SYSC 3010
Computer Systems Development Project
Print Guard
Detailed Design Document

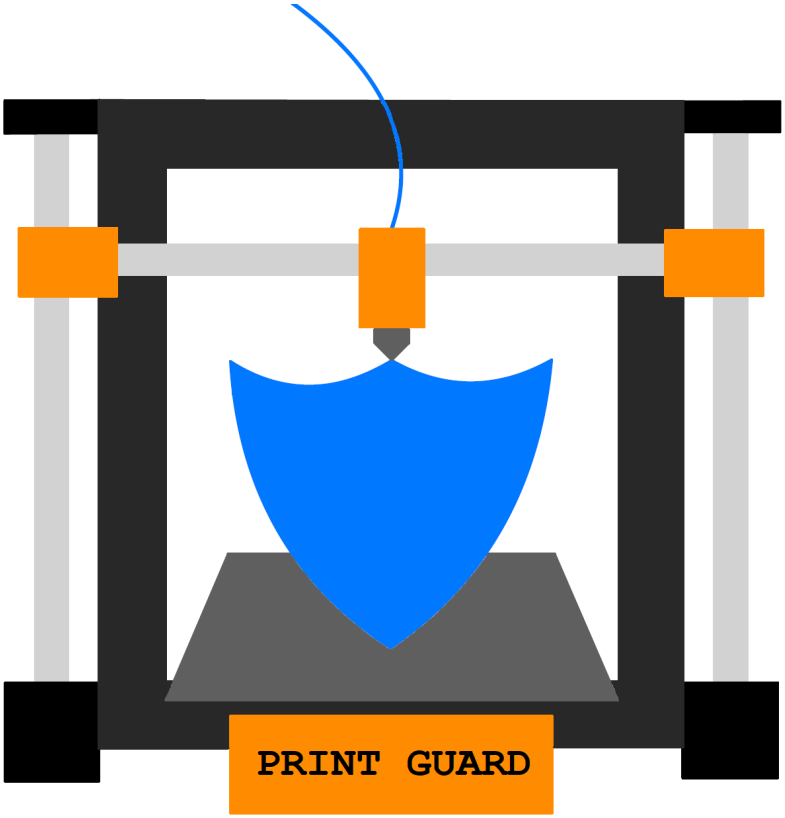

Figure 1: Print Guard Logo.

Group L2-G9

Joseph Vretenar - 101234613
Basma Aboushaer - 101186291
Cholen Premjeyanth - 101184247
Raunak Singh Soi - 101219648
Samual Mauricla- 101233500
TA: Ben Earle

March 13th 2024

Table of Contents

1 Problem Statement

Over the last three decades, there has been substantial growth in the additive manufacturing industry, spanning across both commercial and public sectors [1]. One major growth has been the accessibility of fused deposition modeling (FDM) 3D printers. FDM 3D printers construct products layer by layer by extruding melted thermoplastic material through a nozzle, creating a three-dimensional object based on a digital design. Occasionally, these printers may exhibit unreliability, and for larger designs, they often require extended hours of operation, placing the user in the dilemma of either closely monitoring the 3D printer to prevent potential failures or entrusting it to operate unattended with the hope of success. If a print fails it can cause hours of wasted time as well as lots of filament.

1.1 Functional Requirements

To properly monitor 3D printers, the Print Guard system depends on six functional requirements.

Recording Sensor Data: The system will be able to monitor multiple sensors that measure the external temperature, the enclosure temperature, the printer nozzle temperature, and the printer bed temperature. This will help to determine if a failure is due to one of these factors being off.

Retrieving Camera Footage and Photos: The system will take photos of the print in periodic intervals which will allow the user to go back after and review them for defects. If possible the AI crash detection will also record and store the whole print when a crash is detected to review the full video to determine the exact time of failure.

Controlling the 3D printer: The system should allow for control of the 3D printer, to allow for remote control of the current print. This will help to stop wasted resources as if a failure is viewed the print can be stopped. After many prints the AI crash detection could also be good enough to stop the print on its own based on the footage.

