, scientific, and medical frequency band. Bluetooth provided an alternative connection method for the devices to communicate with the web application in cases where Wi-Fi connectivity was unavailable, allowing greater flexibility in device linking. The ESP32 Development Board we chose supports both Classic Bluetooth and Bluetooth Low Energy (LE), a subset of the Bluetooth 4.0 stack. Bluetooth LE emerged as an appealing option due to its low power consumption, making it well-suited for sensor data collection. For example, the ESP32 Camera Development Board used Bluetooth 4.2, which offered additional features such as the Internet Protocol (IP) profile and enhanced security, making Bluetooth LE a competitive candidate among available low-power communication technologies.

Bluetooth version 4.0 included multiple useful features and modes, allowing unidirectional communication between two or more Bluetooth devices using advertising events. This approach enabled communication without the need to establish a bonded connection, resulting in increased energy efficiency. However, the absence of a handshake process in this mode made it vulnerable to security threats. Additionally, Bluetooth LE broadcast was an unreliable communication method because the transmitter could not confirm whether the data was received by any observers. Bluetooth 4.2 addressed many of these

concerns by offering secure Bluetooth communications and additional capabilities such as Bluetooth Smart, which introduced features attractive for IoT applications. Despite being relatively new at the time, Bluetooth 4.2 still had some unresolved issues, which we considered during our research and testing.

There have also been efforts to connect Bluetooth LE devices directly to the internet, but there remains no widely accepted standard for transmitting IPv6 packets over Bluetooth. Although Bluetooth Smart represents an interesting and promising technology, we identified it as potentially useful as a supplementary communication channel in future iterations of the project. However, since it was still under development and presented certain challenges, we decided to use Wi-Fi connections exclusively for this project due to their widespread support and reliability.

Based on our testing, we ultimately discontinued the use of Bluetooth for device communication due to high latency and connectivity limitations. Nonetheless, if we were to continue developing the project beyond this course, we would implement Bluetooth for configuration purposes during setup—specifically, to allow dynamic entry of Wi-Fi SSID and password—rather than hardcoding these credentials, thereby improving user experience and flexibility and security.

4.2 Design Constraints

Every project has constraints—an inevitable reality that affects every aspect of the work. The following constraints played a significant role in shaping our project, from the initial brainstorming phase through testing and integration of the overall system. Constraints not only define the boundaries of what can be achieved but also motivate innovation and creative problem-solving techniques to optimize our approach. They pushed us to find resourceful ways to work within these limitations, whether it was managing pressure to meet tight deadlines or overcoming technical challenges encountered along the way.

Ultimately, each of these constraints guided us to refine our ideas and successfully build the project within the boundaries they established.

4.2.1 Time

In Senior Design, time was the main constraint shaping every aspect of the project. With a limited timeframe of two academic terms—one of which was the summer semester, offering even less time—every decision had to be made with efficiency in mind. This meant we needed to be wise in choosing components

and constructing the final design to avoid repeating steps. Maintaining open communication among team members and setting solid deadlines for each task helped us stay on track. Since extensions were not granted on official deadlines, we set our own internal deadlines early on to review documents and ensure that submitted work was of the highest possible quality, minimizing the amount of rework needed after receiving feedback.

These early internal deadlines not only helped maintain our progress but also taught us accountability, as we all depended on one another to succeed. Following recommendations from Senior Design professors, one team member stepped up as leader to coordinate deadlines and keep everyone moving at the same pace. Falling behind was not an option given the severe repercussions, including the risk of having to retake Senior Design, which motivated the entire team to stay punctual.

Due to the limited project timeframe, we adopted rapid prototyping and testing cycles. From the outset, we organized parallel task execution—Computer Engineering (CPE) students worked on software while Electrical Engineering (EE) students focused on hardware. This approach allowed us to integrate the components more efficiently later on. Although this was not the ideal working environment, motivation and guidance from our professors provided the push we needed to finish on time.

Ordering parts and designing the PCB were among the most impacted aspects due to time constraints. Early in the project, we experienced delays of more than a week when ordering components from Amazon. To offset and prevent future delays, we ordered backup components for every critical part and planned to fabricate multiple PCBs as backups for our main board. Additionally, following a professor's recommendation, we added LEDs in multiple parts of the PCB for easier debugging and separated the voltage regulators onto dedicated PCBs, connecting them to the main board via pin headers.

Time constraints required us to take further precautions, such as working during spring break and any available time to ensure a fully functional prototype was completed by the end of Senior Design I. This would give us a head start in Senior Design II and help avoid delays caused by component or PCB deliveries, which typically take more than a week. These precautions were absolutely necessary within the given timeframe.

Although the time constraint heavily impacted our project, we recognize that in professional environments, limited timeframes are a daily reality. The urgency to learn new skills and deliver results within tight schedules helped prepare us for the workforce after graduation. This intense pace demanded flexibility and

adaptability, especially since most of us were balancing other courses alongside Senior Design. Without these qualities, the risk of rushing the project and insufficient testing and debugging could be devastating. Therefore, adaptability and flexibility were key not only to passing the class but also to producing a working project that serves as a stepping stone toward our careers.

By recognizing the trade-offs between time constraints and final product quality, we embraced the values of time management and open communication. Time constraints motivated us to work efficiently toward our end goal: a fully functional project with multiple backups that would build skills valuable even after graduation. This tight schedule also reinforced the project's reliability by minimizing the risk of redoing completed parts. With these precautions and vision in mind, we were able to meet all deadlines and produce a successful final product.

4.2.2 Economic Constraint

Another main constraint for our project is the economic constraint. Since we do not have a sponsor, we are relying solely on our own funds to cover expenses, which imposes a strict budget. Our team agreed that each member can contribute up to \$200, making our total budget \$800. Although this budget appears higher than the total cost listed in the bill of materials, we had to order components with faster delivery options to avoid further delays experienced since the beginning of the project. This necessity also encouraged us to order extra components as backups so that each team member could work independently from home without waiting for critical parts from others, ensuring continuous progress and minimizing downtime.

Moreover, because we plan to have multiple backup PCBs with expedited delivery to guarantee uninterrupted progress throughout Senior Design I and II, additional funds are required. Balancing the extra costs of faster delivery and spare components within our limited budget necessitates careful and strategic spending, with efficiency being a key consideration.

Since our team consists of four undergraduate students, cost is a critical factor in every decision we make. With limited funds, we must constantly weigh the trade-offs between quality and cost. For example, when selecting microcontrollers, we chose the best option as outlined in the microcontroller comparison section because it is the main component of our circuit.

Specifically, we selected the ESP32 due to its numerous built-in capabilities, such as integrated Wi-Fi and Bluetooth modules. These features enable it to function as a local server without requiring additional hardware, making the

ESP32 an ideal balance between quality and cost. Its built-in modules save us the expense of purchasing separate communication components.

On the software side, the ESP32's support for ESP-NOW is another key advantage. This protocol allows wireless communication between multiple ESP32 devices, facilitating efficient and reliable connections. By leveraging ESP-NOW, we can enhance system performance and reduce the need for costly precautions against communication instability.

We are also mindful of existing commercial products with similar capabilities, as discussed in Chapter 2 on existing products. Although our project is not intended for mass manufacturing or sale, considering the price points of these comparable products helps us establish a reasonable and competitive cost range. We want to ensure that our design remains viable and advantageous compared to market alternatives.

Overall, the economic constraint significantly influences our project, driving us to be resourceful in utilizing available resources. We must remain conscious of this limitation when selecting components, carefully balancing performance and cost. Additionally, managing delivery options is crucial to maintaining our timeline—finding the right balance between slower, cheaper shipping and faster, more expensive alternatives is key to avoiding delays.

4.2.3 Safety Constraint

Safety is always the number 1 priority when implementing any project, especially one involving an automated system with moving parts like our garage door project so we need to have a safety constraint and implement fail safe mechanisms in case of a network failure we have implemented an option to connect to your device using bluetooth module which is already built in the esp 32.

A failure to consider safety could have severe consequences including injury or property damages, for example if there is a long delay in door opening or closure it may cause unauthorized access to your property or even theft, so we need to have rapid response especially for door closure, moreover the accuracy when checking the door status needs to be incredibly high ($\geq 95\%$ Accuracy) to guarantee that the system actually reflects real time state of the door, also, a low notification delay (≤ 2 seconds) is critical in providing near-instant security alerts, which enable the user have the peace of mind in terms of getting a notification whenever an anomaly is detected, the user can also set multiple anomalies, for example, door opened late at night or leaving the door open for a long period of

time, a failure in controlling the door can result in the garage door closing accidentally which can cause damages.

Most importantly, a failure to consider safety can result in injury, which is why intensive testing and debugging are incredibly important in this project, this trade off between testing, cost and time causes us to carefully balance between doing intensive testing and cost within the time constraint, to achieve this balance, we have to divide up tasks among team members and working on individual parts and then integrate them all together, although this approach be effective in terms of time constraint while doing intensive testing but have impacted our budget.

Safety is the only constraint that we can absolutely not compromise on, no matter the trade off between cost and time, ensuring the safety for our project's users is the absolute priority, that affect our choice of components which have lower tolerance which would be more expensive, choosing the camera for instance, also require us to choose a high quality camera with the option of viewing live feed (advanced goal).

Safety constraint affected our design from the very beginning, for example, we chose a camera (720+p) with AI capabilities instead of an ultrasonic sensor to detect objects in way of the garage door and possibility identifying them (stretch goal), although this approach have some privacy concerns and high power consumption compared to other sensors, but the camera approach offer more visual information that increase security and justify this trade off, this approach give us a significant advantage compared to other market examples such as the Chamberlain MyQ Smart Garage Door Sensor and the Genie's Aladdin connect, positioning of the camera be crucial in terms of detecting the objects in garage door way, not only incorporating this camera with Ai capabilities improve security measure but give us the advantage over our competition to improve system accuracy and maintain the user trust in our product.

4.2.4 User friendly Constraint

The user-friendly constraint focused on designing our system to be accessible and comfortable for all users. Since the core goal of our project was to provide peace of mind, we prioritized minimizing complexity across both the software and hardware components.

On the software side, we developed a streamlined web application interface that facilitated easy navigation and intuitive interaction. Clear labeling and a step-by-step usage guide were incorporated to assist users in understanding and utilizing all features. We also conducted usability testing and gathered user

feedback, which played a crucial role in shaping a more accessible and refined experience for all intended users.

For the hardware, we emphasized ease of assembly and long-term reliability. Our design ensured that the system would be user-friendly during installation and require minimal maintenance. We selected high-quality components, such as a camera capable of high-resolution video streaming, to support accurate object detection while maintaining dependable performance under varying environmental conditions.

As the system remained in its prototype stage, we prioritized flexibility and ease of use to ensure that the product would be convenient for daily use. We understood that even though our project was not intended for commercial release, aligning our design with — and even exceeding — market standards for user-friendliness was essential. Products like Chamberlain MyQ and Genie's Aladdin Connect served as benchmarks that informed our design improvements.

Ultimately, the user-friendly constraint significantly influenced both the software and hardware aspects of the system, making it more accessible and competitive in comparison to existing market solutions.

4.2.5 Manufacturability Constraint

Although our project was not intended for large-scale manufacturing or commercial distribution, manufacturability constraints still played a significant role in shaping our design from the very beginning. The initial concept originated from the personal needs of one of our team members, which inspired us to develop a custom solution that offered peace of mind. While similar products exist on the market, our prototype aimed to integrate the best features—such as affordability, live video streaming, and AI-based object detection—into a single, optimized system.

The manufacturability of our design was influenced by the limitations of our academic lab environment and the tools available to undergraduate students. Additionally, since our printed circuit board (PCB) was ordered from JLCPCB, we were required to adhere to their manufacturing capabilities, including specific constraints on trace widths, component sizing, and board dimensions. To ensure our PCB design was both functional and compliant with these constraints, we consulted with senior design faculty to verify that our design met production and course requirements before placing our order.

Working within these manufacturability constraints provided a valuable, real-world experience. In industry, similar challenges such as limited equipment, fabrication tolerances, and supply chain issues are common. By navigating these constraints, we gained practical insight into the importance of designing with manufacturing feasibility in mind from the outset.

The lessons learned from this process will be especially beneficial if a team member chooses to continue developing the project or pursue a startup venture in the future. With awareness of both technical and competitive factors, we approached the prototype design with the goal of outperforming existing solutions in reliability, functionality, and manufacturability.

In conclusion, manufacturability constraints helped us balance innovation with practicality, reinforcing the need to align our design with real-world fabrication capabilities and encouraging us to refine and optimize our system throughout development.

4.2.6 Sustainability Constraint

While the sustainability constraint was not as critical as the safety or time constraints, it still played a role in shaping our project—particularly with future plans to develop it into a market-ready product. We needed to account for the wear and tear on components operating in a garage environment subject to fluctuating environmental conditions, such as temperature and humidity. The expected operational temperature range was -20°C to 60°C , with humidity levels between 10% and 90% RH.

To address this, we selected durable components that could withstand such environmental stressors. This helped reduce the frequency of maintenance and component replacement. While some parts—like batteries—would inevitably need to be replaced, we extended battery life by minimizing the system’s overall power consumption to between 2 and 6 watts. This directly reduced the need for frequent maintenance and contributed to long-term sustainability.

As part of our senior design, we constructed a scaled prototype of a garage to demonstrate the system’s capabilities through a user-friendly interface. Designing the prototype ourselves allowed us full control over structural decisions, integration of safety features, and performance optimization. This hands-on approach served as a proof of concept and provided a foundation for a future product that could be adapted for any standard garage door system.

We also considered the wear and tear of the prototype itself, as we intended to simulate long-term operation through repeated use within a short timeframe.

This posed a unique challenge since our team lacked prior experience in mechanical design and stress testing. To overcome this, we consulted faculty members with relevant expertise to ensure the prototype's reliability under simulated conditions.

Chapter 5. Comparison of ChatGPT with other Similar Platforms

With the increasing popularity of Artificial Intelligence, our team has benefitted from using ChatGPT as a resource to guide us throughout our project. From searching for bugs in our code and proofreading our technical documentation, to researching hardware and software technology, ChatGPT has provided insights that have cut the research period almost in half, allowing the team to focus on the hardware and software. By utilizing AI-powered assistance, we have been able to enhance our research, improve efficiency, and ensure a more well-rounded approach for our project.

Case Study 1

Prompt: "What kind of motor would be efficient to use to push a garage door button?"

When inputting this question into ChatGPT, the responses received were mostly accurate. The model listed out 3 of the motors that could be implemented in our project, linear actuators, solenoids, and servo motors. We chose the servo motor as our choice early in our project through our own research and all of the characteristics shown were accurate in its results. However there are a few caveats. ChatGPT only lists these three models in its results and no other motors like DC, AC, their brushed or brushless counterparts, or stepper motors are listed. In addition only their general specifications are listed along with their main strengths, no disadvantages or limitations are listed at all along with any specific engineering specifications. Cost is not listed which is a major factor in deciding which component to get in order to cut costs.

Its main recommendations at the end were either a linear actuator or a servo motor. To conserve costs, the servo motor would be the better choice as linear actuators are expensive to acquire compared to getting servo motors. While ChatGPT does help in deciding which part would be preferable to get to press the garage door button, the information generated is left to be desired.

When applying the same prompt into Gemini, their response is more detailed than ChatGPT's, as each component has pros, cons, their ideal form in use of real-world applications, along with major key considerations in case if the part they suggested is not what we want to choose. While Gemini's response is good, the problems that are explained in ChatGPT's response can also be applied to Gemini as well. There is no mention of cost for all of the components and the list of components are also limited to 3. Servo motors are not mentioned as well, which adds to the confusion on Gemini or even ChatGPT added more than 3 motors to their list.

Microsoft Copilot is even worse than both models as only two motors are listed (which were small DC motors and linear actuators), while giving them general descriptions on why to use them. They do not break down the advantages and disadvantages of each component or the cost of obtaining them.

Case Study 2

Prompt: "For a system that requires low power when in use and when it's idle, along with long battery life, what size of battery should I consider using? (AA, AAA, 9V, etc.)"

This prompt was to make sure the model would properly reply with the battery size types, along with the purpose of what the system should do in respect to power operations. In the case of ChatGPT's response, their list of battery size types is mostly accurate to the results found in Section 3.2.1.2, along with major key considerations. The prompt only lists the kinds of batteries that are commonly used while this report also uses batteries that are uncommon, such as 123, CR2, and N batteries. Their list does not go over if there are alkaline or lithium versions of the respective battery, along with not going into detail over the pros and cons of the type. At the end the model suggests different kinds of batteries depending on the situation, such as making the system ultra-low power, size and weight constraints, and rechargeability.

For Gemini's response, the style of their prompt is similar to ChatGPT's. Both include a limitation for 9V batteries in terms of power efficiency and the recommendations are similar. One interesting thing to note on Gemini's side is the inclusion of a lithium battery known as "LiFePO4". Although this is useful to know, we intentionally did not include other lithium battery types as we wanted our system to use replaceable batteries for the sake of convenience.

Like the first prompt, Copilot has the worst response of the three, only mentioning the three battery size types that are arguably the most commonly used. Only AA, AAA, and 9V are listed, while the other two models list out more of a variety like button cells and other lithium type batteries. Each battery listed specifies their capacity, size, lifespan, and use cases, except that 9V batteries only list their capacity and use cases for some reason. This inaccuracy within their response shows the unreliability that AI models can have and why it is important to seek the direct sources for research instead of overly relying on AI.

Case Study 3

Prompt: “I’m using ESP32-CAM to do an object detection project. I need the camera to recognize objects in the way when the garage is open. What are my choices?”

ChatGPT offered me opinions to choose from: TensorFlow Lite for Microcontrollers, Edge Impulse, OpenMV, and YOLO-based External Processing. On the other hand, Deepseek gave me seven opinions, divided into different implementation methods. They were both very helpful and provided step-by-step guides on how to accomplish it, even giving me recommendations on which one works in what scenarios.

These prompts bring insight to a few things. For starters there might be moments where the prompt needs more detail in order for the response to be plausible to believe. Otherwise the response will be somewhat limited. Plus, this shows that AI response models cannot be overly relied on because their results can be made out of random jargon and have minor mistakes in providing facts to components.

Case Study 4

Prompt: “I’m developing an accompanying application to a device that needs to work across multiple platforms. I want to understand the differences in building an APK for Android vs iOS vs web based.”

When typing the prompt into ChatGPT, the responses were given in a very structured manner. It provided me with three primary approaches: native development, cross-platform frameworks, and progressive web apps (PWAs). It correctly outlined the strengths of Android’s Java/Kotlin, iOS’s Swift, and cross-platform solutions like Flutter and React Native.

It gave me detailed information on Apple's security policies, but failed to cite any resources. When prompted to cite where the information was found, the engine fed me back a link to an irrelevant website of Apple. When prompted to give a valid citation, it then linked an un reputable source. This is a setback on the technology, as there is a lot of either phony or outdated information in the internet, and if ChatGPT cannot differentiate between them, it will very easily spew out this information as solid truths to the prompter. If I had been just a curious person typing the prompt in for more information on how application development worked, it would have been very difficult to understand and process the information that was given. The answer seemed long winded and filled with filler content, some of which ended up not being completely accurate. ChatGPT primarily focused on general characteristics and strengths but did not list significant disadvantages of each method. For example, it did not mention the challenges of App Store approval for iOS due to their strict policies – especially citing what these strict policies are and how to develop your app to pass them, or performance trade-offs with hybrid frameworks. Another key omission was cost considerations, such as Apple's developer fees, the expenses of maintaining native apps versus web-based solutions and the comparison with Android APK. ChatGPT ultimately leaned towards cross-platform development (Flutter/React Native) as the best choice but did not provide a scenario-based recommendation tailored to different app needs.

When applying the same prompt into Gemini, the response was more detailed than ChatGPT's. It broke down pros and cons for each approach and provided real-world application examples. Gemini also highlighted key considerations, such as device compatibility, performance needs, and long-term maintenance. However, the same issues persisted—cost was not mentioned, and the list of development approaches was somewhat limited. It lacked a strong emphasis on offline capabilities and app store publishing hurdles, which are critical factors in deciding between native and web-based solutions. However, one pro was when given additional information about my situation, such as “I want to create an application that manages door status and the user can alter the status from the phone,” the engine mentioned more situation based answers such as discussing the limitations of the APKS in terms of device access and how it could potentially affect the response times and database updates and requests with this limitation.

When prompted, Microsoft Copilot only listed native Android (APK) and iOS development, with brief descriptions of why to use them. There was no mention of cross-platform frameworks, PWAs, or costs. Copilot also failed to break down the advantages and disadvantages of each approach or consider real-world scenarios such as scalability, app store restrictions, or cost efficiency. This

made the response less useful for decision-making, as it lacked depth and alternative solutions. It gave a clear cut answer without much detail. In order to gain more information and insight with this engine, the user would have to kind of bombard the system with multiple questions to force a more useful answer from it. The detail that was given by ChatGPT and Gemini in comparison fared better in terms of comparisons between the APKs.

Chapter 6. Hardware Design

This section discusses the systems that have been implemented for DoorSure. Each system is important in making the devices fully operational and show the reasoning behind why we chose these systems and their contribution to the overall project. Along with the hardware design systems the components used for each system will be explained in detail and the reason behind choosing them. Each component will then be displayed on a schematic for each device.

6.1 Subsystems

Garage Status System

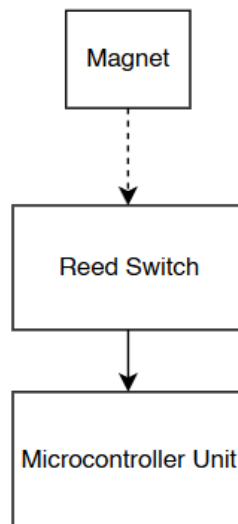


Figure 6.1.1: Garage Status Subsystem

The garage status subsystem is responsible for detecting whether or not the garage door is open or closed. This is one of the main important functions for DoorSure as the user needs to be informed if the garage door has been interacted with as they are preparing to leave their household. The main

component used to detect the status is a reed switch, an electromechanical switch that behaves in different properties depending on the proximity of the magnet from the switch.

For the reed switch to properly function, a magnet is attached to the underside of the garage door. As the primary device containing the reed switch is mounted alongside the edge of the garage door, the placement allows the reed switch to be in full proximity of the magnet's presence. When the user interacts with the web application to either close or open the garage door, the magnet will either be in range or move away from the reed switch. The switch then sends data to the microcontroller, which decides the status of the garage door. The information of the garage door's current status then gets sent to the web app for the user to see that the garage door is either open or closed after that interaction.