Notifying User of Progress: The system will notify the user about print finishing, possible faults from the sensor data, and if possible the AI crash detection should notify the user if a crash is detected.

Live Camera Feed: The system should be able to display a live view of the print in progress that is protected behind username and password on the website.

Temperature Control: The system should be able to adjust the internal temperature with a fan.

2 Design Overview

2.1 System Overview Diagram

Print Guard is a real time 3D printer monitoring system. The deployment diagram for the system can be found below in Figure 2. The system is made up of multiple different nodes consisting of different types of computers and hardware devices. These nodes will be communicating using different types of protocols based on what the node is doing, the explanation of the communication protocols along with the messages can be found below in section 2.2 and 2.2.1.

Following Figure 2 below, the system is made up of six different computers consisting of two Raspberry Pi 4s, two Arduino Nanos, one Prusa Mk4 3D printer, and one desktop computer. The Arduinos will communicate to one RPi to send temperature and humidity data as well as receive data to control a fan. The two RPis will communicate to each other to send images. The 3D printer will communicate to one RPi to send sensor data as well as receive commands. The desktop will be used to receive a camera feed from one of the RPis to use it for AI crash detection.

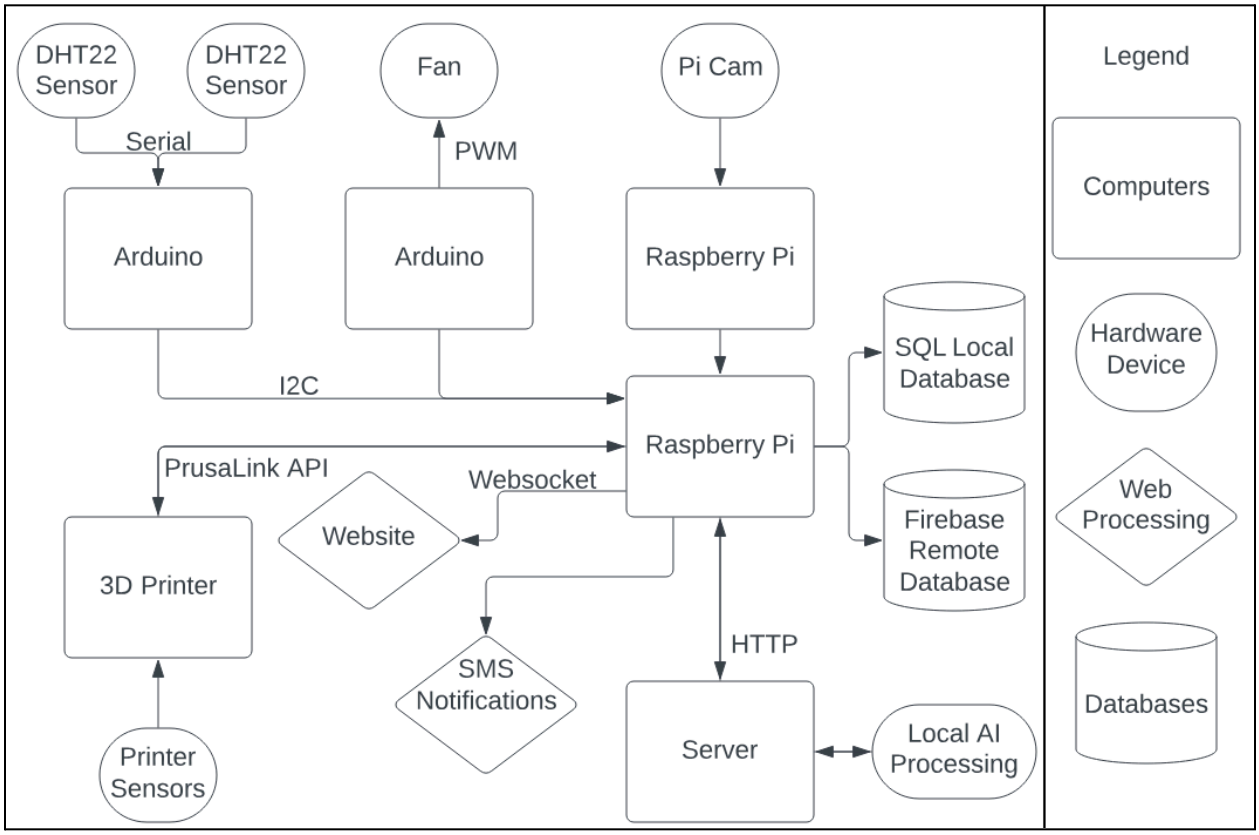

Figure 2: Simple Deployment Diagram

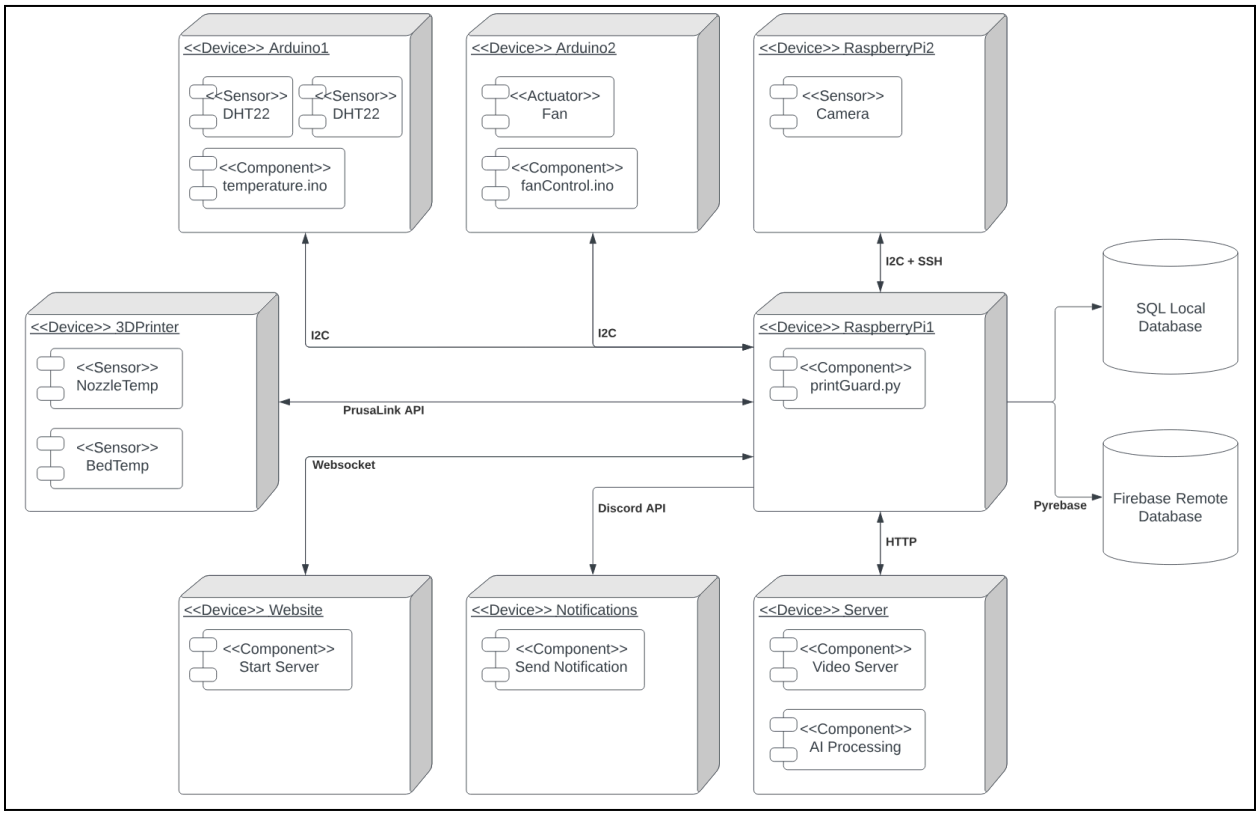

Figure 3: UML Deployment Diagram

2.2 Communication Protocols

The Print Guard system has two different communication layers; the physical layer which consists of the hardware components communicating to computers, and the software layer where computers will communicate remotely to other computers or to the cloud.