Object Detection System

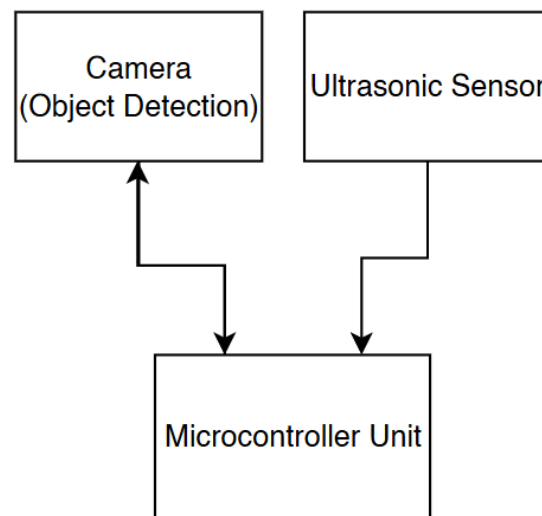


Figure 6.1.2: Object Detection Subsystem

The object detection subsystem is involved in detecting any objects that are obstructing the path of the garage door. Two components are used for this system. The main component that oversees the object detection is the OV5640 camera. The OV5640 camera is currently installed in the primary device that contained the ESP32-S3 MCU, which through the ESP32 camera library was able to support this camera module. The camera requires at least 3.3V of power to be operated. A two way connection is formed between the microcontroller and camera; the microcontroller powers the camera's functionalities while the

camera sends information to the microcontroller for any objects that are detected. The camera is programmed through computer vision to detect objects that are within the garage door's path. Although programming the camera to detect objects in its field of vision can dedicate a great amount of time, we decided to limit the camera to view objects that are mainly common in blocking a garage door's path, like small moving objects and spherical objects.

In the case that the camera fails to function, the ultrasonic sensor operates as a backup or in tandem with the camera. The ultrasonic sensor we are using is the RCWL-1601, requiring 3.3V to power the component via the Vcc pin. The trigger and echo pins are wired to the digital pins on the microcontroller. The RCWL-1601 uses ultrasonic waves to measure the distance from where the sensor is emitting. If an object is within the distance that is measured, the ultrasonic sensor then sends information to the microcontroller, which is then sent to the web app to notify the user that something was blocking the way of the garage door.

Camera Feed System

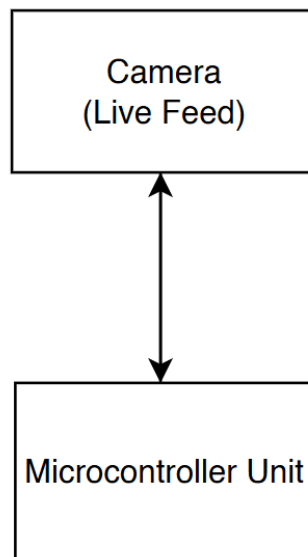


Figure 6.1.3: Camera Feed Subsystem

The camera system is a subsystem that is part of the main system. There is a two way communication for the microprocessor to power the camera while the camera sends data to the microprocessor to send information to the web application. The same OV5640 camera being used to detect objects within the primary device's path is also used as the live feed.

Before switching over to a different MCU and a camera, the primary device originally had the ESP32-WROOM with two OV2640 cameras to perform the different functions, as using one camera to operate both would put strain on the MCU. When switching over to the ESP32-S3 however, the high processing power allowed the camera to operate both functions without any strain and even upgrading to the OV5640 made a minimal difference in performance.

Temperature and Humidity System

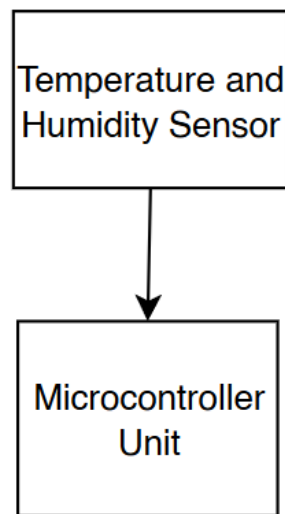


Figure 6.1.4: Temperature Subsystem

The temperature and humidity system serves to analyze the temperature and humidity of the garage room, making sure that both the main and secondary devices are still functional should the room reach abnormal conditions. When certain temperatures or humid values are met, the sensor sends the information to the microcontroller, alerting the user that the devices may risk damage under these conditions and enters into sleep mode until the conditions return to normal. The reason why temperature and humidity are in the same block is that the device mentioned next has both functions integrated into a single device.

The DHT11 temperature and humidity sensor is used for the secondary device, requiring 3.3-5V to operate and a data pin that can be connected to a digital pin on the microcontroller. The range of operation in temperature is 0-50 degrees Celsius (32-122 degrees Fahrenheit), and has an accuracy of around 2 degrees celsius. Additionally, the DHT11 provides relative humidity value in percentage,

having a range of operation of 20 to 90 percent RH with an accuracy of 5 percent RH, the DHT11 uses a resistive humidity measurement component and NTC temperature measurement component, the high accuracy of the sensor makes the system feasible to implement in alerting the user about the status of the devices through the room temperatures and humidity.

The direction of data flow shows when the DHT11 collects the temperature and humidity from the surrounding environment, that data then gets sent to the MCU, which sends the information to be updated on the web application. After sending the necessary information to the MCU the DHT11 goes into low power mode until the next start pulse, which is a highly desirable feature for our project as our devices need to operate with minimal energy being wasted.

Buzzer System

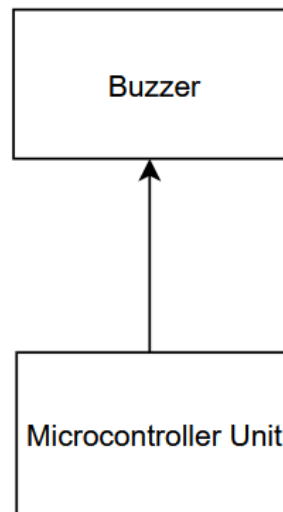


Figure 6.1.6: Buzzer Subsystem

The buzzer subsystem contains the buzzer component for when the device becomes active in use. A sound is initiated indicating that the buzzer is in use until the device finishes its main function. The secondary device has a piezoelectric buzzer installed, being configured to make a sound when, for the primary device, the reed switch is being interacted with, whether the magnet attached to the garage door enters or leaves the proximity of the reed switch. For the secondary device, when the user interacts with the web application to initiate the button press, the buzzer goes off when the servo motor is initiated.

The piezoelectric buzzer has two pins; one for data and the other for ground. The data pin can be wired to a GPIO pin that is capable of transmitting pulse width modulation signals. This is so the microcontroller unit can send PWM signals for the piezo element to vibrate at different oscillations in order to create the buzzer noise.

Servo Motor System

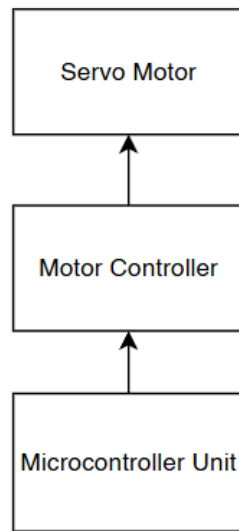


Figure 6.1.7: Servo Motor Subsystem

On the secondary device, the servo motor system enables the use of the servo motor to press the garage door button. The subsystem consists of an MG90 servo motor, which has a motor controller built in. This is powered by the microcontroller, providing the necessary directions to the servo motor to properly function. We chose this servo motor due to having its advantages over the related SG90 servo; the MG90 contains metal gears which maximize longevity and minimize maintenance frequency. Despite the inclusion of the metal gears the component is light in weight and manages to provide a stall torque of 1.8kg/cm in a small size, which is sufficient enough for our button pressing device. The servo motor has 3 pins: Vin, Gnd and PWM. As the minimum voltage required for the MG90 motor to operate is 4.8V, the servo is being provided with regulated 5V of power, with common ground being wired to the ground pin on the MCU. For PWM, this can be connected through a GPIO

data pin. As the MG90 has the motor controller built in, it provides stable data signals when the user opens or closes the door on their application.

On the servo schematic that will be discussed later on in the report, the MG90 can be seen as the male pinhead; the servo motor chosen uses female jumpers for pin connections, which is easier to connect to the servo system PCB as we can install male headers on the PCB for simple hardwiring.

Communication System

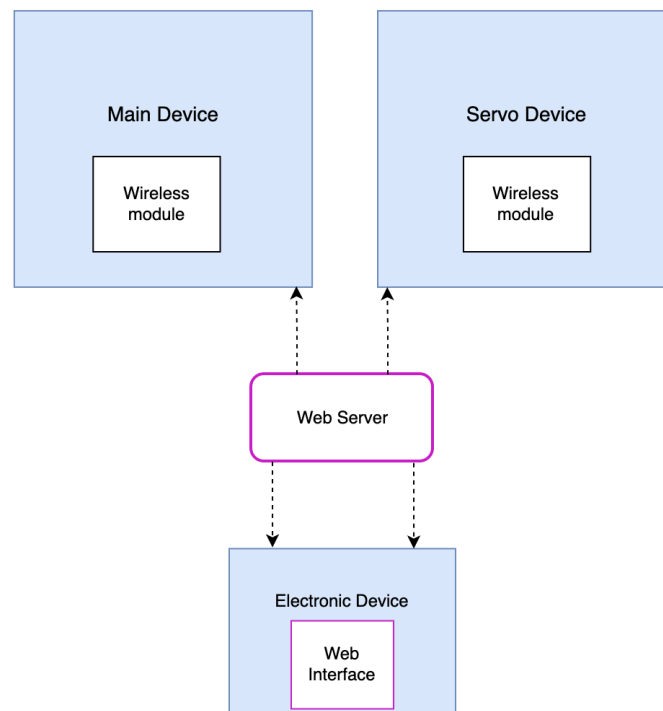


Figure 6.1.8: Communication Subsystem

The communication subsystem for DoorSure allows each device to communicate to the web server when the user interacts with the web application. Each device is equipped with an ESP32 microprocessor, which has integrated wireless communications. Although the devices can connect to each other through applications like ESP-NOW, this would sacrifice the ability to connect to the web server. Through intensive testing and research we learned that there would be no possible way to communicate with the web server while communicating with each other, as only one kind of communication can only be active while the system is in use, meaning that if we want to incorporate the ability for the devices to connect to each other and use other wireless communications to the web server, the system must immediately shut down one

kind of communication before immediately switching to the other type, which is not viable for a system that needs to have a quick response time for notifying the user in relations to the garage door. The most efficient, practical and feasible solution is to have each device communicate directly to the web server instead of each other. This solves the issue of the method of communication as instead of using two different communication styles to interact with the devices to the web app, having the devices communicate directly to the user application with only one method eliminates the need of delay switching to another communication method and in the long run not complicate in further troubleshooting as using two different methods would result in more microprocessors to be added to accommodate the two wireless setups.

In our wireless communication system, the main source of communication between the ESP32 processors in the main and secondary devices with the web application is wifi, as this allows the user to connect to the devices through the web app at a different location if the user is not present as their household. In terms of other communication types like JTAG, UART and I2C. The OV6540 camera module is connected to the ESP32 through the SDA and SCL pins via I2C, allowing serial communication to occur between the two devices.

Power Management System

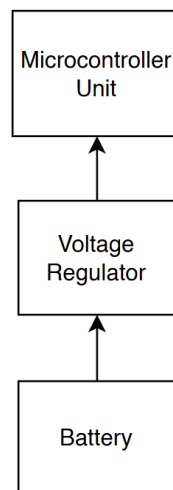


Figure 6.1.9: Power Subsystem (general)

The primary and secondary devices utilize their own power systems as they are not sharing the same MCU and components. To make troubleshooting any issues with power management less troubling and time consuming they are assembled on a daughter board for each respective system. This allows for the

separate system to be easily replaced as they are not integrated into a main board, compared to if the power system was fully integrated on the main pcb. The construction of the power system is identical for both devices to reduce any chance of unique variety and to make debugging and replacing parts less time intensive.

Battery Choice

For consistency reasons the main and secondary devices are powered by the same battery. As listed in Chapter 3 in the batteries section, lithium batteries were the main choice of primary battery to use. Compared to carbon-zinc and alkaline, lithium batteries have a longer capacity and while slightly expensive to buy, they last for a good amount of time in the long run. As for the battery size, we decided to choose 556065 batteries as the main source of power. This can be attributed to a few aspects to our reasoning for choosing an obscure battery size. Compared to AA and AAA batteries, which can only hold up to 1.5V, 556065 batteries can hold up 3.7v, requiring less batteries to be used to power the devices. The 556065 batteries also have a small form factor, meaning that we can use at most two batteries for each device without taking up much space. The downside of using the 556065 batteries is their prices as lithium batteries are more expensive compared to alkaline batteries, although the price paid for a two pack of the 556065 batteries is worth the longer battery life of the devices, additionally, the 556065 are lipo batteries which mean they are also rechargeable batteries which would also lower our overall cost of ownership.

Power Regulation

Both main and secondary devices require the use of a voltage regulator in order to supply a constant voltage to the devices while minimizing any risk of potential voltage spikes or drops during operation. Switching regulators were prioritized over linear regulators as switching regulators are highly efficient in power consumption and produce less heat when dissipating.

We chose buck regulators to generate a 3.3V and 5V output. Using these buck regulators allow us to use two lipo 556065 3.7v batteries connected in series to power the primary and secondary device. As they are installed on the daughter boards, we can easily switch them with our backup LM2576 regulators in the case that the AP regulators do not work in our favor under load, which would also help us in debugging and maintenance.

Table 6.1.10 Switching Regulators

Parameters	AP63203	AP63205
Input voltage	7.4v	7.4v
Output Voltage	3.3V	5V
Reference voltage	0.8V $\pm$ 1%	0.8V $\pm$ 1%
Quiescent Current	22 μ A	22 μ A
Switching frequency	1.1 Mhz	1.1 Mhz

For a general outlook of the power systems as shown on Figure 6.7, two 556065 lipo batteries are connected in series to the voltage regulators (3.3v and 5v), which powers the microcontroller using the 3.3v regulator and most peripherals, while 5V regulator powers the servo motor on the secondary device. The specific components used for the power system have been explained earlier in this section.

6.2 Power Distribution

Before approaching the entire design of the hardware, an overview of the power distribution of both the primary and secondary devices need to be observed.

Two 556065 Li-Po batteries are connected in series for a voltage input of 7.4V; as they have a capacity of 2500mAh, the components need to have a system current draw of under 2.5A.

These are the component's power specifications for the primary device:

Table 6.2.1: Primary Device Component Power Specifications

Component	Voltage	Current
ESP32-S3	3.3V	500mA
OV5640	3.3V	140mA
RCWL-1601	3.3V	2.2mA

With the given voltage and current values for the components, we can apply the total power consumption of the device with the power formula:

$$P = V \times I$$

$$P = 3.3 \times (500mA + 140mA + 2.2mA)$$

$$= 2.11W$$

Figure 6.2.2: Primary Device Component Power Formula

The output of the primary device's system power draw is 2.11W, which when considering power losses, the total consumption can be rounded to 3W. To find the system current draw, we can divide the system power draw by the input voltage to get a current draw to 0.41A, which is sufficient for system power.

For the secondary device, these are the component's power specifications:

Table 6.2.3: Primary Device Component Power Specifications

Component	Voltage	Current
ESP32-WROOM	3.3V	500mA
DHT11	3.3V	0.2mA
MG90S	5V	250mA

With the given voltage and current values for the components, we can apply the total power consumption of the device with the power formula:

$$P = (3.3V \times (5mA + .3mA)) + (5 \times 250mA)$$

$$= 2.9W$$

Figure 6.2.4: Primary Device Component Power Formula

The output of the primary device's system power draw is 2.9W, which when considering power losses, the total consumption can be rounded to 3.2W. To find the system current draw, we can divide the system power draw by the input voltage to get a current draw to 0.43A, which is also sufficient for system power.

6.3 USB to UART bridge

The CP2102 is the USB to UART bridge that allows us to program and flash information to the ESP32-WROOM and the ESP32-S3. The USB to UART bridge allows a straightforward connection between the computer's USB to the ESP32 serial pins.

Figure 6.5 in (appendix C) is a schematic of the CP2102 from Silicon Labs showing how the chip connects to a USB port on the left side of the figure, the VBUS pin would be the 5V from the USB port, and D+ and D- would be the data connection from the USB port. We can observe optional components such as the 4.7Kohm pull up resistor as option 1 which can be added to increase noise immunity, option 2 would give us the ability to power other devices from the on-chip regulator by adding the 4.7 microFarad capacitor. In conclusion, these optional components offer more flexibility and improved performance to our overall circuit.

6.4 JTAG and UART

UART, which stands for universal asynchronous receiver transmitter, is a hardware feature that handles asynchronous serial communication. In our project, we are using UART pins to program the ESP32 chip. First, we set the communication parameters which includes baud rate. We then assign pins for connection which in our case would be the pins showcased in the UART connection table below which are critical for transmitting and receiving data. Before initiating UART communications, we would need to install the driver specified by the datasheet for our CP2102 USB to UART bridge, which converts the USB signal to serial signal that the ESP32 chip understands, then we can initiate the UART connection, transmitting and receiving data from the MCU and USB connection. We are using the UART connection for programming of the MCU and consequently our entire board.

Below is the UART connection to our microcontroller as specified by the ESP32 datasheet.

Table 6.3.1: UART Connections

Pins	Name	Functionality	Connection
1	TX	Transmit data between the MCU and USB to serial converter	TXD0 on the MCU
2	RX	Receive data between the MCU and USB to serial converter	RXD0 on the MCU
3	GND	Ground	Ground

Additionally we included JTAG (Joint Test Action Group) pins to our MCUs for debugging. Although we can debug without the need to setup JTAG by using the idf.py monitor and connecting to the ESP32 through the UART pins, we chose to add JTAG pins as an additional method of debugging. Both methods are integrated under the Eclipse ide in order to provide a quick and easy transition between compiling and debugging code with its built-in capabilities other than directly from the terminal window. JTAG provides additional advantages as well, as it lets us pause code execution and inspect variables in real time which would be a major advantage especially during the prototype and testing phase. Since the idf.py monitor route uses the UART pins, we can use the JTAG pins to debug in case of damage occurring to the UART pins after programming. This ensures our project is able to withstand damages and continue to function as usual while allowing us to debug. We may remove these pins in future versions of our project, since we would be more confident in our code and our design but for the prototype phase, we recognize that solving errors that may face us is part of the process for developing a product.

Below is the JTAG connection to our microcontroller as specified by the ESP32 datasheet.

Table 6.4.2: JTAG Connections

Pins	Name	Functionality	Connection
1	3.3v	Power supply for the JTAG interface	3.3V regulator
2	MTD0	Test data out	(IO15) pin 23 on the MCU

3	MTCK	Test Clock	(IO13) pin 16 on the MCU
4	MTDI	Test data in	(IO12) pin 14 on the MCU
5	MTMS	Test mode select	(IO14) pin 13 on the MCU
6	GND	Ground	Ground

6.5 Schematic Design

Primary Device Schematic

This schematic is composed of the primary device, which contains the components that have been explained earlier in the main system. The heart of the main system schematic is the ESP-32 S3 N8R8 variation, acting as the main source of data communication between the components along with wireless communication to the user web application.

As mentioned above, we included UART pins which connect to programming pins via our custom PCB daughter board mentioned down below. The primary device serves as our object detection and classification peripherals by the ultrasonic sensor and the OV5640 camera, additionally it includes the reed switch on one of the unused pins that were exposed.

All extra pins are exposed as a precaution so we can easily change the pre planned pins for the peripherals which eliminate a process of redoing the board in case of any damages, this process would be incredibly useful for us due to the limited time frame of the summer term.

For the servo motor unit, we added a male header which would make connecting to the MG90S easier as the MG90S wiring ends off with female headers, which is convenient for when we need to switch servo motors for maintenance and testing purposes. The servo motor is powered by a regulated 5V from the daughter board as illustrated in the schematic, which is connected from the Vcc pin on the servo motor to the power source. The second pin is used for the PWM control signal from the ESP32's GPIO pin 18 and the last pin is the GND pin, having it as common ground with the MCU. For the DHT11 connection, this follows the same pin layout as the MG90S (Vcc, Data, Ground, the fourth pin is unused), but for the data pin, it is connected to pin 25 in order to output the value of both temperature and humidity as serial data. Additionally a 5k pull-up resistor is connected to the 5V regulator due to having a connecting cable shorter than 20 meters as instructed by the datasheet.

According to the ESP32-WROOM-32 datasheet, it is advised to add an RC circuit delay of a resistance of 10k Ohm and a capacitance of 1 microFarad at the EN pin to ensure the power supply to the ESP32 during power-up; additionally we would add 0.1 microFarads which would act as a decoupling capacitor, since there may be a sudden increase in the current draw, this can potentially cause power rail collapse, therefore it is recommended to add the 10 microFarads to the power rail.

Additionally, a buzzer is be installed on the Secondary system, which is used to create sound giving us an additional advantage in debugging and testing; it can also be considered as another safety layer as we may program it to produce a sound after a small delay from the servo motor's motion so that it can be when the garage door is in motion providing an audible feedback, offering a clear and a real time indication of movement.

Lastly, we added UART and JTAG pins in our design since it is a common practice and recommended in the ESP32 datasheet, which was explained more in the JTAG and UART section. The UART is connected by TXD0, RXD0 and GND in this order, the JTAG is connected to IO14, IO12, IO13 and IO15 via 100 ohm resistors for each line. We added a 3.3V pin as a reference for the debugger and GND pin as well to the JTAG pins. UART pins are added to our schematic since it is simpler and is used by the built in bootloader while the JTAG allows true hardware debugging which may help us in debugging and testing our project as a whole.

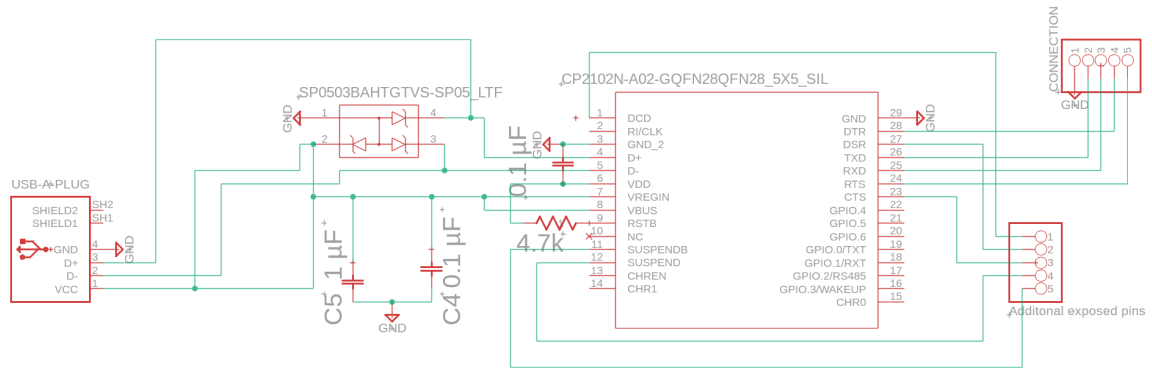


Figure 6.5.3: USB to UART Converter Schematic

Temperature and Humidity sensor Schematic

The temperature and humidity sensor would be the simplest schematic we have, as it only contains 3 components, the DHT11 sensor is responsible for measuring the temperature and humidity of our garage, the 10k ohm resistor is there to ensure a reliable digital signal to decrease the noise and fluctuations for a correct readings and the male header which connects to the secondary device.

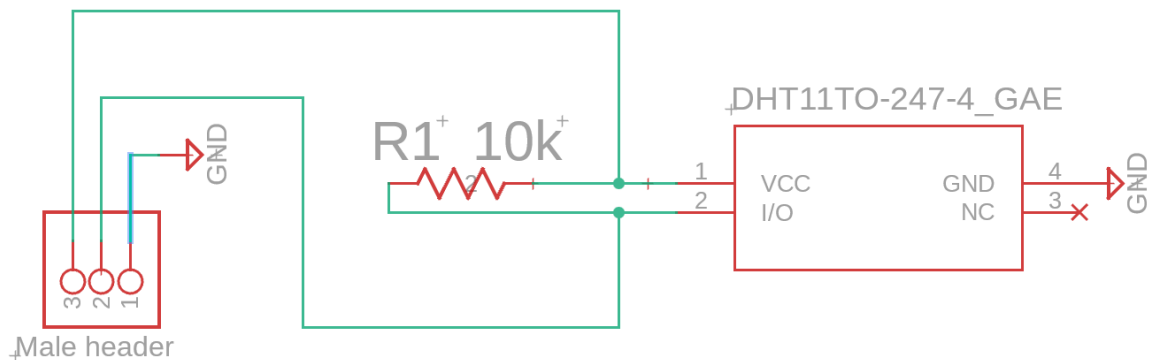


Figure 6.5.4: Temperature and Humidity sensor Schematic

Power Systems Schematic

Below is a schematic of the voltage regulators we are going to be using in our project. Both schematics have the same design overall but the regulators are swapped between them.

Additionally, we have backup regulators of LM2576 which may be used in case the AP regulator didn't work as expected, these backup regulators offer us a peace of mind since under intensive testing the AP regulators may not act as expected, the schematics for the LM2576 regulators can be found in appendix C.

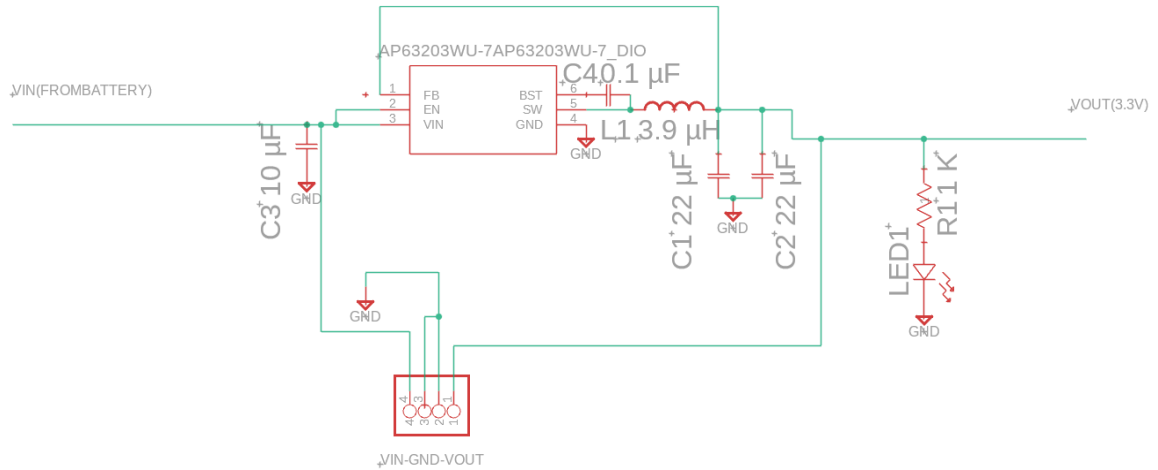


Figure 6.5.5: 3.3V regulator Schematic

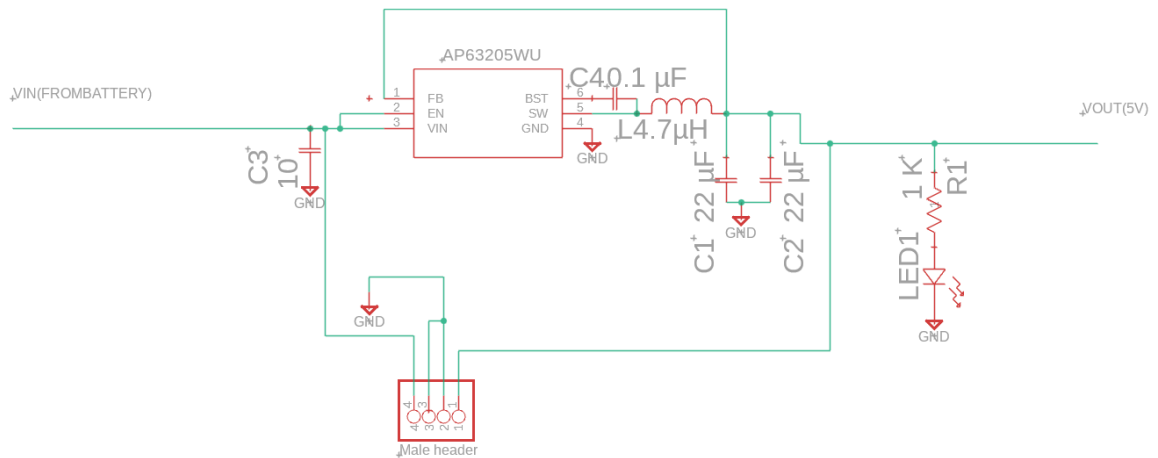


Figure 6.5.6: 5V regulator Schematic

Chapter 7. Software Design

7.1 Use Case Diagram

Before beginning the development of the software components for DoorSure, it was critical to first define and understand the overall system behavior. This foundational step helped clarify what features our application needed to provide, how the system would function, and how each component would interact. To accomplish this, we created a Use Case Diagram, shown in Figure 7.1, which served as a visual guide to the functional relationships between users, system components, and services.

The system involved a total of three actors: the User, Firebase, and the Garage Door System, which was composed of two ESP32 microcontrollers. As described in Chapter 2, our team's goal was to design a web-based application that would serve as the primary user interface. Through this application, the user could perform key actions, including the most essential feature: remotely opening and closing the garage door. For the purpose of the use case model, the two ESP32 microcontrollers were abstracted into a single actor labeled "Garage Door System" to simplify the diagram. It is important to note that while these microcontrollers operated as part of the same system, they were not physically connected. Instead, they communicated wirelessly with each other and with Firebase to coordinate their tasks and respond to user commands.

Each ESP32 had specific responsibilities. One unit was dedicated to controlling the servo motor responsible for the door's movement, while the other managed the reed switch and camera module for monitoring door position and detecting potential obstructions. These distinct roles and their hardware details are explained further in other diagrams and hardware sections of this report.

The use case diagram was instrumental in shaping our early development process, ensuring that we had a clear blueprint for implementing both front-end user interactions and back-end system logic.

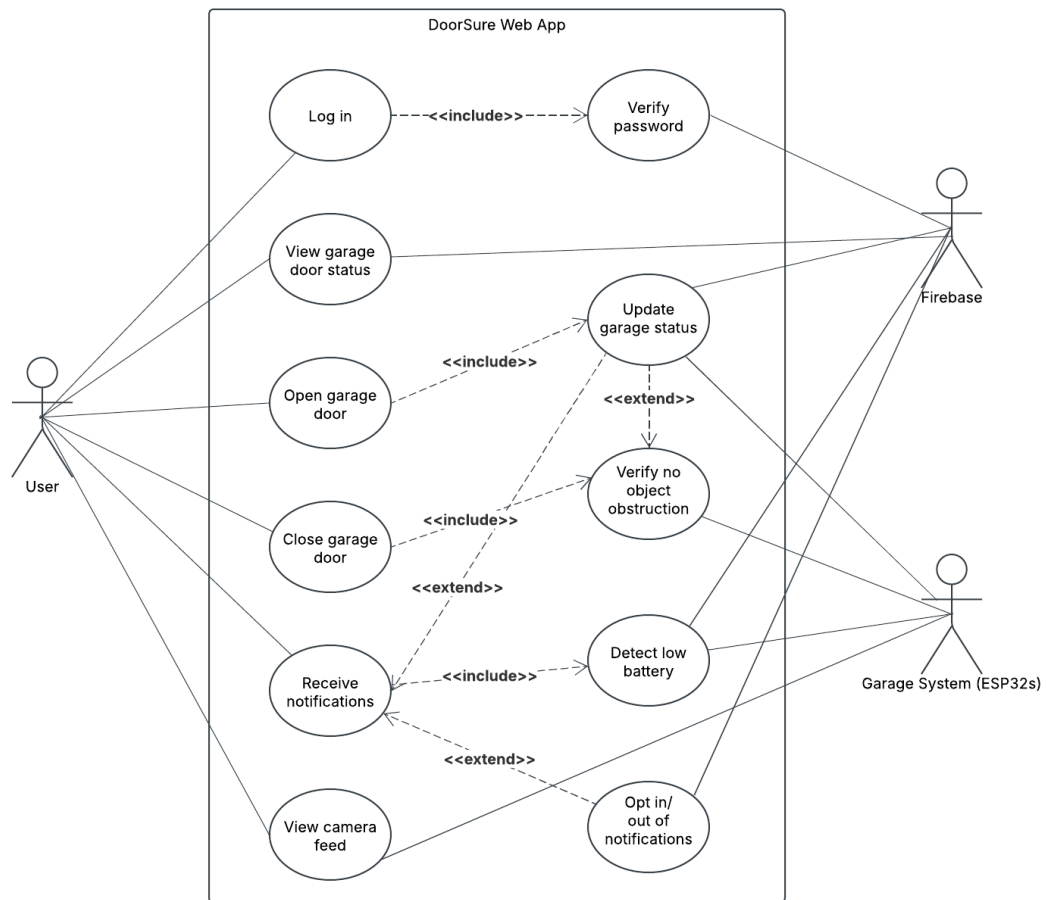


Figure 7.1: Use Case Diagram

7.2 Entity-Relationship Diagram

Before setting up the database, we need to understand how different entities interact with one another. The Entity-Relationship Diagram helps us lay out the relationships between key components, ensuring that all the data is stored correctly.

As shown in Figure 7.2, The Garage is the core of the system, linking everything together. Each User has a unique User_ID, which connects to a Garage_ID, allowing them to control. The Sensor entity tracks whether the garage door is open or an object is in the way, ensuring that users get real-time updates. The system also includes a Camera linked to the garage for live video monitoring. The Button Device represents the physical button that users can press to open or close the garage. Additionally, the Battery Status entity tracks battery type, charge level, and the last updated time. This helps the system track the battery level and alert the user. Finally, the Notification entity keeps users

informed with all kinds of messages, such as garage status changes, sensor detections, and low battery alerts.

This database structure organizes the system's hardware and software and supports the full functionality of the DoorSure System.

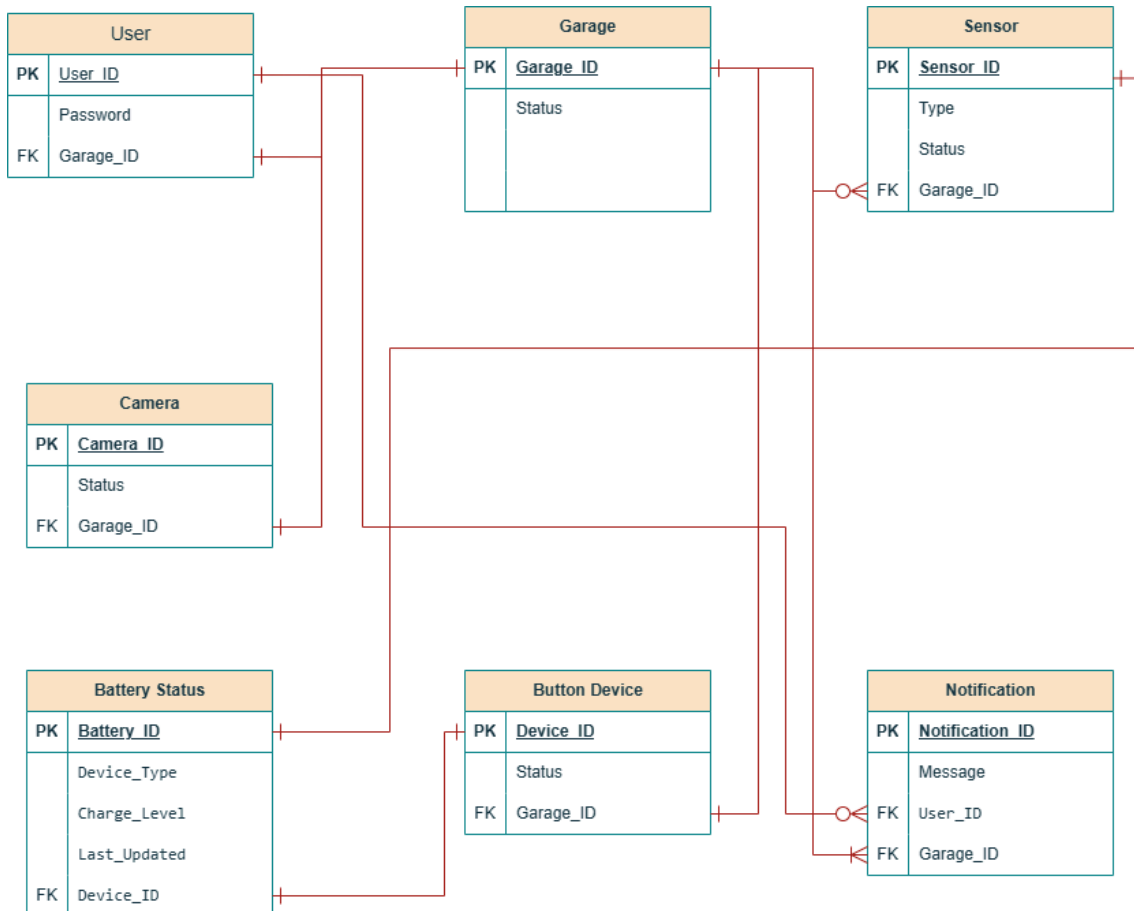


Figure 7.2: Entity-Relationship Diagram

7.3 Class Diagram

Before starting the software development process, it's crucial to outline the different parts of our system and their interactions. This diagram helps us organize the structure by breaking down key components and their relationships. This diagram ensures that everything works together before we start implementing the system.

As shown in Figure 7.3, we can see the system is built on multiple key classes, each representing a specific aspect of the garage door detection system. The User class handles authentication, allowing users to log in and manage their garage. The Garage class is responsible for checking the door's current status and providing a method to open and close it. The ButtonDevice class simulates the physical garage button, enabling users to operate the door remotely. Safety is our top priority, which is why the Sensor class is included to detect obstacles and prevent the door from closing on objects. Additionally, the Camera class allows users to monitor their garage with live video feed.

Power management is another important aspect of our project. The BatteryStatus class keeps track of battery levels, making sure the system remains operational. Finally, the Notification class keeps users informed, sending alerts with all kinds of information.

The DoorSure system seamlessly integrates hardware and software. This well-defined class diagram not only helps us organize the garage door system efficiently but also monitors and controls every process.

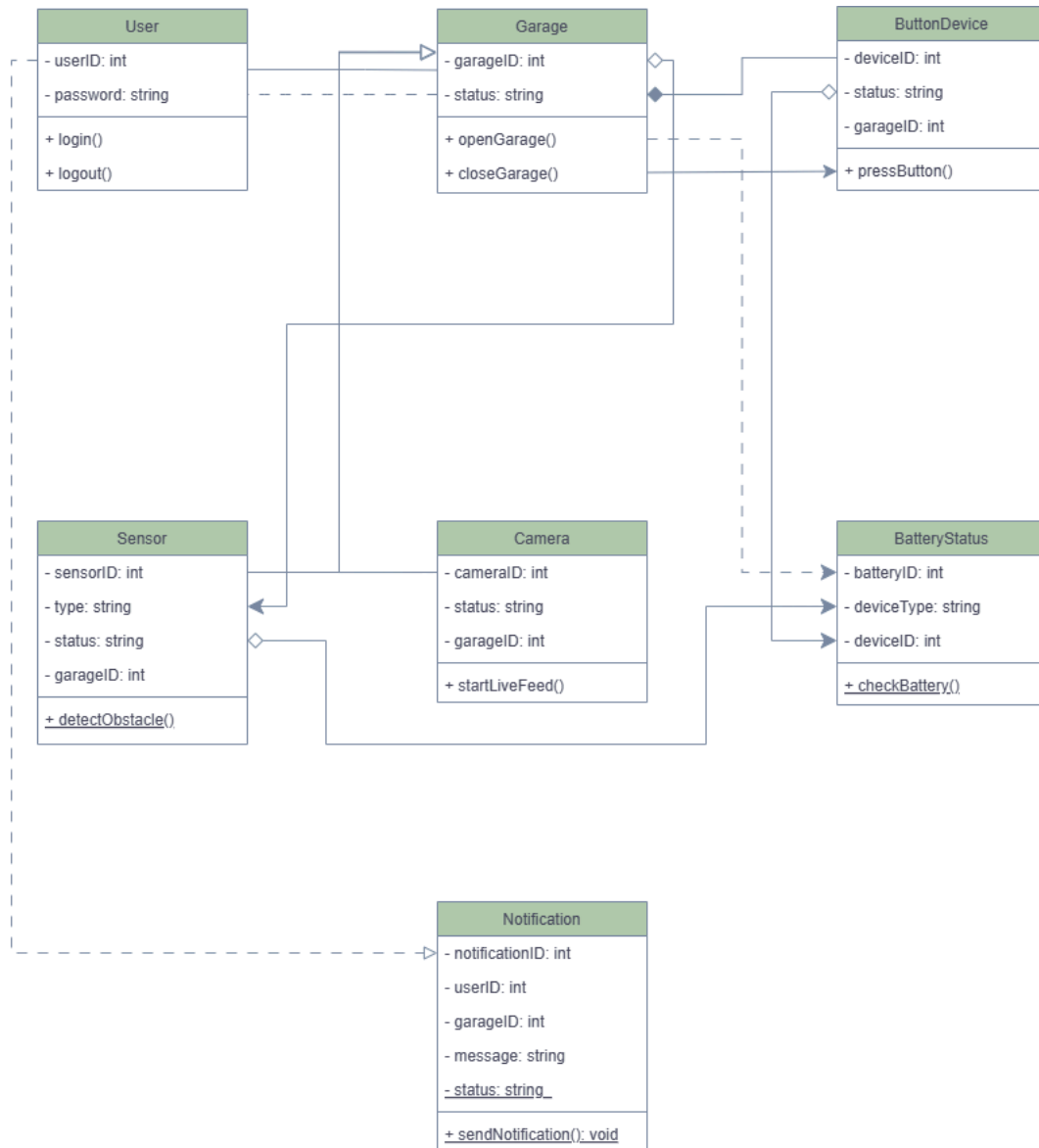


Figure 7.3: Class Diagram

7.4 State Diagram

The State Diagram in Figure 7.4 breaks down how the DoorSure system responds to different states based on user actions. The system starts in an Idle state, continuously waiting for user input. From this state, it can transition based on what happens next. If the user enables the live feed, the system moves to the Live Feed state, allowing real-time video streaming. If the battery is running low,

the system enters the Low Battery Warning state to notify the user to change the battery.

The primary interaction happens when the user presses the button to open or close the garage. If the door is closed and the button is pressed, the system moves to Garage Opening and then to Garage Opened. If the door is opened and the button is pressed, the system moves to Garage Closing and then to Garage Closed. However, if the system detects an object in the way, it stops and notifies the user instead of proceeding.

This state diagram helps us map out exactly how the system should react in different situations, making it smooth and safe. By outlining these different states and their interactions, the diagram provides an overview of how the system should respond to real-world scenarios.



Figure 7.4: State Diagram

7.5 Sequence Diagram

The sequence diagram, seen in Figure 7.5, outlines the process flow of the DoorSure system when a user interacts with the garage door via the web application. It clearly illustrates how the system components—Web Application, Firebase, Secondary Device, and Primary Device—communicate with one another in response to the user's open and close commands.

When the user presses the open button on the web application, a request is sent to Firebase with the action labeled 'open'. Firebase reflects this change and the Secondary Device moves the servo to open the garage door. Afterward, it sends an updated 'door status' back to Firebase, which is logged for real-time monitoring. The Primary Device performs a self-call when the physical garage door moves, updating the Firebase 'door-status' accordingly.

When the user presses the close button, a similar request labeled 'close' is sent to Firebase. Before any action is taken, the Secondary Device checks for object obstruction by querying the Primary Device. If an obstruction is detected, an error is displayed to the user and the close action is aborted. If there is no obstruction, the Secondary Device proceeds to move the servo to close the garage door and updates the 'door status' accordingly.

This sequence diagram helps clarify the flow of control and decision-making within the system. By detailing how each component interacts and reacts to various inputs and conditions, the diagram ensures that all operations are executed safely and efficiently, particularly in scenarios where object detection plays a critical role in preventing accidents.

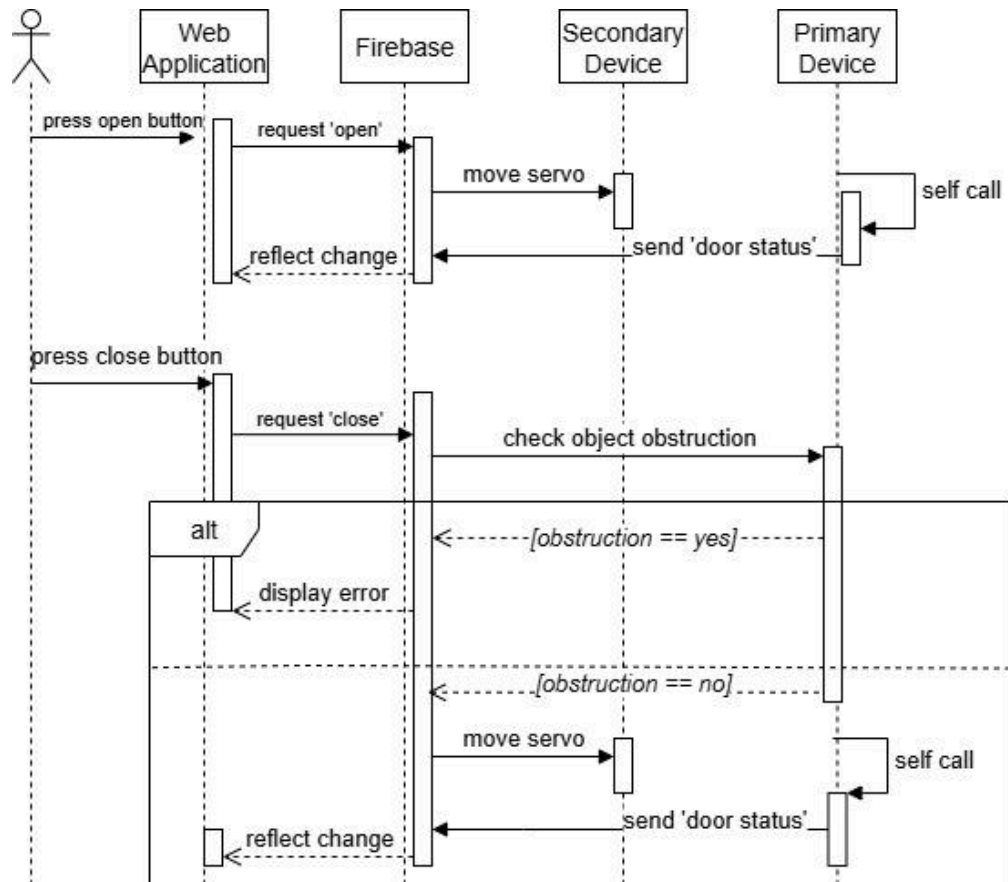


Figure 7.5: Sequence Diagram

7.6 User Interface

As stated previously, nearly all user interactions with our system were performed through the web application. Therefore, it was absolutely crucial that the user interface was clean-cut and user-friendly. Additionally, since our application is mainly meant to be viewed on a mobile platform as opposed to a laptop or PC, we made sure to design the pages with this in mind. Our goal was to make the application as intuitive and easy to navigate as possible to ensure a smooth and accessible user experience. We utilized Penpot to generate the initial UI prototype, which allowed us to simulate the visual layout and flow of user interactions across the application. This helped us validate usability early on before final implementation. The web application consisted of four main HTML pages: a login page, a home page, settings page, and a live camera feed page.