2.2.1 Physical Layer Communication

The Print Guard system uses three different communication protocols at the hardware level, the two types of connections are from Arduinos to hardware devices, and from Arduinos to RPis. The Arduinos will use the serial communication protocol to read the DHT22 sensor data, while also using a PWM signal to control the speed of the fan. Both of the Arduinos are connected to one RPi through the use of the I^2C protocol.

2.2.2 Software Layer Communication

The software level of the Print Guard system will use multiple different communication protocols to send and receive data. The main protocols will be websockets to host a website, secure shell (SSH) to communicate between RPis, the Firebase API to connect to the cloud

database for data storage, the PrusaLink API to communicate with the printer, and HTTP to communicate with the desktop for AI crash detection.

2.2.3 Communication Protocol Table

2.2.3.1 Hardware Messages

Hardware messages consist of the messages to get data from sensors and control devices through both layers of communication. These will allow the system to get sensor data, control the fan, and control the camera.

Sender	Receiver	Message	Parameters	Return
Arduino 1	DHT22 Internal	getInternal	`sensorName`	`temp, humidity`
Arduino 1	DHT22 External	getExternal	`sensorName`	`temp, humidity`
Arduino 2	Fan	changeSpeed	`PWM (0 - 255)`	`none`
RPi 1	Arduino 1	getTemps	`none`	`temp1, humidity1, temp2, humidity2`
RPi 1	Arduino 2	changeSpeed	`bool: 1(up), 0(down)`	`none`
RPi 2	Pi Camera	takePhoto	`imageName, resolution`	`none`
RPi 2	RPi 1	sendPhoto	`imageName`	`none`
RPi 1	3D Printer	getTemps	`printer`	`nozzleTemp, bedTemp`
RPi 1	3D Printer	shutdown	`printer`	`none`
RPi 1	3D Printer	getRemaining	`printer`	`printTime, printPercentage`

Table 1: Communication Protocol Table for Hardware Devices

2.2.3.2 Database Storage

The database storage will consist of sending the information to both sqlite3 and firebase for local and remote storage. This will allow for a local backup to be pushed to the firebase if the wifi goes down.

Sender	Receiver	Message	Parameters	Return
RPi 1	SQLite	storeDHT	`temp1, humidity1, temp2, humidity2, datetime`	`none`
RPi 1	SQLite	storePrinter	`nozzleTemp, bedTemp, imageName, datetime`	`none`
RPi 1	Firebase	storeDHT	`temp1, humidity1, temp2, humidity2, datetime`	log if connection cannot be established
RPi 1	Firebase	storePrinter	`nozzleTemp, bedTemp, imageName, datetime`	log if connection cannot be established

Table 2: Communication Protocol Table for Database Storage

2.2.3.3 Website Control and Notifications

The website will be constantly updated with new data as it is obtained. The website will also allow for remote control of the 3D printer so the messages are sent both ways. The notifications as of right now will be sent through Discord using a bot and Discords API.

Sender	Receiver	Message	Parameters	Return
RPi 1	Webserver	updateTable1	`temp1, humidity1, temp2, humidity2, datetime`	`none`
RPi 1	Webserver	updateTable2	`nozzleTemp, bedTemp, imageName, datetime`	`none`
RPi 1	Webserver	updateGraph	`none`	`none`
RPi 1	Webserver	updatePrint	`printTime, printPercentage`	`none`

websocket	RPi 1	getView	`password`	`cameraView, or error + log if authFailed`
websocket	RPi 1	shutdown	`bool: 1(yes), 2(no)`	`Shutdown Message`
RPi 1	Discord	highTemp	`none`	`none`
RPi 1	Discord	lowTemp	`none`	`none`
RPi 1	Discord	crashDetected	`none`	`none`
RPi 1	Discord	unauthorized Access	`none`	`none`

Table 3: Communication Protocol Table for Website and Notifications

2.3 Message Sequence Diagrams

To meet the six functional requirements of the Print Guard system described in Section 1.1, the following use cases will be described with message sequence diagrams below.