The login page, shown on the left in Figure 7.6.1, featured a minimalist design that included our project logo, a welcome message, and a login button. We implemented a simple login with any email as our authentication method. For the

actual final UI design depicted on the right in Figure 7.6.1, we ended up not incorporating Google login authentication to allow users with any email domain to sign up. As such, an actual login and sign up needed to be done separately.

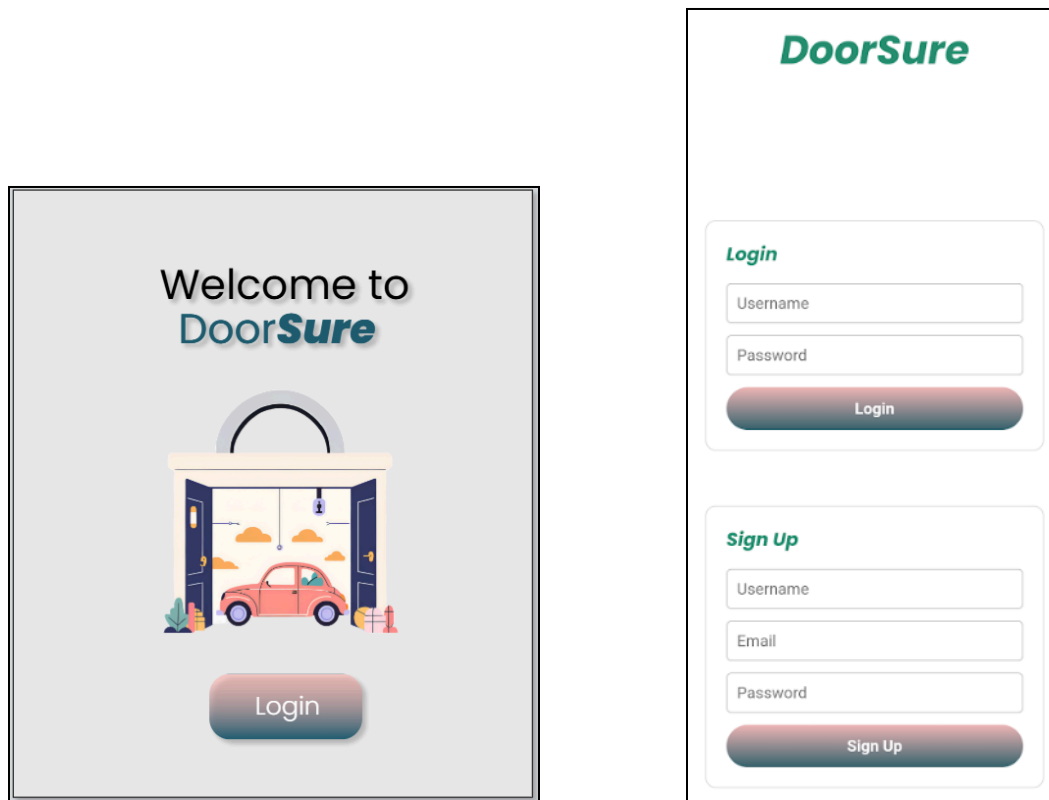


Figure 7.6.1 Prototype Login Page (left), Final Login Page (right)

From the prototype settings page showcased on the left of the Figure 7.6.2, users were able to adjust their notification preference settings. Additionally, they can choose a custom timeout value from a dropdown menu to specify how many minutes the garage door could remain open before a notification was triggered and also opt out.

The final prototype mainly remains the same, except with an additional logout button to ensure authentication is reset as well as adding an area where the user can change the value of garage length, a value crucial to ensuring the ultrasonic is able to determine if an object is obstructing the garage door path.

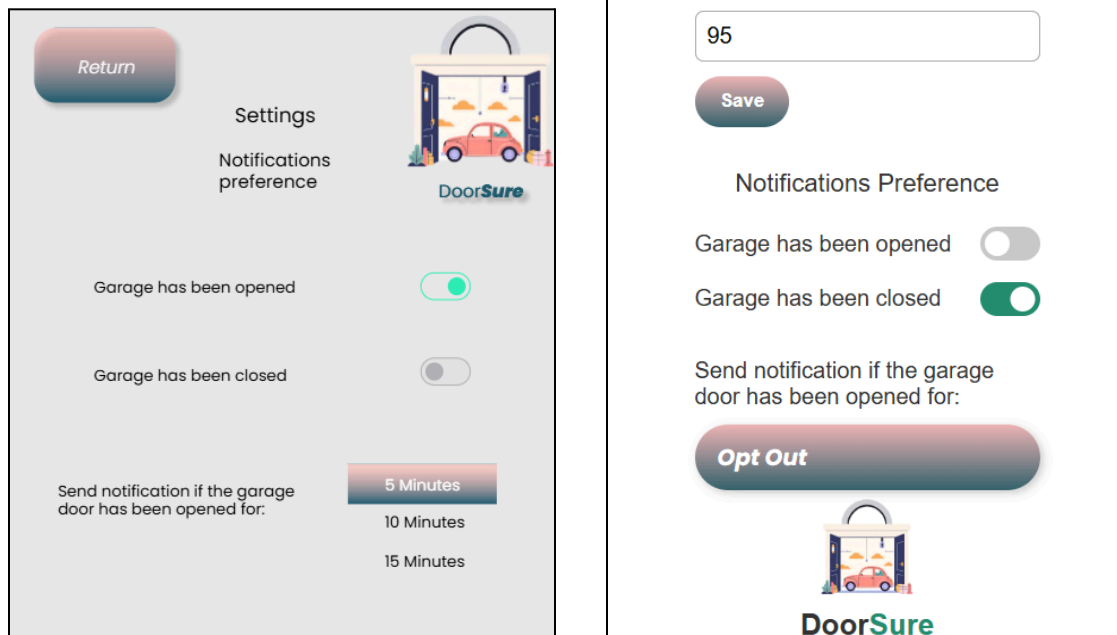


Figure 7.6.2 Prototype Settings Page (left), Final Settings page (right)

The home page displays the current status of the garage, retrieved from the Firebase Realtime Database (RTDB). From this page, users are able to press a single toggle button. If the garage status was “open,” the user could press the button to initiate a close command. Conversely, if the garage status was “closed,” the same button allowed the user to open the garage. Each button press triggered an update to the RTDB—provided there were no obstructions

detected in the path of the garage door. In the final UI design, we can see the User has changed to display the logged in username, as well as display the values read in from the temperature and humidity sensor from the secondary device.

In addition to this primary function, the home page also included a gear icon, which served as a navigation button to the settings page. Users could also select the “View Live Feed” button, which redirected them to the live feed page, shown on the in Figure 7.6.4. These functions remained the same in the final implementation.

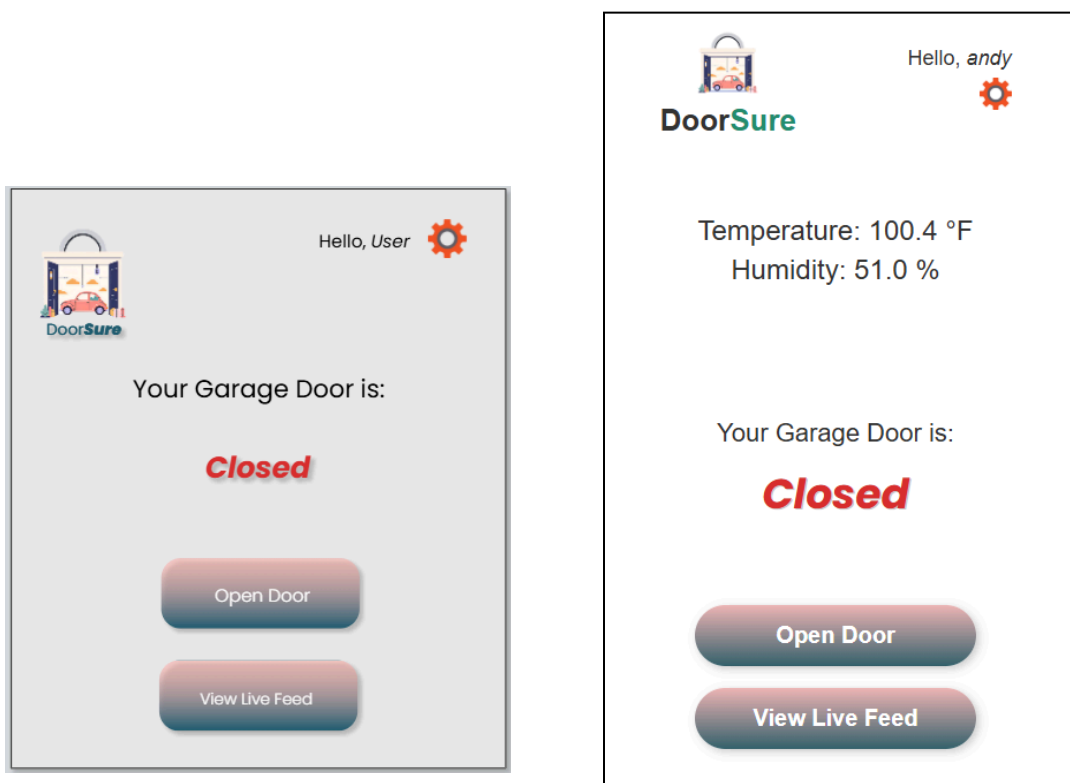


Figure 7.6.3 Prototype Home Page (left), Final Home Page (right)

The live feed page simulates real-time video streams by uploading images. This was done to bypass the limitation of HTTP/HTTPS streaming of using Firebase. Also, in order to limit power consumption, stream control buttons were added. This makes it so that the camera component stays idle unless explicitly activated by the user. Whenever the user closes this page, the stream automatically turns off.

In the event that an obstruction prevents the garage door from closing, the object can be identified by pressing the start identification button which then uploads the image of the stream to Google Gemini. Once the image has been analyzed, the results are then displayed in the proper area for the user to see on the UI.

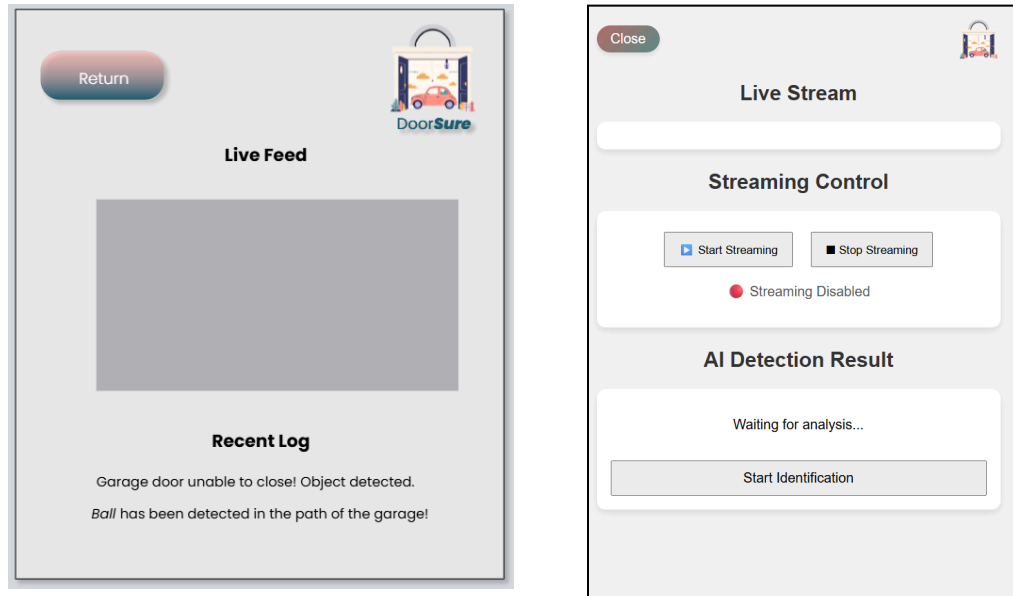


Figure 7.6.4 Prototype Live Feed Page (left), Final Live Feed Page (right)

7.7 Development Environments and Tools

7.7.1 Programming Languages

Our web application system was programmed using the standard languages – HTML, CSS, and JavaScript language. This ensures cross-browser compatibility as the system follows modern web development standards. The program code for the chosen microcontrollers, the ESP32, was written in C.

7.7.2 IDEs

As ESP32s are the microcontroller chosen for our project, we utilize Arduino IDE for all source code written and uploaded to the microcontrollers. Along with using Arduino IDE, we also utilize VSCode for all programming not related to the ESP32. Although our web application is directly linked and deployed using Firebase for testing purposes, we require the IDE to write programs and debug.

7.7.3 Libraries and Dependencies

Within the Arduino IDE, we utilize various libraries provided through both GitHub and the built in library manager in Arduino. The libraries used are as follows:

- ArduinoJson by Benoît Blanchon
 - This library can be accessed by including `<ArduinoJson.h>`
 - A powerful JSON serialization/deserialization library for embedded systems like the ESP32. It is used to parse, generate, and manipulate JSON data efficiently. In this project, it plays a key role in formatting data exchanged between the ESP32 and cloud services such as Firebase, ensuring proper data structure for communication.
- ESP Async WebServer by ESP32Async
 - This library can be accessed by including `<ESPAsyncWebServer.h>`
 - This library works in tandem with the Async TCP, responsible for creating an Async WebServer object to create an object that listens to a server port number. This allows communications with the web application.
- Wi-Fi ESP32 Library
 - This library can be accessed by including `<WiFi.h>` after adding the ESP32 into device manager in Arduino.
 - This library was used to connect the ESP32s to the WiFi network. It was also used to identify the IP Address of the WiFi connection, allowing a websocket to then be generated with the web application for communications with Firebase.
- Firebase Arduino Client Library For Arduino Devices by Mobizt
 - This library can be accessed by including `<Firebase_ESP_Client.h>`.
 - This library allows us to define our Firebase project configurations and authentication tokens, allowing the ESP32 access to write and read data from the Firebase RTDB.
- ESP32Servo by Kevin Harrington and John K
 - This library can be accessed including `"DHT.h"`.
 - This library was used to create a servo object in the program code, which then allowed us to program angle movements on the servo.
- DHT Sensor Library by Adafruit

- This library can be accessed including `<ESP32Servo.h>`.
- This library was used to use and access data from the Temperature and Humidity sensor, and display the temperature and humidity variables on our web application, and alter the values on firebase accordingly.
- ESP32 Camera by Espressif
 - This library can be accessed by including `<esp_camera.h>`.
 - This library was utilized for initial testing using the ESP32-CAM prior to our PCBs being assembled. It enables the configuration, initialization, and capture of images or video frames directly from the camera sensor. This library was not used in the final code as we changed camera modules to meet PCB fabrication requirements.

7.7.4 Web Application Deployment and GitHub

The web application for DoorSure was tested and deployed using Google Cloud Platform's Firebase. RTDB was used to store rapidly changing values, such as the garage door status, which needed near real-time synchronization across devices.

During development, we conducted functional and integration testing using a combination of Visual Studio Code, the command-line terminal, and local host environments to validate front-end and back-end behavior prior to live deployment. These tools allowed us to quickly test changes and verify real-time communication with Firebase.

To ensure proper version control and enable robust collaboration, we maintained a GitHub repository throughout the development cycle. This approach provided each of the four team members with consistent, up-to-date access to all source code and configuration files. Additionally, it served as a safeguard against accidental data loss or software-breaking bugs by enabling us to track changes and revert to previous commits as needed.

Chapter 8. System Fabrication/ Prototype Construction

8.1 Prototype Construction

For Senior Design 1, we built the system using the chosen components for the Primary and Secondary devices.

The Primary System consists of an ESP32, one camera component, and a reed switch.

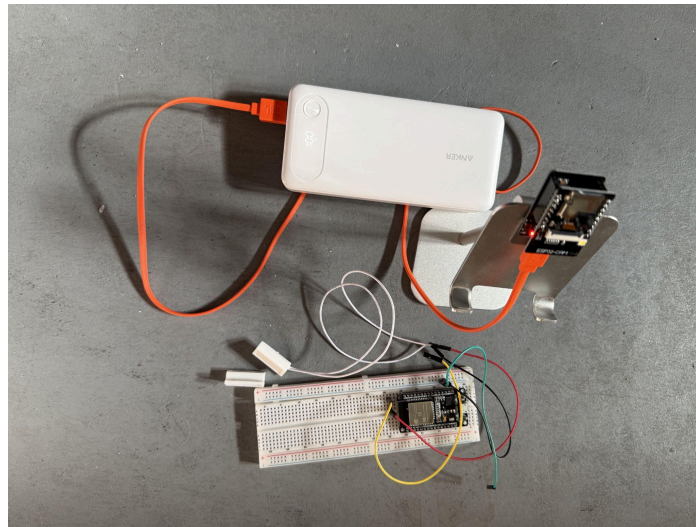


Figure 8.1.1 Primary System Initial Prototype

The Secondary System consists of an ESP32, a servo motor, a temperature and humidity sensor, and a buzzer.

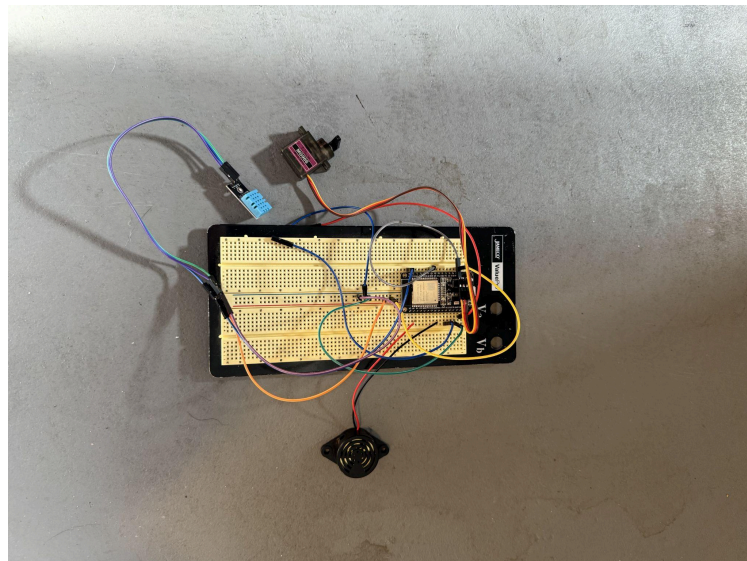


Figure 8.1.2 Secondary System Initial Prototype

The Object Detection system combines the primary device system with an additional ESP-CAM, labeled the “AI Thinker.” The AI Thinker device will be changed to just a camera component with its own MCU for improved processing power in order to meet PCB fabrication requirements for Senior Design II.

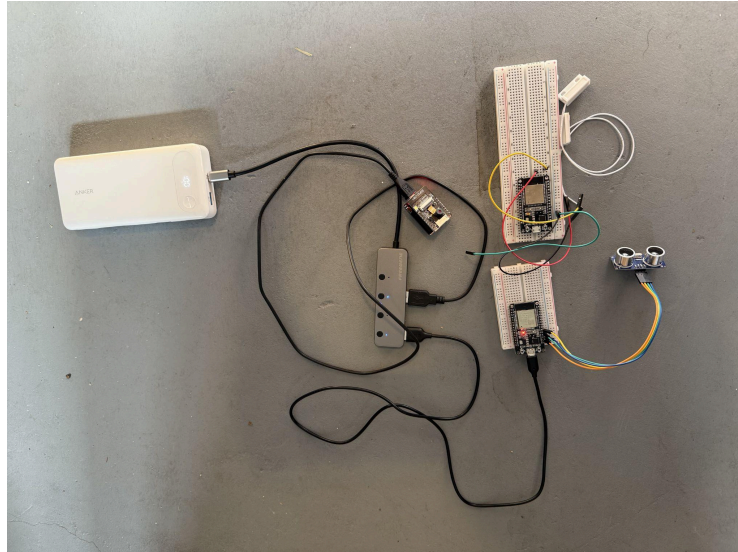


Figure 8.1.3 Object Detection System Initial Prototype

For Senior Design 2, these designs have been finalized, with the primary system containing the OV5640 camera, reed switch, and the ultrasonic sensor, being attached to the finalized primary device’s pcb design.

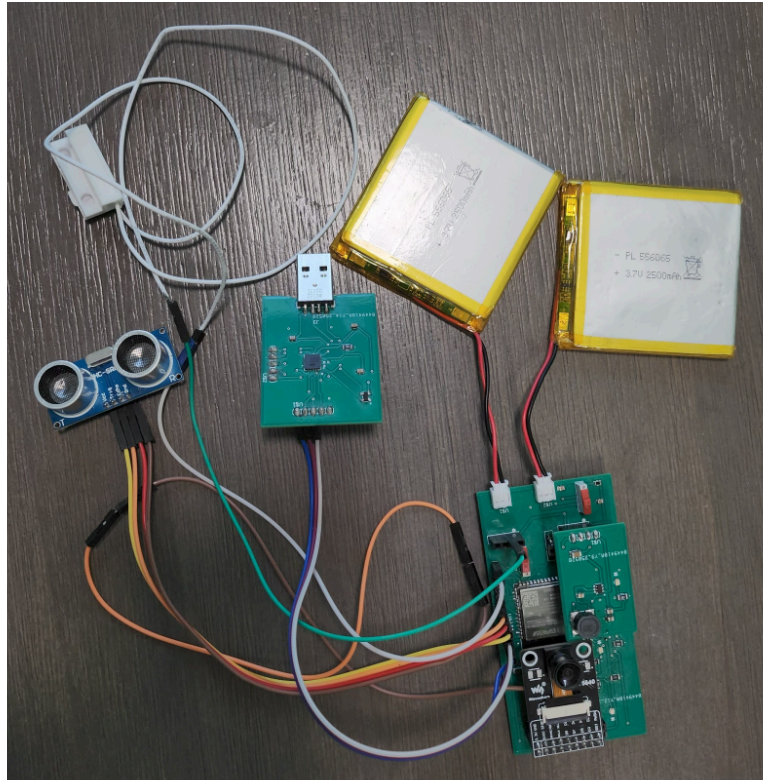


Figure 8.1.4 Primary Device Final Prototype

The same is applied to the secondary device, with the MG90S servo motor, DHT11, and buzzer attached to the finalized pcb design of the secondary device.

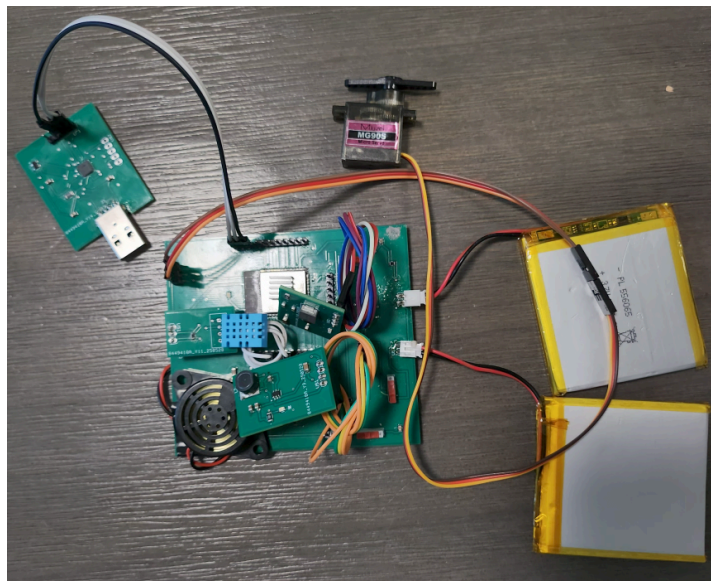


Figure 8.1.4 Secondary Device Final Prototype

Both devices are powered by two 3.7V 556065 li-po batteries, along with the AP63203 and AP63205 voltage regulators for the devices. In addition, UART pins are exposed to connect the dedicated USB to TTL pcb to the devices as well.

8.2 PCB Layout

We followed all required and recommended guidelines, given to us by UCF and the latest IEEE standards. We followed IPC-2221 which identifies generic physical design principles. As we are manufacturing our PCB by JLCPCB we had to follow their guidelines, setting the trace width for all our boards as 6 mm and the via's diameter would be 0.25 mm. Furthermore, for all PCB boards we are using 2 layer PCB boards which allow us to wire components even if they intersect in some point using vias, which is illustrated for each individual system we have.

All schematics and PCBs are done by Autodesk Fusion 360, with component designs are downloaded by Ultra Librarian and SnapEDA, additionally, all wires are manually routed to ensure correct connection on the top and bottom layers using vias to switch connection between layers, the placement of all components which minimizes all routings while keeping the functionality of it since some components need to be on the edge of the board to keep its functionality such as the servo motor and the ultrasonic sensor, additionally, due to the recommendations of our advisors, we added LEDs for power indication for our two main boards.

In total we are implementing 6 different custom PCBs: The Primary board, Secondary board, USB to UART converter, temperature and humidity sensor, and two buck voltage regulators, 3.3v and 5v.

All voltage regulators are daughter boards, which is illustrated in Figure 8.2.4, and are connected to the main boards by header pins as illustrated in the bottom right corner for all systems. The USB to UART converter is implemented as a daughter board as well and is connected to the primary and secondary board via 3 wires, lastly, the temperature and humidity sensor is also implemented as a daughter board and is connected to the secondary board PCB.

8.2.1 Primary board PCB

The ESP32-S3 WROOM-1 N8R8 microcontroller on the primary board is placed along the PCB edge to optimize antenna performance and minimize RF interference. High-frequency components, such as capacitors, were kept away from the antenna area. Peripheral modules—including the ultrasonic sensor, camera, reed switch, and power regulation circuits—connect via header pins to facilitate debugging and modular replacement. All GPIO pins are routed to external headers, enabling flexible I/O reassignment and easier rework without PCB modifications. The board layout emphasizes modularity, maintainability, and reliable wireless performance.

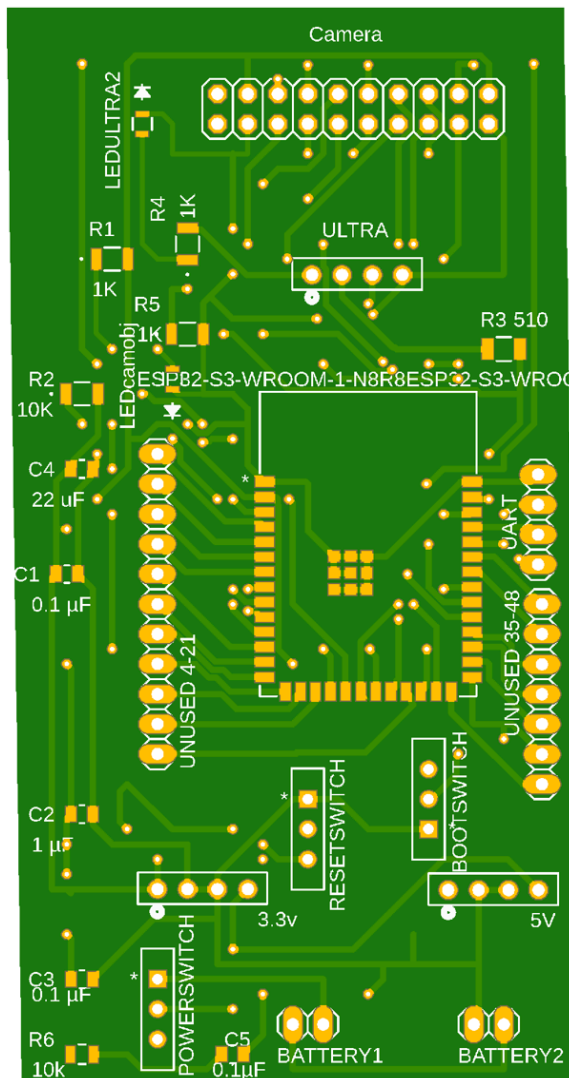


Figure 8.2.1 Primary board PCB layout

8.2.2 Secondary board PCB

The secondary device utilizes the ESP32-WROOM chip since this device needs less RAM and flash memory compared to the primary device. This device is responsible for managing the buzzer, a custom temperature and humidity sensor and a servo motor for the physical door control. Like the primary device the ESP32 chip is placed on the edge of the board to minimize the interference of other components with the wireless connectivity of the chip. The servo motor is essential on this board since it is responsible for physically pressing the button that alters the garage door status, similar to the primary device, the unused pins on the ESP32 chip to ease the rerouting of pins in case of any damages to the dedicated pins, all peripherals are plug in to minimize any damages to the board and ease debugging and maintenance process.

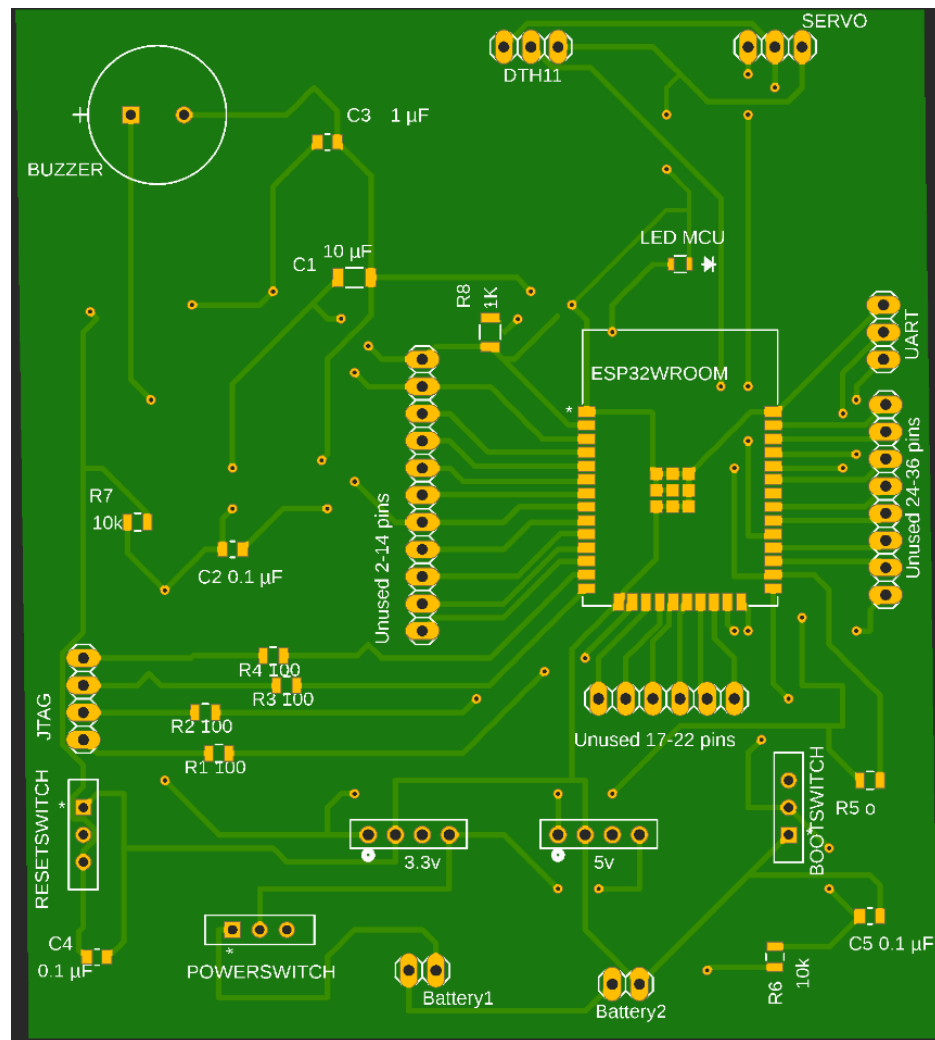


Figure 8.2.2 Servo system PCB layout

8.2.3 Voltage Regulator PCB (3.3V and 5V)

The DoorSure system employs two custom buck regulators, providing 3.3V and 5V outputs as daughter boards on the primary and secondary devices. The 3.3V regulator powers the ESP32 microcontrollers and most peripherals, while the 5V regulator supplies the servo motor on the secondary device. The 3.3V regulator is based on the AP63203, and the 5V regulator utilizes the AP63205, both selected for their high efficiency and reliable performance. Additionally we have developed backup regulators using LM2576 since they can be more reliable against loads, the LM2576 designs can be found in appendix C, but since the AP6320x regulators have smaller PCB footprint and reliable for our system, it was decided to use them for our project.

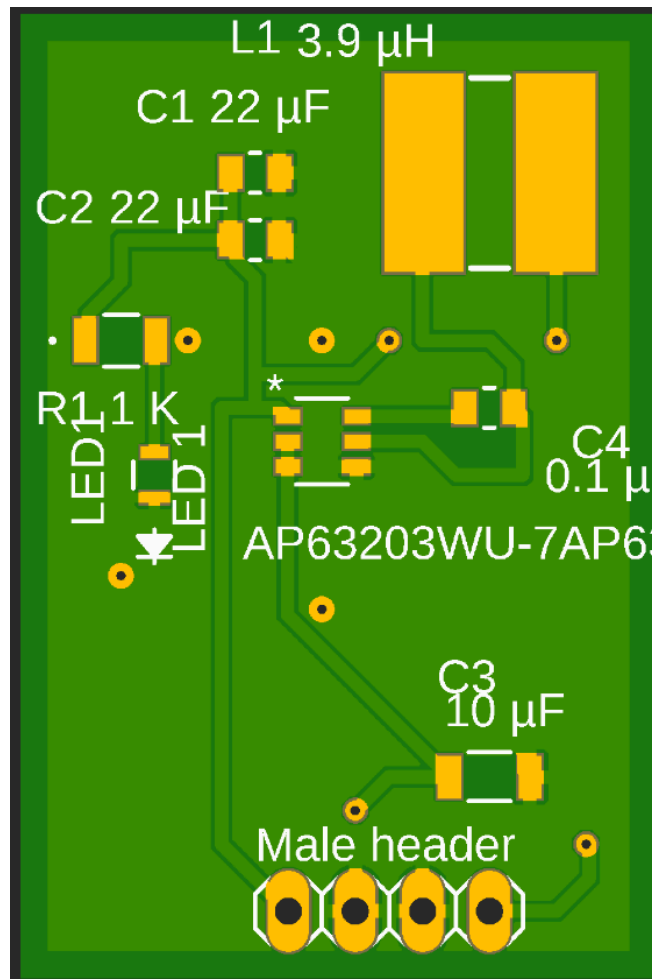


Figure 8.2.3.1 3.3V Voltage Regulator PCB layout

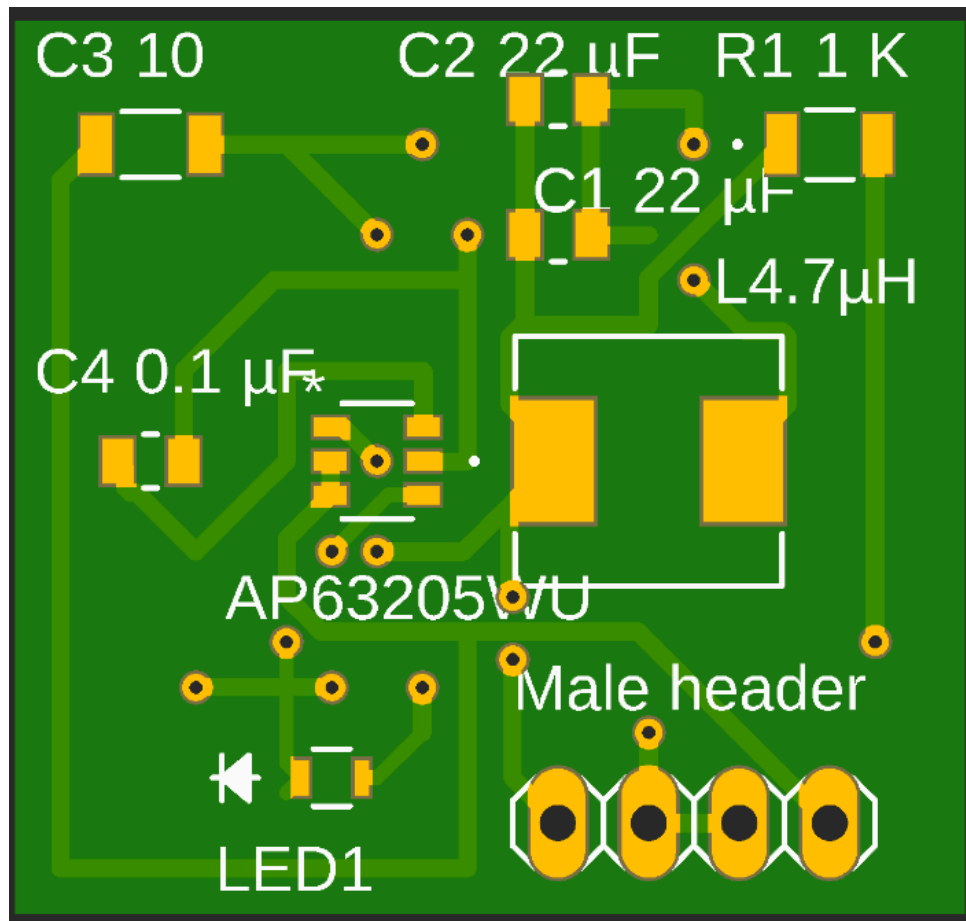


Figure 8.2.3.2 5V Voltage Regulator PCB layout

8.2.4 Temperature and Humidity sensor

The custom temperature and humidity sensor that was developed for this project is responsible for making sure that the environment of the garage won't compromise the overall system. The design of this board is minimal with only two components: The 10k resistor and DHT11 component itself. This is attached to the secondary device via header pins.

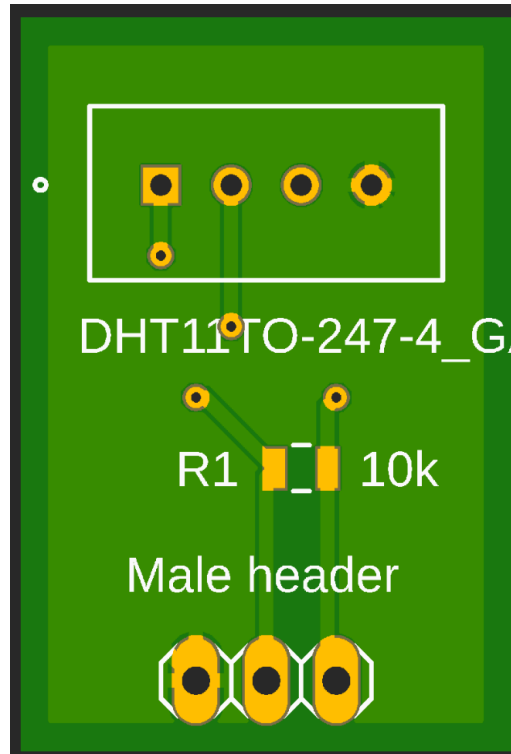


Figure 8.2.4 Temperature and humidity sensor PCB layout

8.2.5 USB to UART converter

The USB to UART converter is implemented on a daughter board. This decision was carefully chosen in consideration of time for soldering, installing and debugging the CP2102 chip on the board, as having the chip directly on the primary and secondary devices would increase complexity of sorting out problems on the main PCBs.

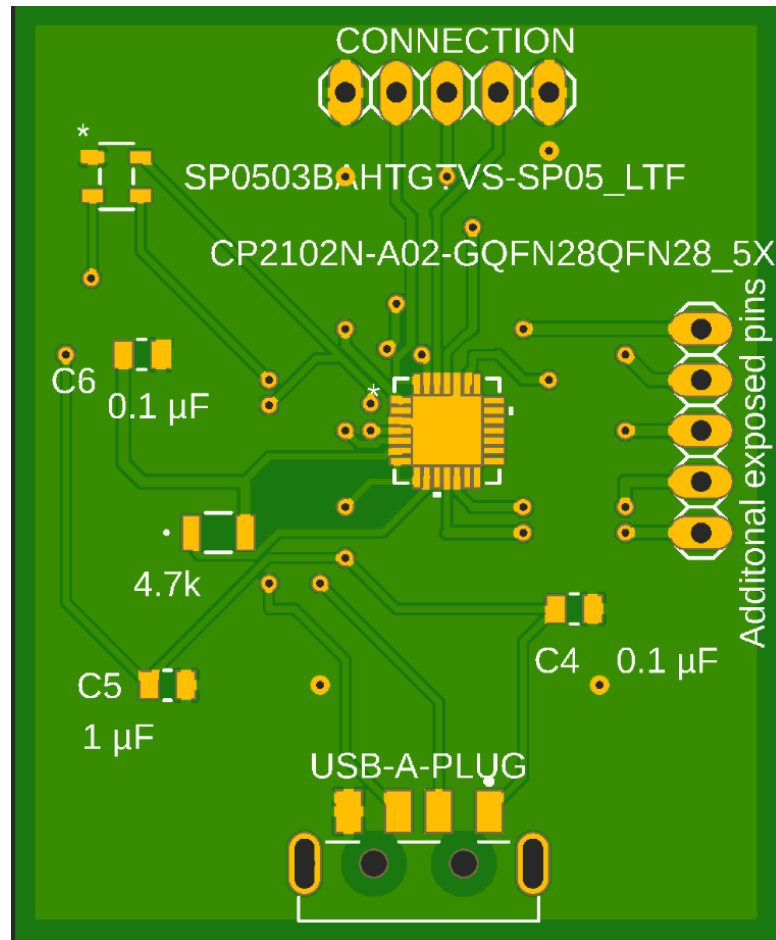


Figure 8.2.5 USB to UART converter PCB layout

8.3 Housing Design

After finalizing our PCB design, we proceeded to model the housing using the dimensions of the fully assembled primary and secondary devices. This was done through a combination of utilizing calipers for initial measurements and a tape measure for fine tuning throughout the housing design process. The housing design process went through many revisions and 3d prints before achieving the looks and appearances that we were satisfied with. The overall physical design and size of DoorSure needs to be non-intrusive looking and not have a bulky appearance. We plan on making the design for both the main and secondary device having a small form factor in mind so it does not stick out when being installed on the garage door. Handling the device should be the type where the user does not need to frequently troubleshoot the device, only needing to when the device's batteries are low.

The primary device's installation can be placed on the floor in the upright position without any use of adhesives, though if the user is afraid of the device being knocked over they can place some adhesives on the bottom of the housing to increase stability. The placement is crucial, as the device has to also be placed near the edge of the garage door in order for the camera and ultrasonic sensor to have the maximum view, range, and performance for their necessary functions. The housing of the primary device is designed to have a small form factor, being unobtrusive to its surroundings and being the size of around 2 Rubik's cubes. This has been achieved through the use of components that are not bulky or larger than our housing design. All components that are used for the primary devices are used in the housing.

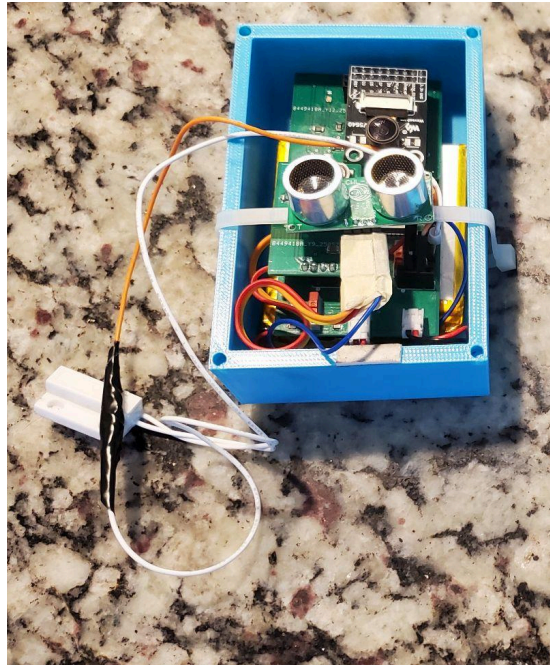


Figure 8.3.1 Front view of Primary Device with Housing

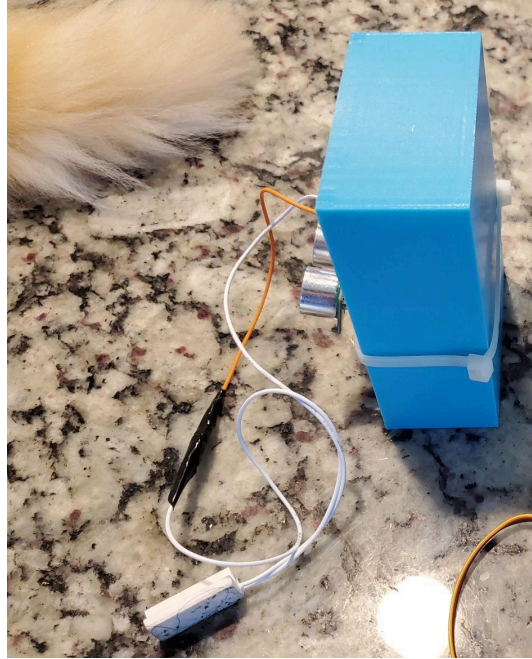


Figure 8.3.2 Side View of Primary Device with Housing

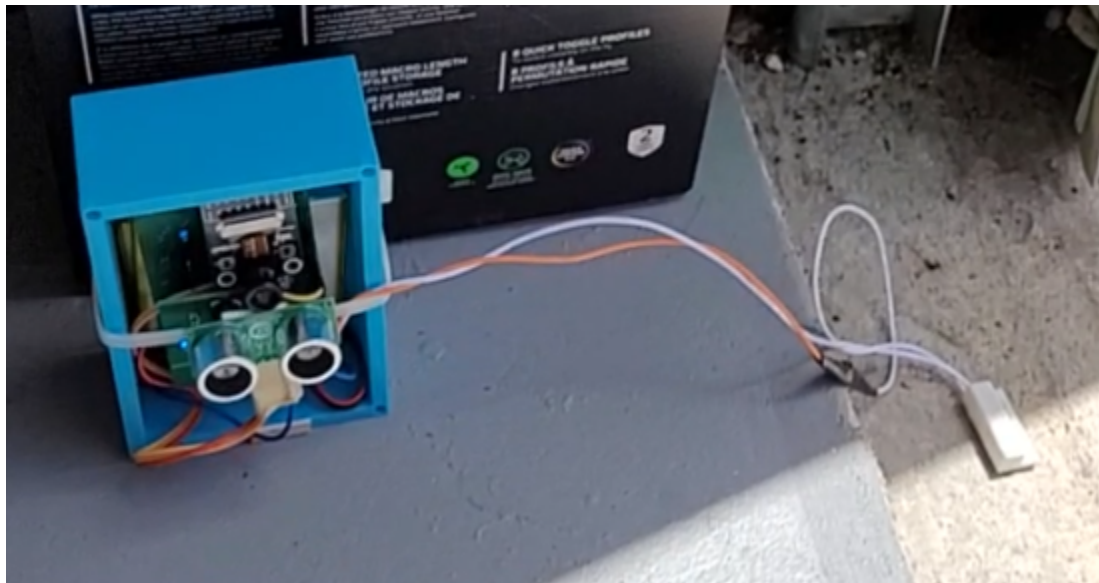


Figure 8.3.3 Mounting of the Primary Device with Housing

The secondary device's design also follows the design protocols with the primary device, only time being no larger than four Rubik's cubes due to the design of the PCB. Like with the primary device, we have achieved the housing design through the specific selection of components that does not intrude on

changing the size of the housing. The extended design of the servo motor creates flexibility of the mounting location of the motor, as garage door buttons do not arrive in the same design. As for mounting the secondary device's housing, mounting adhesive tape is applied on the back of the device to mount the device on the wall. While the risk of paint removal can occur if the user wants to reinstall the device on a different location, this can be mitigated by applying a layer of scotch tape on the wall, leaving no paint residue on the mounting tape as a result. The servo is also mounted using mounting tape, preferably a mountain tape with a strong weight hold to prevent the servo from falling off the mount.