2.3.1 Message Sequence Diagram 1: Recording Sensor Data

In this system, an Arduino device is responsible for periodically collecting sensor data, typically at 15-second intervals. The collected data is then transmitted to a Raspberry Pi 1, which serves as the central processing hub. Upon receiving the sensor data from the Arduino, RPi 1 performs several tasks. Firstly, RPi 1 stores the received sensor data into a local SQLite database for future reference and data integrity. This database serves as a primary repository for historical sensor readings. Additionally, RPi 1 synchronizes the received data with a remote Firebase database, ensuring that the sensor readings are accessible and up-to-date across multiple platforms and devices. Furthermore, RPi 1 updates a Graphical User Interface (GUI) to visually represent the latest sensor readings in real-time. Users can monitor these graphical representations to gain insights into the environmental conditions captured by the sensors. At the same time, RPi 1 monitors the temperature readings to ensure they fall within predefined acceptable ranges. If the temperature reading deviates from the specified thresholds, indicating potential anomalies such as excessively low or high temperatures, RPi 1 triggers a notification mechanism. This mechanism alerts the user about the abnormal temperature conditions, allowing them to take appropriate actions, such as adjusting environmental controls or investigating potential issues with the sensor or the monitored environment. Overall, this system enables the continuous recording of sensor data, facilitates real-time monitoring through a graphical interface, and provides timely notifications to users in the event of temperature anomalies, contributing to efficient environmental monitoring and management.