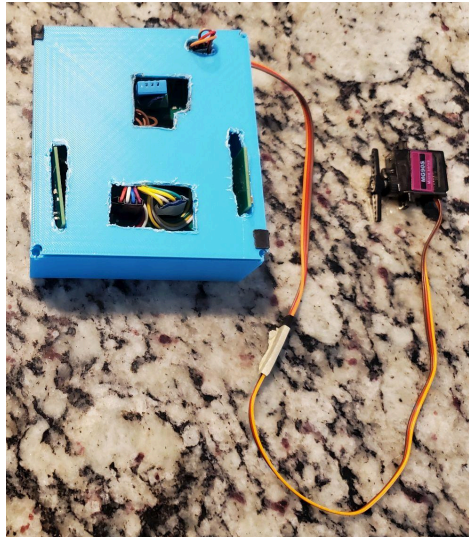


Figure 8.3.4 Front view of Secondary Device with Housing

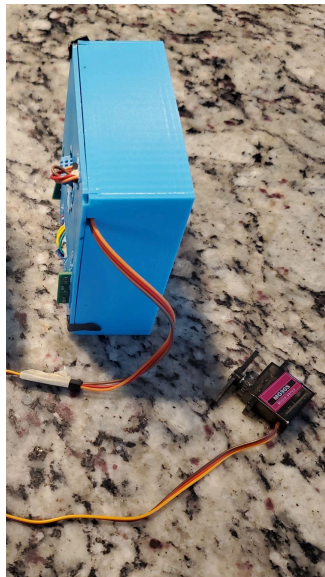


Figure 8.3.5 Side view of Secondary Device with Housing

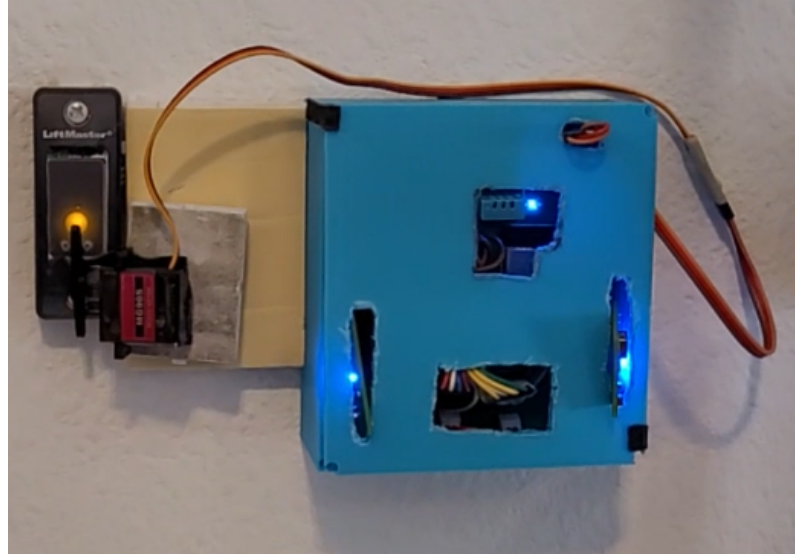


Figure 8.3.5 Mounting of the Secondary Device with Housing

Chapter 9. System Testing and Evaluation

9.1 Alter State of Garage Testing

The state of the garage door is determined using solely the reed switch component. One part of the reed switch is attached to the bottom of the garage door while the other is stuck to the floor of the garage, attached to the primary device ESP32. When doing initial testing, we powered the microcontroller through USB from our laptops. We successfully positioned the two components of the reed switch so that when they are in close proximity, the web app reflects the garage door as closed value '0'. When the components are separated, it reports the door as open value '1.' When the user presses "Open Door" on the web app, the servo motor activates and exerts force on the garage door button, opening the door. This causes the reed switch components to separate, marking the door state as "Open" which is then reflected in Firebase and subsequently on the web app.

During testing, we faced an obstacle when connecting the reed switch to the ESP 32 due to the weaker material of the reed switch wires. We mitigated this by

using a female to male jumper wire to connect it to a breadboard. Additionally, initially using data pins that lacked an IO caused the reed switch to debounce, making the web app reflect incorrect statuses of the garage door, improperly switching between open/closed despite no movement of the reed switch positions. Changing to a correctly handled IO data pin on the MCU resulted in the system working with 100% accuracy. No debounce errors occurred in our testing after the data pin change and the reed switches correctly identified the garage doors in a state of close or open.

We also connected the primary device with different networks to compare the delay between when the reed switch and magnet come/leave the proximity and when the status changes on the web application. The networks we chose were home WiFi and a WiFi hotspot. Although the hotspot connection was fairly fast and responsive, there were moments where the status change took around 3-5 seconds longer than expected and at times the status did not change at all. With the home wifi, the status changed immediately under both states. Ideally, the primary device should be connected to the network that is stable without any sudden drops and deliver high speed. Tests were performed with hotspots to mimic live demo conditions.

The servo system is responsible for pressing the garage door button once activated by the user via the web app, as demonstrated in our presentation videos under the documents tab of our website. Testing for this system involved determining the optimal placement of the servo motor to ensure it pressed the button at the correct angle and with appropriate torque—balancing between avoiding damage to the button and applying enough force for the garage door to respond. This was crucial for maintaining the overall reliability of our project. During testing, the servo was activated 100% of the time when the user pressed the button AND when there were no object obstructions. However, despite the movement occurring without fail, there were a few times where the servo was placed at a wrong angle, and therefore would not apply the necessary torque needed to press the garage door. It was crucial to set up and do enough testing before our live demo to ensure the servo was placed correctly to showcase the system.

9.2 Object Detection Testing

Despite getting highly accurate results for the object obstruction testings, there have been times where we've observed occasional inconsistencies in the ultrasonic distance sensor readings. Most of the time, the sensor consistently reports a distance of around 42 inches (for our testing garage length of 93in), which we associate with the presence of an object in the path of the garage. Occasionally though, the reading abruptly jumps to 95in, suggesting a sudden

disappearance of the object, before quickly reverting back to 42in. These inconsistencies typically occur within a single second or across just one or two cycles, indicating a brief glitch rather than a real change in the environment.

To address issues with inconsistent ultrasonic sensor readings, we implemented a data smoothing strategy. Specifically, the system only updated the `object_obstruction` value in Firebase—used to prevent the servo from activating when set to 1—if the last three consecutive ultrasonic readings fell within a ± 8 -inch range of each other. For example, a sequence such as 41, 44, and 43 inches was considered consistent and accepted, while a set like 42, 92, and 41 inches was deemed erratic and rejected. This filtering mechanism helped prevent momentary spikes or sensor noise from triggering false obstruction states, thereby improving overall system stability.

Additionally, we observed that when the `object_obstruction` variable remained at 0 (indicating no obstruction), the servo mechanism operated with 100% reliability. During testing, the servo consistently responded to button presses issued through the web application, activating within one second of receiving the Firebase trigger. There were no issues related to debouncing with the servo motor.

We also found that the ultrasonic sensor's performance could be disrupted by nearby electronic devices emitting electromagnetic interference. This interference sometimes resulted in inaccurate distance measurements. To mitigate this, we covered exposed electronics with foam, which helped absorb excess ultrasonic sound and shield the sensor from noise. With this adjustment, the ultrasonic sensor operated much more reliably.

9.3 Livestream Testing

During our livestream system testing, we encountered a limitation with the ESP32S3: it only allows live streaming over the local network. This means the user must be connected to the same Wi-Fi network as the ESP32S3 to view the live feed.

This limitation is partly due to the ESP32's use of HTTP for hosting the livestream, while Firebase operates over HTTPS. Modern browsers block insecure content HTTP from being loaded within secure HTTPS applications for security purposes, preventing the livestream from being embedded directly in the Firebase hosted web application.

Since the mini demo (for Senior Design 1) was conducted locally, with both the ESP32 and the user on the same network, we were able to successfully view the live feed.

During the livestream testing in Senior Design 2, we adopted an alternative approach by capturing images at regular intervals and uploading them to Firebase Storage. Our web interface then fetches and refreshes these images in sequence, creating the illusion of a live video stream. This method is fully compatible with HTTPS.

We also reduced the resolution to 1280 x 1024 pixels and limited frame rate to 5 frames per second to balance image clarity with responsiveness. In addition, we have learned that the camera with viewing any kind of motion can only occur with Google Chrome. Any other browser only shows a still image, but the AI detection works regardless of browser choice.

9.4 Object Identifier Testing

The ESP32S3 development board was used to identify objects obstructing the garage door, an essential function in our project. When the ultrasonic sensor detects an obstruction, the system prevents the servo motor from moving, thereby stopping the garage door from closing if the user attempts to do so via the web app. A message is displayed on the application to alert the user, along with an option to view a snapshot of the obstruction. The system then attempts to identify the object with high confidence using our trained AI model.

Previously, our AI model was trained to recognize three specific objects: a soda can, a measuring tape, and a small dog ball. However, during testing—particularly while filming the mini demo—we encountered some challenges. Since the demo was recorded during the day, strong daylight created glare on the camera, which negatively affected the model's object recognition accuracy. This issue was further compounded by the relatively small size of the target objects as well as the differences in the testing images compared to the actual testing setting.

To fix the issue we had with Edge Impulse, we utilize Google Gemini, a cloud-based AI model. The ESP32S3 captures an image and uploads it to Firebase, then analyzed via Gemini API. The model returns a natural-language description of the detected object which fits better for our needs.

9.5 Power Testing

Prior to prototype testing, we used power from our selected battery source to quickly mock up the system using a breadboard setup. During this early stage, we included an LCD screen we had on hand to monitor the battery performance. We observed that the battery drained at a steady, slow rate when powering the primary device system, indicating a reasonable baseline for power consumption. However, further testing revealed that the AI Thinker ESP CAM was the primary contributor to excessive power drain, due to its continuous image capture for object detection and obstruction analysis.

This insight has led us to reconsider how and when the camera should be powered. We plan to revise the code for the AI Thinker module when switching to a new camera component during PCB fabrication to support more efficient low power modes. For instance, the camera would remain off by default and only activate when a user specifically accesses the web app to check for obstructions.

Throughout prototype testing, USB power from laptops was used to supply the servo and the main system. For the object detection system, a power bank was used to power the ESP32 development board. These boards include onboard voltage regulators, allowing them to reliably power connected components without needing direct USB power once the code has been uploaded. This setup enabled consistent operation of the object detection camera through the ESP32's onboard power.

However, during testing, we observed that the AI Thinker ESP32-CAM significantly drained the power bank, dropping from 100% to 40% within just a few hours of continuous use. This highlighted the need for better power management, especially for battery-operated IoT devices. We have explored different approaches to prolong the device battery to at least 3 months of use, as per our specifications table.

For Senior Design II, we have switched to the OV5640 breakout camera component which connects directly to the primary device, which significantly lowers the power usage. Additionally, to maximize efficiency, the OV5640 is responsible for the AI obstacle classification and live feed which is only activated when the user requests the camera to start streaming or AI detects. This approach lowers the power consumption which led us to have the primary and secondary board powered by two 3.7V 556065 lipo batteries for each device; that helped us meet our 3 months battery life requirement.

9.6 Senior Design II Final Testing

As we transition into Senior Design II, our primary focus is on refining and finalizing our project to ensure it is reliable, polished, and not only meets but excels above our basic and advanced goals. With the PCB designs nearly complete and ready for fabrication pending advisor review, our immediate goal is to fine-tune these designs with the guidance of our advisors. This includes reviewing component placement, trace routing, and ensuring compatibility with our chosen hardware modules.

One key upgrade in our hardware plan is the transition from the ESP32-CAM (used during our prototyping and testing phase) with the OV2640 module to a more integrated and compact solution. For the final version, we used the OV5640 camera module, which connects directly to the ESP32 chip on our custom PCB. This change is necessary to meet fabrication and form factor requirements, and we will allocate additional time to test and verify this new integration.

Additionally, we investigated secure methods for streaming live video from our camera module to the Firebase-connected web app. Since the ESP32 streams over HTTP and Firebase require HTTPS for secure content delivery, instead of continuous streaming. The ESP32 captures images and uploads them to Firebase Storage. Our web interface then fetches and refreshes these images to create the illusion of a live video stream.

To support the system demonstration in a realistic way during our final live demo, we installed the system on an actual garage door which allowed us to showcase the full functionality of our system—including the servo mechanism, object detection, reed switch reporting, and user control via the web app. Alongside this, we have developed and fabricated custom enclosures for each system component, taking into account both PCB dimensions and the layout of the garage model.

We also built multiple backup boards for each subsystem to ensure redundancy and prevent hardware failure from affecting our final presentation.

Chapter 10. Administrative Content

10.1 Budget

After researching the project and discussing the allocated budget, our team agreed that each member could spend up to \$200, which would be divided among the components listed in the bill of materials (BOM). With four team members, our total budget came to \$800. The significant gap between the actual amount spent on components and the allocated funds gave us the flexibility to ensure faster delivery times for components and PCBs. This buffer also proved invaluable when some components had to be repurchased due to damage during testing and assembly. Ultimately, the extra room in our budget allowed us to explore more creative and innovative solutions as challenges arose throughout the project.

10.2 Bill of materials

Table 10.2: Bill of Materials

Components	Details	Vendor	Quantity	Unit price	Total price
Microcontroller	ESP 32s wroom 2.4 GHZ dual mode WIFI + bluetooth dual-core microcontroller integrated with antenna RF AMP filter	Amazon	3	\$8.99	\$26.96
Microcontroller	ESP32 S3 Development Board N8R8	Amazon	2	\$7.59	\$15.19
Ultrasonic Sensor	Ultrasonic Sensor module	Amazon	1	\$6.99	\$6.99

Camera	Waveshare OV5640 Camera Board	Amazon	1	\$24.99	\$24.99
Reed switch	4 pcs door sensor magnetic switch	Amazon	1	\$7.99	\$7.99
Servo motor	MG90 180-degree servo motor 4PK	Amazon	1	\$13.99	\$13.99
Temperature and Humidity Sensor	DHT11 Sensors	Amazon	1	\$9.99	\$9.99
Replaceable batteries	556065 3.7V Lipo battery 2500 mAh	Amazon	2	\$11.49	\$22.98
Battery charger	3.7V Lipo Battery Charger	Amazon	1	\$5.99	\$5.99
PCBs	Printed circuit boards	JLCPCB	1	\$195	\$195
Other Components	Resistors, capacitors, inductors and regulators	Digikey	1	\$250	\$250
Total:					\$585

10.3 Work Distribution

Initially, we held multiple meetings to evaluate each team member's personal and technical strengths. Based on these discussions, we created a work distribution table that aligned tasks with our respective skill sets. As students managing different class and work schedules, we emphasized flexibility and collaboration from the beginning. Throughout the project, we remained committed to supporting one another—stepping in to assist with tasks when needed, while still holding each member accountable for steady progress toward our milestones.

Now that we have completed the project, we can confidently say that this collaborative approach played a critical role in our success. We maintained our commitment to regular communication by holding two meetings each week, which allowed us to monitor progress, redistribute tasks as needed, and ensure that everyone remained on track.

Our team consisted of two Electrical Engineering students and two Computer Engineering students, which naturally influenced the initial division of hardware and software tasks. However, to challenge ourselves and grow as engineers, we intentionally included a mix of responsibilities across disciplines. This approach not only enhanced our technical skills but also allowed us to demonstrate the breadth of knowledge we've gained throughout our time at the University of Central Florida.

Table 10.3: Work Distribution

Hardware		
Task	Primary	Secondary
Prototype Design	All	All
PCB Design	Mina Ghandour	Khang Nguyen
Power System Engineering	Khang Nguyen	Mina Ghandour
Housing Design	Khang Nguyen	Mina Ghandour
Microcontroller Selection	Khang Nguyen	Felix Liu
Microcontroller Implementation	Andrea Rosa	Mina Ghandour
Battery Selection	Mina Ghandour	Khang Nguyen
Battery Integration	Khang Nguyen	Mina Ghandour
Camera Selection	Mina Ghandour	Felix Liu
Camera Integration	Felix Liu	Khang Nguyen
Reed Switch and Temp/Hum Integration	Khang Nguyen	Andrea Rosa

Servo Motor Integration	Andrea Rosa	Mina Ghandour
Software		
Task	Primary	Secondary
Cloud Server Management	Andrea Rosa	Felix Liu
WebSocket Integration	Andrea Rosa	Felix Liu
Full Web App RTDB Management	Andrea Rosa	Khang Nguyen
Ultrasonic and Camera RTDB Management	Felix Liu	Andrea Rosa
Object Obstruction System	Felix Liu	Andrea Rosa
AI Identification	Felix Liu	Mina Ghandour
User Authentication	Andrea Rosa	Felix Liu
Notification System	Andrea Rosa	Mina Ghandour
UI Design	Mina Ghandour	Andrea Rosa
UI Implementation	Andrea Rosa	Felix Liu
Senior Design Project Website	Andrea Rosa	Mina Ghandour

10.4 Milestones

Below are the team's hard deadlines for Senior Design I and Senior Design II. These milestones were initially generated based on the course outline. Throughout the project, we closely followed these dates to ensure we remained on track—especially given the accelerated pace of the summer semester. Meeting these deadlines was essential to our ability to complete the course with a functional, finished prototype.

While no specific dates initially were established in advance with our review committee, we made it a priority to keep our reviewers informed of our progress through regular updates. This helped maintain alignment and ensure feedback

could be incorporated in a timely manner. It's also worth noting that the milestones for Senior Design II were intentionally broad at the start, as the exact timeline and workload became clearer once the second phase of the project began. Testing and finishing of the final PCBs along with our last bits of documentation were our highest priority in Senior Design II. We ensured to stay on top of the deadlines specified in the end of semester lectures and kept in constant contact with our reviewers to ensure an easy live demonstration.

Table 10.4.1: Milestones for Senior Design I

Task		Start	End	Status
1	Team Formation	Week 1		Completed
2	Project Ideas	Week 2		Completed
3	Divide & Conquer	1/17/2025	1/23/2025	Completed
4	ABET Lectures	1/21/2025	2/13/2025	Completed
5	Divide & Conquer Meeting	1/28/2025		Completed
6	Create and Update the Group Website	1/28/2025	2/7/2025	Completed
7	ABET Quizzes (A to G)	2/8/2025	3/7/2025	Completed
8	Spring Break	3/17/2025	3/22/2025	N/A
9	Midterm Report	2/7/2025	3/25/2025	Completed
10	Midterm Report Meeting	3/26/2025	3/28/2025	Completed
11	Update Group Website With Midterm Report	3/28/2025	4/4/2025	Completed
12	Final Report & Mini Demo Video	4/4/2025	4/22/2025	Completed

Table 10.4.2: Milestones for Senior Design II

Task		Start	End	Status
1	Build Prototype	May 2025		Completed
2	Software Build	June 2025		Completed
3	Testing Phase	July 2025		Completed
4	Final Presentation	7/24/2025		Completed
5	Final Report	7/29/2025		Completed

Chapter 11. Conclusion

The completion of our Senior Design I and II projects marks the successful development and demonstration of DoorSure, a comprehensive smart garage system that integrates multiple hardware and software components into a cohesive and functional product. This project provided us with invaluable experience and practical skills directly relevant to our future engineering careers as we approach graduation from UCF.

Throughout the development process, we applied our knowledge in embedded systems, IoT architecture, cloud-based services, power management, and custom PCB design to create a prototype that meets key functional requirements such as garage door status monitoring, obstruction detection and AI-based identification, servo-driven door actuation, and live camera streaming. Each subsystem was carefully selected and integrated to ensure a balance between cost efficiency, reliability, and user-centered design. Central to our architecture is the ESP32 microcontroller family, chosen for its native support of Wi-Fi and Bluetooth, which enabled real-time data exchange within the Firebase cloud ecosystem. The microcontrollers interface with peripherals including a reed switch, an ultrasonic sensor, and a servo motor to provide precise door control and obstruction sensing. Cloud-hosted Firebase databases acted as the communication bridge, facilitating secure, low-latency synchronization between hardware, software, and the user interface.

A significant technical challenge addressed during the project involved navigating the limitations imposed by HTTP/HTTPS security protocols when attempting to implement real-time video streaming from the ESP32-CAM

module. Due to the lack of native HTTPS support on the microcontroller and the security restrictions of modern browsers that prevent HTTP-based streams from being rendered within secure (HTTPS) web environments, we were unable to directly embed a true live feed into our application. This constraint led us to experiment with multiple alternative methods of delivering camera footage to the web interface. After testing and evaluating the available options, we ultimately adopted a solution that periodically captures still images from the ESP32S3 and uploads them to Firebase Storage. These images are then displayed sequentially in the web application to emulate the appearance of a live stream. While not a true video feed, this approach allows users to remotely monitor garage activity with near-real-time feedback, all within the security and compatibility constraints of a modern HTTPS-enabled web application.

On the software side, Firebase's Firestore and Realtime Database services were instrumental in enabling low-latency, bidirectional communication between the user interface and the embedded devices. While live video streaming from the ESP32 was constrained by HTTPS restrictions, we devised a method of capturing many images and turn them into a live stream.

The AI-based object recognition module, a key stretch goal, was initially implemented by training a lightweight model to identify common garage door obstructions. Limited by hardware constraints such as camera resolution and lighting conditions, this feature was then transformed into using Google Gemini which significantly enhances the system's safety and usability, setting DoorSure apart from existing commercial products.

During Senior Design II, we transitioned to the OV5640 camera module to enable custom PCB integration, which required significant firmware revisions and careful PCB design refinements under faculty guidance. Our focus on electrical reliability, thermal management, and trace routing improved the overall system robustness and manufacturability. Additional improvements included optimizing AI accuracy, reducing power consumption through sleep modes, enhancing mobile responsiveness of the web application, and extensive testing to ensure performance across varied environmental conditions. These advancements collectively strengthened DoorSure's viability as a practical and market-competitive smart garage solution.

In conclusion, our work in Senior Design I and II established a solid foundation in embedded systems, cloud integration, and IoT product development. The lessons learned and challenges overcome throughout the project have prepared us to face real-world engineering problems and have positioned us to further develop DoorSure toward a commercially viable product.

APPENDIX A - References

General Statistics and Market Research

- [1] FBI — Property Crime Statistics.
- [2] US Burglary Statistics 2024.
- [3] CHAMBERLAIN Smart Garage Control Product.
<https://www.amazon.com/dp/B08GD3D9YJ>
- [4] CHAMBERLAIN myQ Smart Indoor Security Camera.
<https://www.amazon.com/dp/B0D2LW9M4P>
- [5] Genie Aladdin Connect Smart Garage System.
<https://www.amazon.com/dp/B016QB7K5O>

Technical Articles & Cloud Services

- [6] “Pros and Cons of Cloud Storage.” Secure Storage.
- [7] IPC-2221A Design Standards.
<https://www.ipc.org/TOC/IPC-2221A.pdf>
- [8] JLCPCB Capabilities Overview.
<https://jlcpcb.com/capabilities/pcb-capabilities>
- [9] IEEE: The Evolution of Wi-Fi Technology.
<https://standards.ieee.org/beyond-standards/the-evolution-of-wi-fi-technology-and-standards/>

Microcontroller & Hardware Datasheets

- [10] ESP32-WROOM-32 Datasheet.
https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf
- [11] ESP32-CAM v1.6 Board Reference.
https://github.com/SeeedDocument/forum_doc/blob/master/reg/ESP32_CAM_V1.6.pdf
- [12] ESP32-CAM Module Datasheet.
<https://www.handsontec.com/dataspecs/module/ESP32-CAM.pdf>
- [13] MG90S Servo Motor Datasheet.
https://www.electronicoscaldas.com/datasheet/MG90S_Tower-Pro.pdf
- [14] DFR0602 Sensor Board Datasheet.
https://media.digikey.com/pdf/Data%20Sheets/DFRobot%20PDFs/DFR0602_Web.pdf
- [15] CP2102 USB to UART Bridge Datasheet.
<https://www.silabs.com/documents/public/data-sheets/CP2102-9.pdf>

Technical Education & Sensor References

- [16] Beaud, D. "LiDAR: What Is It and How Does It Work?" YellowScan.
<https://www.yellowscan.com/knowledge/how-does-lidar-work/>
- [17] CD-Team. "IR Sensor Basics." Electronics For You, 28 Aug 2023.
<https://www.electronicsforu.com/technology-trends/learn-electronics/ir-led-infra-red-sensor-basics>
- [18] "A Complete Guide to Reed Switches." RS Components, 1 Feb 2023.
<https://uk.rs-online.com/web/content/discovery/ideas-and-advice/reed-switches-guide>
- [19] JU. "Hall Effect – Principle, Theory, Formula, Applications & FAQs." BYJU'S.
<https://byjus.com/physics/hall-effect/>
- [20] MaxBotix Inc. "How Ultrasonic Sensors Work."
<https://maxbotix.com/blogs/blog/how-ultrasonic-sensors-work>
- [21] Monolithic Power Systems, Inc. "Types and Functions of Sensors in Automotive Systems."
<https://www.monolithicpower.com/en/learning/mpscholar/automotive-electronics/automotive-sensing-and-actuators/types-and-functions-of-sensors>
- [22] Sparks, G. "Database Modelling in UML." Methods & Tools.
<https://www.methodsandtools.com/archive/archive.php?id=9>

APPENDIX B - Parts

[1] Servo Motor Part

https://www.amazon.com/DIYables-Degree-Arduino-ESP8266-Raspberry/dp/B0BPFXTZ73/ref=asc_df_B0BPFXTZ73?mcid=6caed98ed8e23609961ab0c9ef94c526&hvocijdid=12273598725803836168-B0BPFXTZ73-&hvexpln=73&tag=hyprod-20&linkCode=df0&hvadid=721245378154&hvpos=&hvnetw=g&hvrnd=12273598725803836168&hvpone=&hvptwo=&hvqmt=&hvdev=c&hvdvcmdl=&hvlocint=&hvlocphy=9011767&hvtargid=pla-2281435176658&th=1

[2] AA Duracell replaceable batteries

https://www.amazon.com/dp/B00000JHQ6/ref=twister_B0CX2ZM9CF?encoding=UTF8&th=1

[3] DoorSensor

https://www.amazon.com/DIYables-Magnetic-Arduino-ESP8266-Raspberry/dp/B0B3D7BM4K/ref=asc_df_B0B3D7BM4K?mcid=3da3e052d4453e6d825a02bcd1e45e49&hvocijdid=187620812624435686-B0B3D7BM4K-&hvexpln=73&tag=hyprod-20&linkCode=df0&hvadid=721245378154&hvpos=&hvnetw=g&hvrnd=187620812624435686&hvpone=&hvptwo=&hvqmt=&hvdev=c&hvdvcmdl=&hvlocint=&hvlocphy=9011767&hvtargid=pla-2281435177818&psc=1

[4] UltrasonicSensor

https://www.amazon.com/HC-SR04-HC-SR04P-Ultrasonic-Distance-Measuring/dp/B07KNTQ4C2/ref=asc_df_B07KNTQ4C2?mcid=9601b6d8b1df3f319306a5d594025b6e&hvocijdid=14937213940166474350-B07KNTQ4C2-&hvexpln=73&tag=hyprod-20&linkCode=df0&hvadid=721245378154&hvpos=&hvnetw=g&hvrnd=14937213940166474350&hvpone=&hvptwo=&hvqmt=&hvdev=c&hvdvcmdl=&hvlocint=&hvlocphy=9057291&hvtargid=pla-2281435178138&psc=1

[5] ESP-32S Development Board (MCU)

https://www.amazon.com/ESP-WROOM-32-Development-Dual-Mode-Microcontroller-Integrated/dp/B07WCG1PLV/ref=asc_df_B07WCG1PLV?mcid=7cc24e50dc423e8e9f1039421f865b8a&hvocijdid=3062641011911185649-B07WCG1PLV-&hvexpln=73&tag=hyprod-20&linkCode=df0&hvadid=721245378154&hvpos=&hvnetw=g&hvrnd=3062641011911185649&hvpone=&hvptwo=&hvqmt=&hvdev=c&hvdvcmdl=&hvlocint=&hvlocphy=9011767&hvtargid=pla-2281435177418&th=1

[6] ESP32-CAM-MB 5V WiFi Bluetooth

https://www.amazon.com/ESP32-CAM-ESP32-CAM-MB-Bluetooth-Development-CH340G/dp/B0CBQ7DDM8/ref=asc_df_B0CBQ7DDM8?mcid=e0c8b396601

[e33dd9dbe848de410cfea&hvocijdid=758869657899358644-B0CBQ7DDM8-&hve_xpln=73&tag=hyprod-20&linkCode=df0&hvadid=721245378154&hvpos=&hvnetw=g&hvrnd=758869657899358644&hvpone=&hvptwo=&hvqmt=&hvdev=c&hvdvcmdl=&hvlocint=&hvlocphy=9057291&hvtargid=pla-2281435177818&psc=1](https://www.amazon.com/dp/B0CBQ7DDM8?hve_xpln=73&tag=hyprod-20&linkCode=df0&hvadid=721245378154&hvpos=&hvnetw=g&hvrnd=758869657899358644&hvpone=&hvptwo=&hvqmt=&hvdev=c&hvdvcmdl=&hvlocint=&hvlocphy=9057291&hvtargid=pla-2281435177818&psc=1)

[7] Colored LEDs

[Amazon.com: \(100 Pcs\) MCIGICM 5mm LED Light Diodes, LED Circuit Assorted Kit for Science Project Experiment \(Multi-Colored - 5 Color\) : Industrial & Scientific](https://www.amazon.com/dp/B07GKJG1K1?hve_xpln=73&tag=hyprod-20&linkCode=df0&hvadid=721245378154&hvpos=&hvnetw=g&hvrnd=758869657899358644&hvpone=&hvptwo=&hvqmt=&hvdev=c&hvdvcmdl=&hvlocint=&hvlocphy=9057291&hvtargid=pla-2281435177818&psc=1)

Camera development board

<https://www.adafruit.com/product/4095>

<https://www.adafruit.com/product/5955>

<https://www.aliexpress.com/item/32971057846.html>

https://www.amazon.com/dp/B0CX93M5DW/ref=sspa_dk_detail_1?psc=1&pd_rd_i=B0CX93M5DW&pd_rd_w=bnyNA&content-id=amzn1.sym.7446a9d1-25fe-4460-b135-a60336bad2c9&pf_rd_p=7446a9d1-25fe-4460-b135-a60336bad2c9&pf_rd_r=TCX8W2RAC75ZEV0N69FZ&pd_rd_wg=lxlpP&pd_rd_r=9331f95f-ce24-4a98-bbda-07eb9f03fa55&s=photo&sp_csd=d2lkZ2V0TmFtZT1zcF9kZXRhaWw

https://www.amazon.com/Seeed-Studio-XIAO-ESP32-Sense/dp/B0C69FFVHH/ref=dp_prsubs_d_sccl_1/133-6285833-1780855?pd_rd_w=kpYYq&content-id=amzn1.sym.3ad0ccdf-fd9f-4ec9-a400-2b1165fdcd58&pf_rd_p=3ad0ccdf-fd9f-4ec9-a400-2b1165fdcd58&pf_rd_r=TCX8W2RAC75ZEV0N69FZ&pd_rd_wg=trlfY&pd_rd_r=6626049d-9130-4dee-acc2-0a208f942ec3&pd_rd_i=B0C69FFVHH&th=1

Camera modules

https://www.amazon.com/Acxico-OV2640-Wide-Angle-Million-Interface/dp/B09DSZDCTX/ref=asc_df_B09DSZDCTX?mcid=8820535597353f8192dda29338d8c3cd&hvocijdid=4629284419412209855-B09DSZDCTX-&hvexpln=73&tag=hyprod-20&linkCode=df0&hvadid=730352155585&hvpos=&hvnetw=g&hvrnd=4629284419412209855&hvpone=&hvptwo=&hvqmt=&hvdev=c&hvdvcmdl=&hvlocint=&hvlocphy=9057291&hvtargid=pla-2281435177858&psc=1

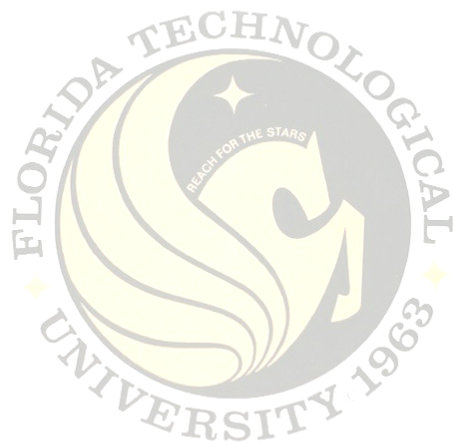
https://www.amazon.com/Acxico-OV2640-Wide-Angle-Million-Interface/dp/B09DSZDCTX/ref=asc_df_B09DSZDCTX?mcid=8820535597353f8192dda29338d8c3cd&hvocijid=4629284419412209855-B09DSZDCTX-&hvexpln=73&tag=hyprod-20&linkCode=df0&hvadd=730352155585&hvpos=&hvnetw=g&hvrnd=4629284419412209855&hvpone=&hvptwo=&hvqmt=&hvdev=c&hvdvcmidl=&hvlocint=&hvlocphy=9057291&hvtargid=pla-2281435177858&psc=1

https://www.ebay.com/itm/394461368220?chn=ps&mkevt=1&mkcid=28&google_free_listing_action=view_item&srsltid=AfmBOorpglYxkWtnIEJ1Pt8cWfRWSebc-ENZ5UrcLODkEi8wcL-ZGgGrhgk&gQT=2

https://www.ebay.com/itm/335617990655?chn=ps&trkparms=isp_r%3D1&amdata=enc%3A17qftNN8zSACBwVsSenjsuQ34&norover=1&mkevt=1&mkrid=711-117182-37290-0&mkcid=2&mkscid=101&itemid=335617990655&targetid=2320093655185&device=c&mkttype=pla&googleloc=9057291&poi=&campaignid=21676663813&mkgroupid=175573447188&rlsatarget=pla-2320093655185&abclid=10012304&merchantid=101681819&gad_source=1&gclid=CjwKCAjwvr--BhB5EiwAd5YbXqjR0RGnOY-kwcHpHAIWSZufxXYeCg1BcSH9k0Sq4Heypm9PXilJQxoCfJ4QAvD_BwE

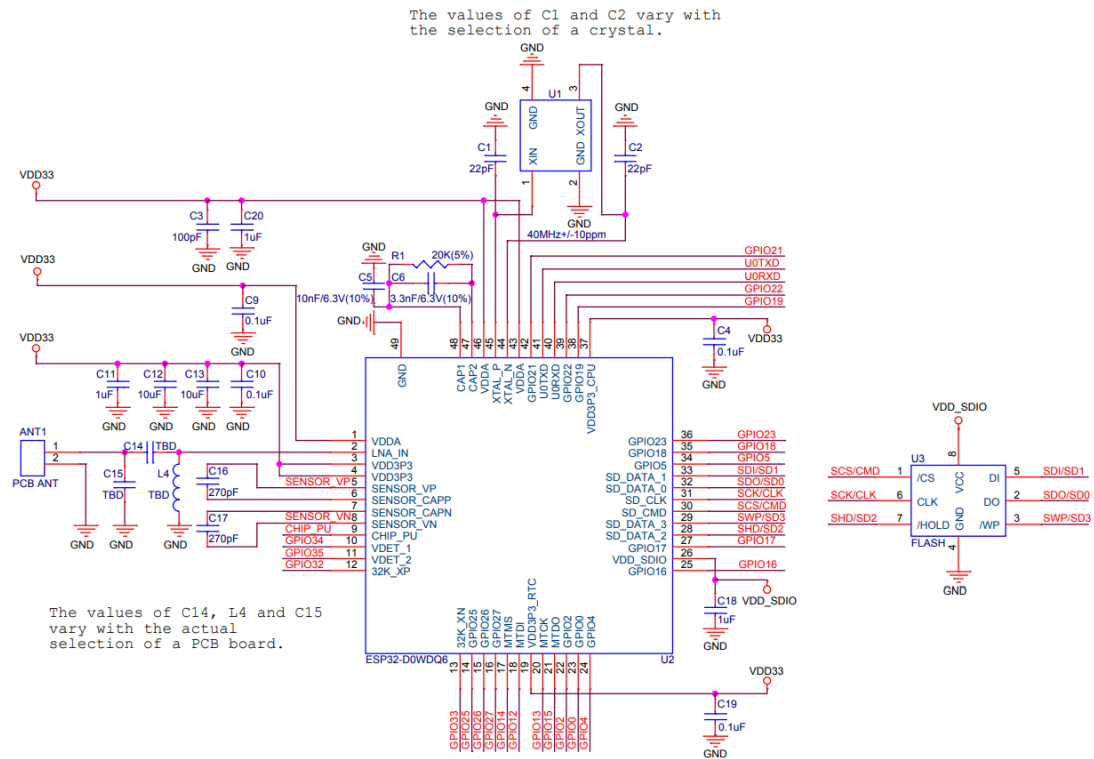
https://www.aliexpress.us/item/3256806050145546.html?src=google&pdp_npi=4%40dis%21USD%214.72%214.72%21%21%21%21%21%40%2112000036408798141%21ppc%21%21%21&src=google&albch=shopping&acnt=708-803-3821&isdl=y&slnk=&plac=&mtctp=&albbt=Google_7_shopping&aff_platform=google&aff_short_key=UneMJZVf&gclsrc=aw.ds&albagn=888888&ds_e_adid=&ds_e_matchtype=&ds_e_device=c&ds_e_network=x&ds_e_product_group_id=&ds_e_product_id=en3256806050145546&ds_e_product_merchant_id=5295628431&ds_e_product_country=US&ds_e_product_language=en&ds_e_product_channel=online&ds_e_product_store_id=&ds_url_v=2&albcpr=20542171673&albag=&isSmbAutoCall=false&needSmbHouyi=false&gad_source=1&gclid=Cj0KCQjw4cS-BhDGARIsABg4_J3R1nI3f9pUUD5bPUe5Bu7yGQe2qRuB4Qx9N7x1ICfZVezF_OE_NygaAvm3EALw_wcB&gatewayAdapt=glo2usa

<https://www.amazon.com/Taidacent-Interface-Sensor-NT99141-fisheye/dp/B097872PWY?th=1>

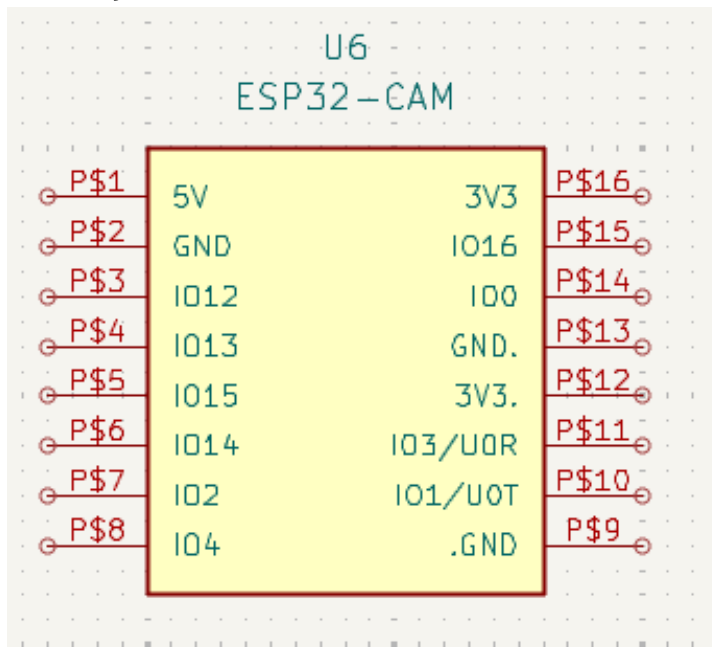


APPENDIX C - Data sheet

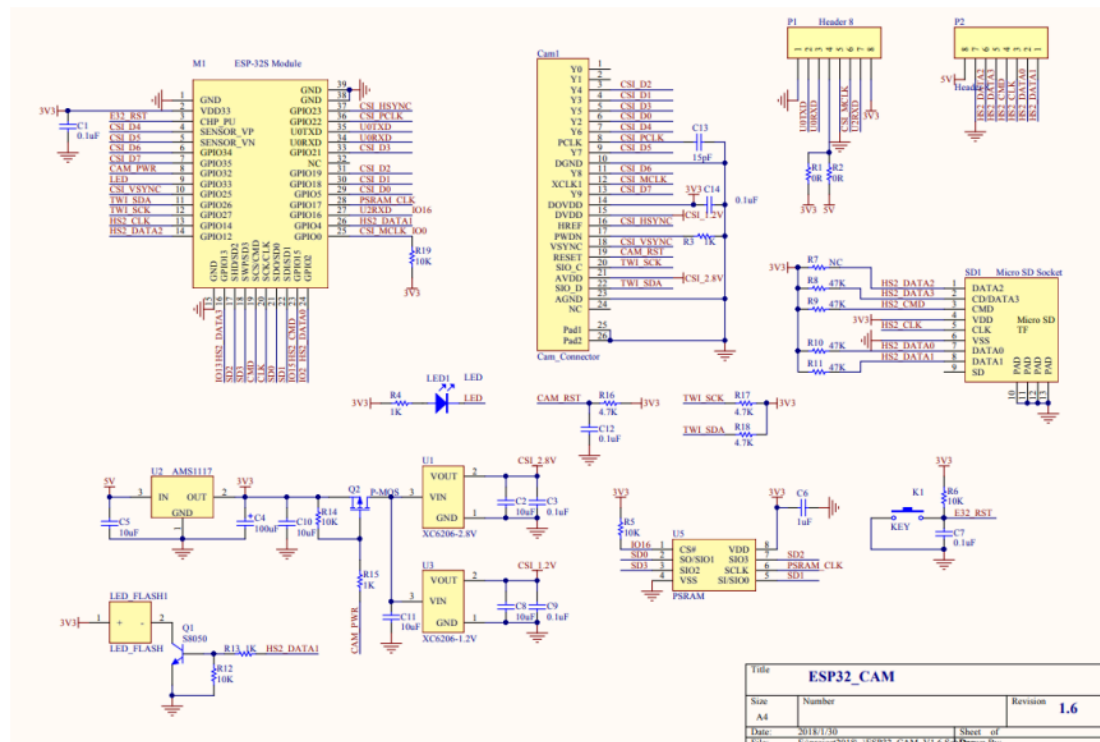
6.1.1: Schematics of ESP32-WROOM-32 from Espressif