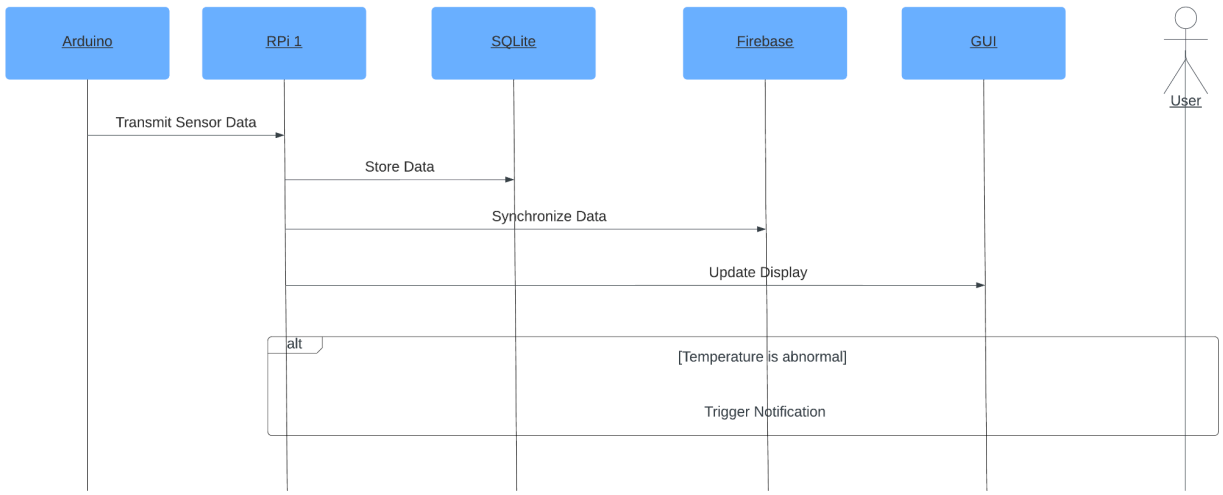

Figure 4: Message Sequence Diagram for Recording Sensor Data

2.3.2 Message Sequence Diagram 2: Capturing Photos

The RPI camera will capture images periodically to assess the item being printed. If any defects are noted with the item, the user can analyze the photos and capture where the defect occurred. Firstly, the user prints an item, which automatically activates RPi 1, which then sends a message to RPi 2 to capture photos of the item being printed. Once RPi 2 receives the message to capture a photo, it captures a photo and sends the picture back to RPi 1. RPi 1 now has the image, and adds the image to SQLite, Firebase, and finally updates the GUI, which firmly should be ready for the next photo that is required to be taken.

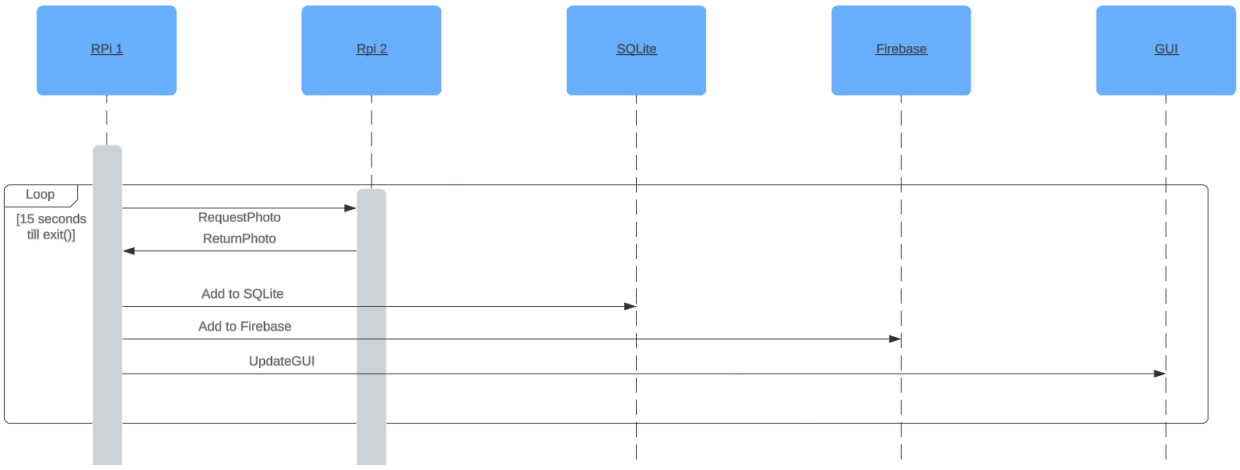

Figure 5: Message Sequence Diagram for Capturing Photos

2.3.3 Message Sequence Diagram 3: 3D Printer Control

To save resources, the Print Guard system allows for control of the 3D printer through the GUI. This will allow the user to send messages to the printer remotely to make changes. The printer will only be able to be controlled by admins on the web server. This will help to prevent anyone from controlling the printer. Depending on the command being sent, the system will ask for confirmation. In the sequence diagram shown below the user is trying to shut down the printer. The user will send the command through the website, which will ask for confirmation to execute the command. On confirmation, the GUI will notify RPi 1 that the command has to be sent to the printer. The RPi will then send the command to the printer.