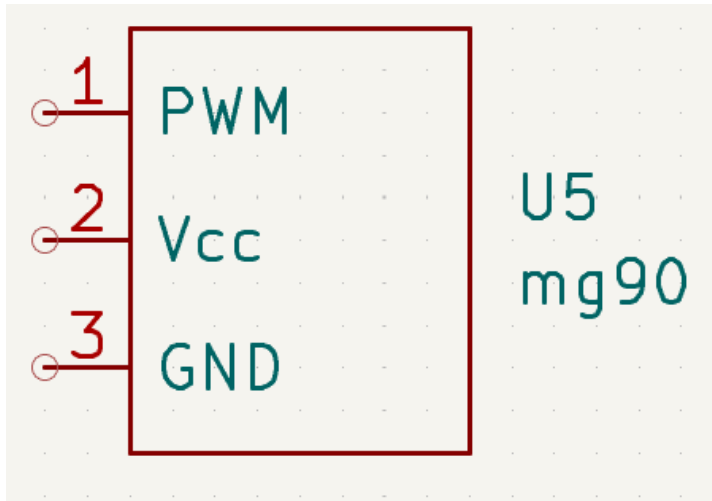
6.1.2: Symbol of ESP32-CAM



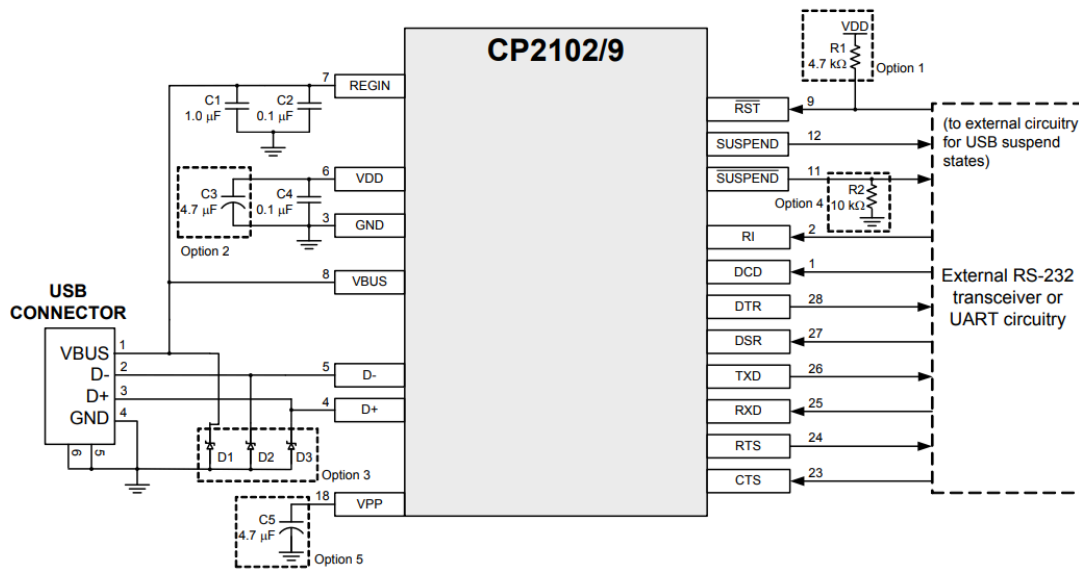
6.1.3: Schematic of ESP32-CAM



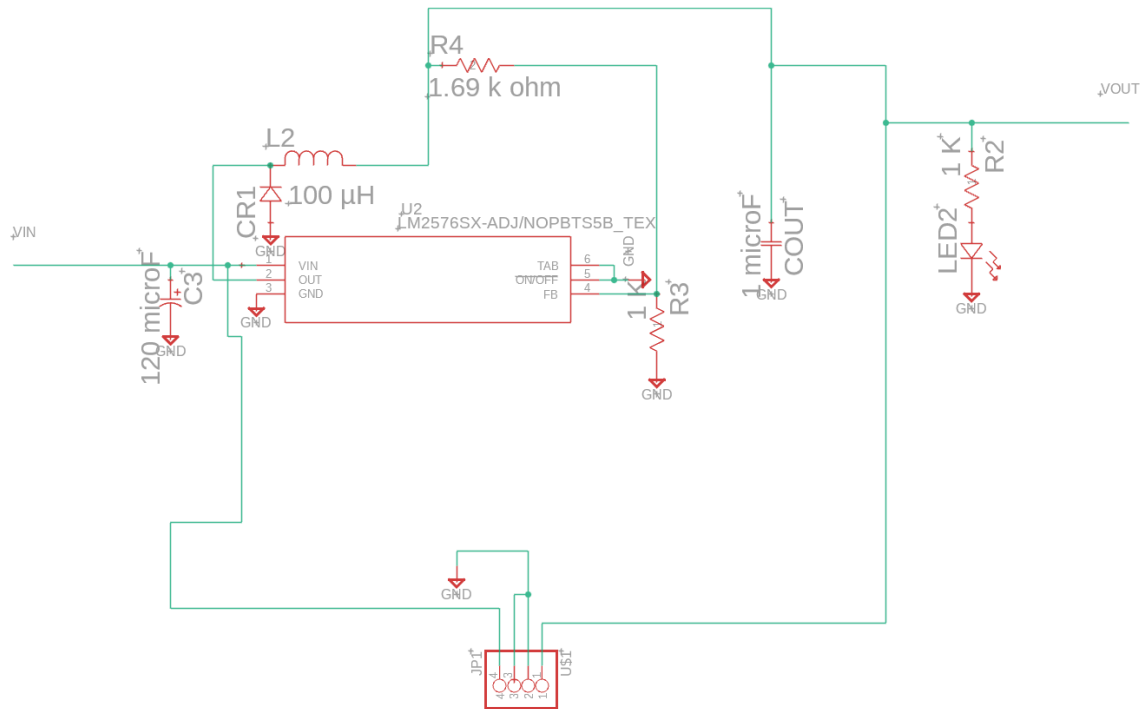
6.2.1: Symbol of MG90s Servo Motor



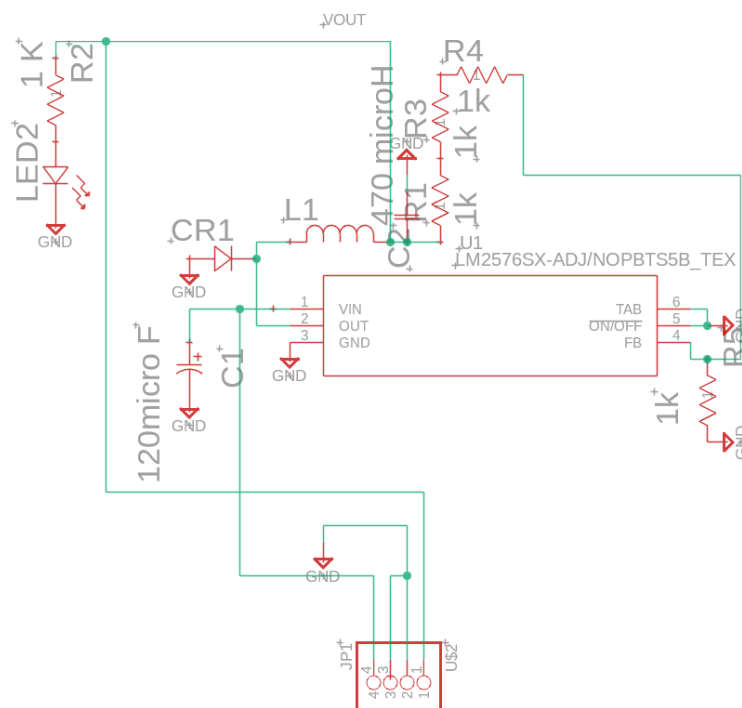
6.5: CP2102 schematic from Silicon Labs



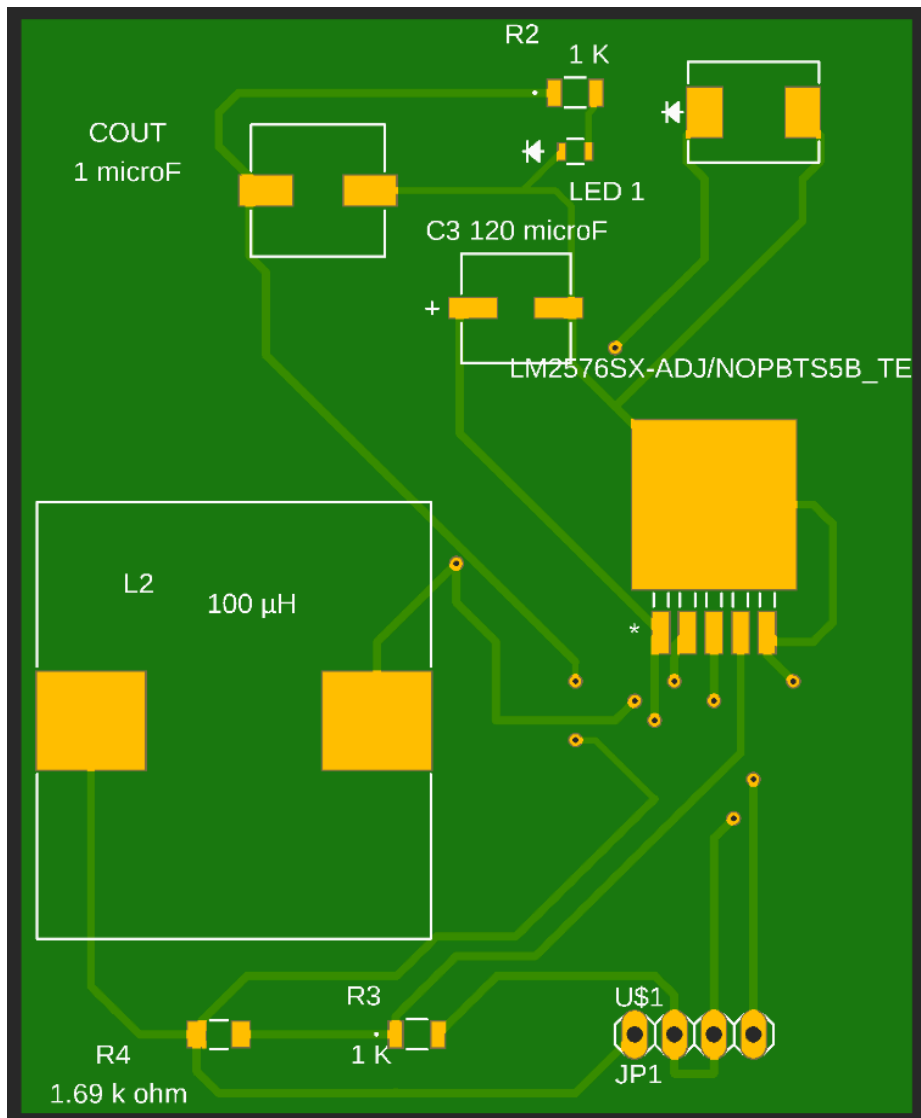
6.4.7 LM2576 3.3v regulator



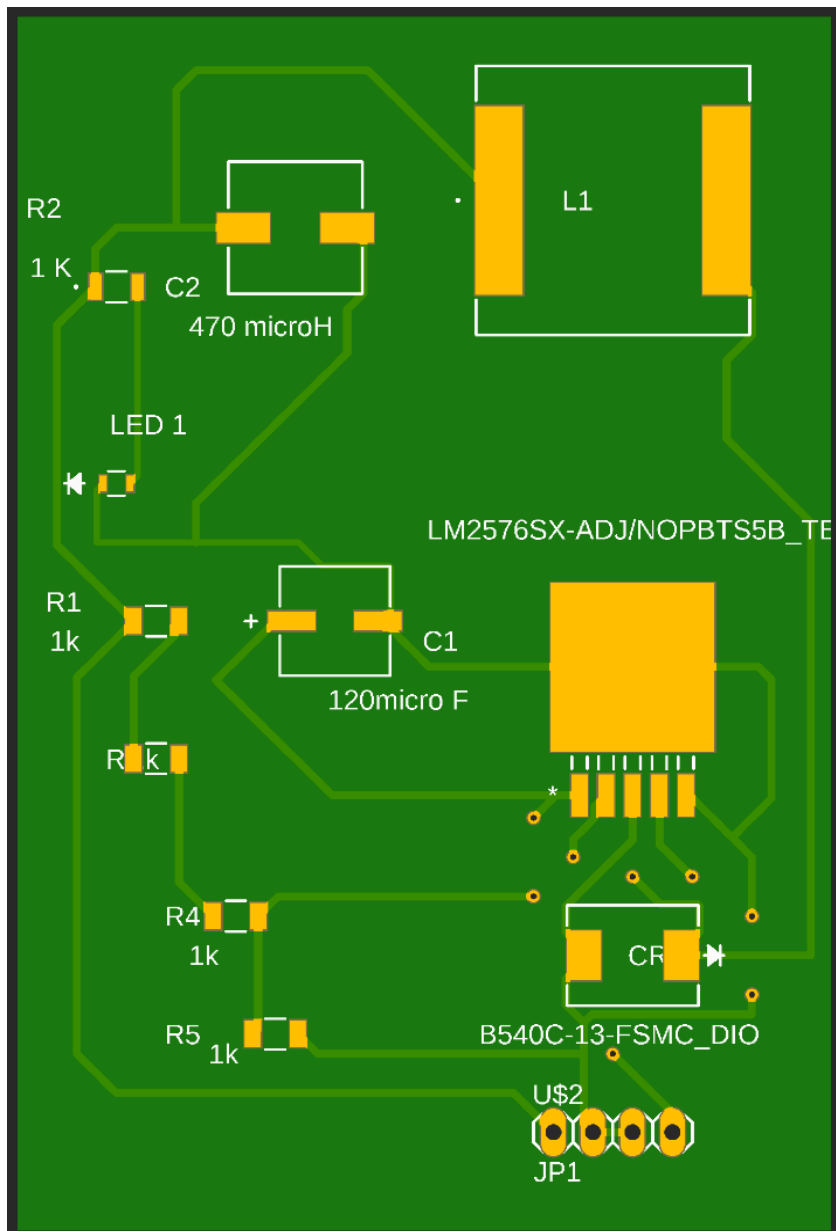
6.4.8 LM2576 5v regulator



LM2576 3.3v regulator PCB



LM2576 5v regulator PCB



APPENDIX D - LLM Prompts

Prompt 1:

What kind of motor would be efficient to use to push the garage door button?

ChatGPT response:

A small linear actuator or a servo motor would be efficient for pushing a garage door button. Here's why:

Servo Motor

Precise control over movement and force.

Can be programmed to push and release the button reliably.

Compact and low power consumption (e.g., a small 9g or 20g servo).

Linear Actuator

Converts rotational motion into a straight push.

Provides consistent force to press the button.

Available in various stroke lengths to match the button's travel distance.

Solenoid Actuator

Directly pushes and retracts when powered.

Simple and fast operation, but limited control over force and speed.

A small servo motor with a pushrod or a short-stroke linear actuator would be the best balance of efficiency, precision, and ease of control with an ESP32 or other microcontroller. Do you have any size or power constraints?

Gemini Response:

When considering a motor to efficiently push a garage door button, several factors come into play, including precision, power, and size. Here's a breakdown of suitable options:

1. Solenoids:

Pros:

Simple and cost-effective.

Provide a linear push motion, ideal for buttons.

Quick response time.

Cons:

Limited stroke length.

May require careful mounting for precise alignment.

They are either on, or off. There is no inbetween.

Ideal for:
Short, direct button presses.

2. Linear Actuators:

Pros:
Precise control over stroke length and speed.
Available in various sizes and power ratings.
Durable and reliable.

Cons:
Generally more expensive than solenoids.
May require more space.

Ideal for:
Situations requiring adjustable or controlled button presses.

3. Stepper Motors:

Pros:
Extremely precise positioning.
Excellent for applications requiring repeatable movements.
Can be used to turn a screw type mechanism to create linear motion.

Cons:
Require more complex control circuitry.
May be overkill for a simple button press.

Ideal for:
Applications where precise control and repeatability are paramount.
Key Considerations:

Force:
Determine the force required to depress the garage door button.

Stroke Length:
Measure the distance the button needs to be pressed.

Control:
Consider how you will control the motor (e.g., relay, microcontroller).

Power Source:
Ensure the motor's voltage and current requirements are compatible with your power source.

Mounting:
How will the motor be mounted, and how much space is available.
In most cases, for a simple garage door button push, a solenoid or a small linear actuator would be the most efficient and practical choice.

I hope this information is helpful.

Copilot response:

To press a garage door button efficiently, you would want to use a small DC motor or a linear actuator. Here's a breakdown:

Small DC Motor: They are compact, lightweight, and energy-efficient. Ideal for creating a mechanism with gears and levers to simulate the pushing motion. Examples include micro gear motors or hobby motors.

Linear Actuator: They convert rotary motion into linear motion, providing precise control over the push action. They're slightly more complex but offer reliable performance for this purpose.

A microcontroller like an Arduino can easily control both of these motors, allowing you to automate the process.

Does that sound like what you need, or would you like more information on either of these options?

Prompt 2:

for a system that requires low power when in use and when it's idle, along with long battery life, what size of battery should I consider using? (AA, AAA, 9V, etc.)

ChatGPT response:

The battery size depends on your system's power requirements (both active and idle), expected runtime, and physical constraints. Here's a breakdown of common options:

Key Considerations:

1. **Power Consumption:** Measure the system's current draw in both active and idle states.
2. **Voltage Requirements:** Ensure the battery voltage matches your system (or use a regulator).
3. **Capacity (mAh or Wh):** Higher capacity means longer runtime.
4. **Self-Discharge Rate:** Lower self-discharge is better for long-term idle systems.

Battery Options:

- **AAA (1.5V, ~1000 mAh)** – Small, lightweight, good for ultra-low-power applications.
- **AA (1.5V, ~2500 mAh)** – Better for moderate-power devices, lasts longer than AAA.

- **9V (~500 mAh)** – Higher voltage, but low capacity, not ideal for long battery life.
- **Lithium Coin Cells (e.g., CR2032, 3V, ~240 mAh)** – Good for very low-power applications.
- **Li-ion (e.g., 18650, 3.7V, ~3000 mAh)** – Rechargeable, high capacity, suitable for ESP32-based projects.
- **LiPo (e.g., 3.7V, 1000-5000 mAh)** – Great for portable devices, but needs careful handling.

Best Choice for Low Power & Long Battery Life:

- If your system is **ultra-low power** (e.g., ESP32 in deep sleep most of the time), **AA lithium batteries** or **Li-ion (18650)** would be ideal.
- If weight and size are a concern, consider **AAA lithium** or **coin cells** for very low power use.
- For a **rechargeable solution**, **Li-ion** or **LiPo** is best.

Do you have an estimate of your system's power consumption? That would help narrow down the best battery choice.

Gemini Response:

When choosing a battery for a system that demands low power consumption and long battery life, both during active use and idle states, several factors come into play. Here's a breakdown of considerations and common battery types:

Key Considerations:

- **Power Consumption:**
 - The system's current draw (measured in milliamps, mA) is crucial. Lower current draw translates to longer battery life.
 - Consider peak current demands versus average current draw.
- **Idle Current:**
 - Even in idle mode, some systems consume power. Minimizing this is essential for long-term battery life.
- **Battery Capacity:**
 - Measured in milliamp-hours (mAh), capacity determines how much current a battery can deliver over time. Higher mAh means longer runtime.
 -
- **Battery Chemistry:**
 - Different chemistries offer varying characteristics:

- **Alkaline (AA, AAA, etc.):** Inexpensive and readily available, suitable for low-drain devices.
-
- **Lithium-ion (Li-ion):** High energy density, rechargeable, ideal for devices with higher power needs.
-
- **Lithium Iron Phosphate (LiFePO₄):** Very long cycle life, safe, and stable, suitable for applications needing longevity.
-
- **NiMH (Nickel-Metal Hydride):** Rechargeable, lower self-discharge than older NiCd batteries.
-
-
-
-
- **Self-Discharge:**
 - All batteries lose charge over time, even when not in use. Low self-discharge batteries are essential for long idle periods.
 -

Battery Types:

- **AA and AAA:**
 - Excellent for low-power devices.
 - Widely available and relatively inexpensive.
 - Rechargeable NiMH AA/AAA batteries (like Panasonic Eneloop) offer a good balance of capacity and low self-discharge.
 -
- **9V:**
 - Often used in devices with specific voltage requirements.
 - Generally lower capacity compared to AA or AAA batteries.
 - May not be the most efficient choice for maximizing battery life in low-power applications.
- **Lithium-ion (18650, etc.):**
 - Offer high energy density, making them suitable for devices with varying power needs.
 - Rechargeable, providing long-term cost savings.
 - Require proper charging circuitry and safety precautions.
- **LiFePO₄:**
 - These are growing in popularity, and provide a very long life. They are a good choice when longevity is a high priority.

Recommendations:

- For very low-power applications with long idle periods, high-quality alkaline or low self-discharge NiMH (like Eneloop) AA or AAA batteries are often the best choice.
-
- If the device requires a higher amount of power, or is intended to be used very frequently, then a lithium ion battery or a LiFePO4 battery would be a better choice.
- Always consider the device's voltage requirements and current draw when selecting a battery.

To make the best choice, it is very important to know the power consumption requirements of your system.

Copilot Response:

For a low-power system that aims for long battery life, you might want to consider using AA or AAA batteries. Both are commonly used, easily available, and can provide adequate power for many low-power devices. Here's a quick comparison to help you decide:

AA Batteries:

- **Capacity:** Generally higher (around 2000-3000 mAh for standard alkaline)
- **Size:** Larger than AAA
- **Lifespan:** Longer due to higher capacity
- **Use Cases:** Ideal for devices with slightly higher power consumption (e.g., remote controls, portable lights)

AAA Batteries:

- **Capacity:** Lower (around 1000-1200 mAh for standard alkaline)
- **Size:** Smaller, more compact
- **Lifespan:** Shorter compared to AA due to lower capacity
- **Use Cases:** Suitable for smaller devices with lower power requirements (e.g., small electronic gadgets, clocks)

9V Batteries:

- **Capacity:** Typically much lower in terms of milliampere-hours compared to AA or AAA (around 400-600 mAh)
- **Use Cases:** Best suited for devices specifically designed for 9V (e.g., smoke detectors, some walkie-talkies)

Ultimately, the choice depends on your specific power requirements and the physical space available in your device. If you need higher capacity and longer lifespan, AA might be the better option. If space is limited, AAA could be more appropriate. For niche applications requiring 9V, then of course, the 9V battery would be the one to use.

Let me know if there's more detail you'd like!

Case Study 3:

I'm using ESP32-CAM to do an object detection project. I need the camera to recognize objects in the way when the garage is open. What are my choices?"

For your ESP32-CAM object detection project—where the camera checks for **obstacles when the garage is open**—you have several options depending on **accuracy**, **latency**, **power usage**, and **memory constraints**. Here's a breakdown of your main choices:

♦ 1. **Use TensorFlow Lite for Microcontrollers (TFLite Micro)**

✅ Pros:

- Designed to run on resource-constrained devices like ESP32-CAM.
- Lightweight models like **Person Detection** can fit in memory (~250KB RAM used).

! Limitations:

- Only supports very **small models** (usually grayscale, 96x96 input).
- Hard to recognize multiple object types unless you customize or retrain the model.

Use When:

- You're only detecting **presence** of an object (e.g., person, blob).

Example:

- [TensorFlow Lite Person Detection demo](https://github.com/tensorflow/tflite-micro/tree/main/tensorflow/lite/micro/examples/person_detection)

♦ 2. **Use ESP-WHO (Espressif's Face/Object Detection Framework)**

✅ Pros:

- Developed by Espressif, optimized for ESP32-CAM.
- Supports **face detection, face recognition**, and **simple object detection**.
- Well-documented examples.

! Limitations:

- More about **face/shape recognition** than object classification (no COCO-style labels).
- Detection only, not classification.

Use When:

- You want basic obstacle detection by shape/movement without needing object types.

Example:

- `camera\_web\_server` or `face\_detection` example in [ESP-WHO GitHub](https://github.com/espressif/esp-who)

♦ 3. **Use External Processing (ESP32 sends frames to another device)**

✅ Pros:

- Run **full YOLO, MobileNet, or SSD** on a **Raspberry Pi**, **NVIDIA Jetson**, or even a **cloud API** (like Google Vision, AWS Rekognition).
- Much higher accuracy and flexibility.

! Limitations:

- Requires **Wi-Fi bandwidth** and latency.
- Needs a more complex setup.

Use When:

- You want to recognize multiple object types (e.g., person, bike, box, pet).
- ESP32-CAM acts as a **streaming camera**, not the main processor.

♦ 4. **Motion Detection with Basic CV (OpenCV style)**

✅ Pros:

- Lightweight—detect moving objects by comparing frames.

- Doesn't need ML model.

! Limitations:

- Only detects **movement**, not object type.

Use When:

- You only care if **something is in the way**, not what it is.

So Which One Should You Use?

Goal	Best Choice
-----	-----
Detect presence of *any* object	ESP-WHO or TFLite Micro
Detect if a *person* is there	TFLite Micro with Person Detection
Detect *specific objects*	Offload frames to external device
Detect *motion only*	Frame diff (basic CV)

Would you like help:

- Picking a pre-trained model?
- Setting up TFLite Micro?
- Sending ESP32-CAM images to a server for external object detection?

Let me know how advanced you want this to be.

Logo image generated using Canva's AI tools.

Declaration: We hereby declare that we have not copied more than 7 pages from the Large Language Model (LLM). We have utilized LLM for drafting, outlining, researching, and proofreading purposes.