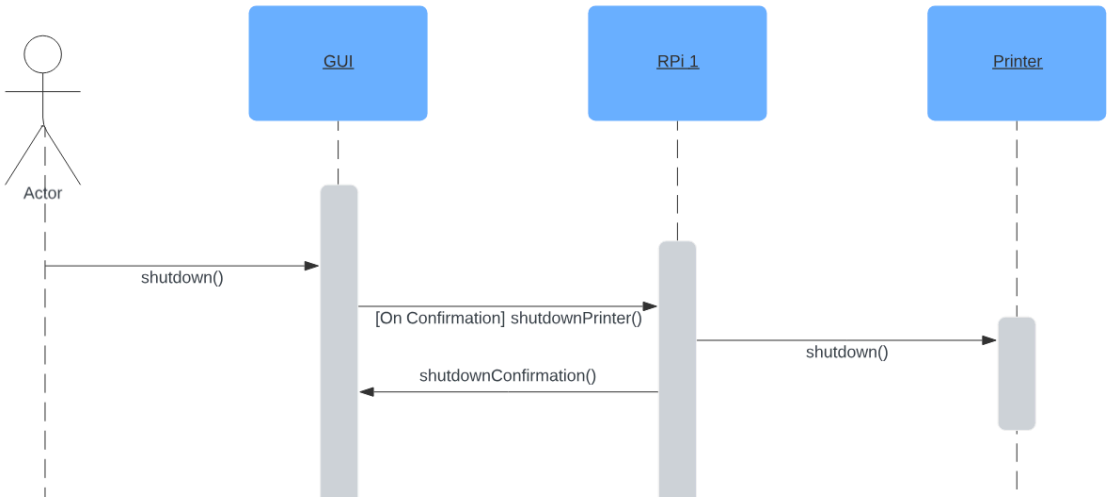

Figure 6: Message Sequence Diagram for 3D Printer COntrol

2.3.4 Message Sequence Diagram 4: Notifying Users

In this system, the RPi 1 is mostly responsible for handling any errors that arise and informing the user of them via Discord. To begin, if and when an error does occur, RPi 1 sends a message to RPi 2 to take a photo. The reason for the photo is to analyze the issue that occurred. RPi 2 will then take a photo as requested by RPi 1. Once RPi 2 takes the photo, the user will receive an error message via Discord letting the user know what the error is and what caused it. RPi 1 will then get the data from the Sensor data and look to figure out what the cause of the error was. RPi 1 will load the recorded error issue into Sqlite. This will allow data to be structured and organized for storage in the event of previous errors. Additionally, error data is also transferred to the Firebase database. This will help with accessing data virtually while also viewing data at real time.

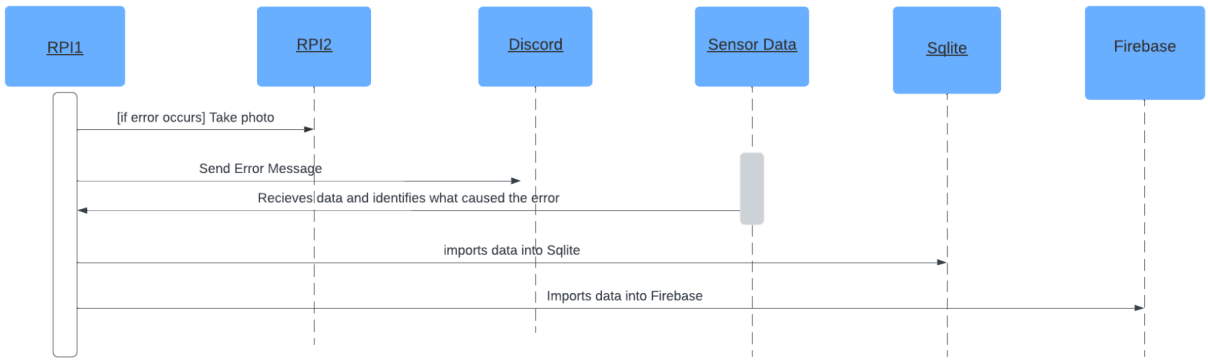

Figure 7: Message Sequence Diagram for Notifying Users

2.3.5 Message Sequence Diagram 5: Live Camera Feed

The user will have the ability to view the item being printed from the website at any given moment. In order to do so, the user will have to access a live stream protected behind a password. This password will allow for improved security for the live feed so only authorized users can view it. The user will request to view the stream through the GUI, this will prompt them to enter a password. Once they enter the password, if it is correct, they will be allowed to view the stream. The GUI will tell RPi 1 that they can view the stream, which will tell RPi 2 to start the view. Then the GUI will be updated to show the live view.

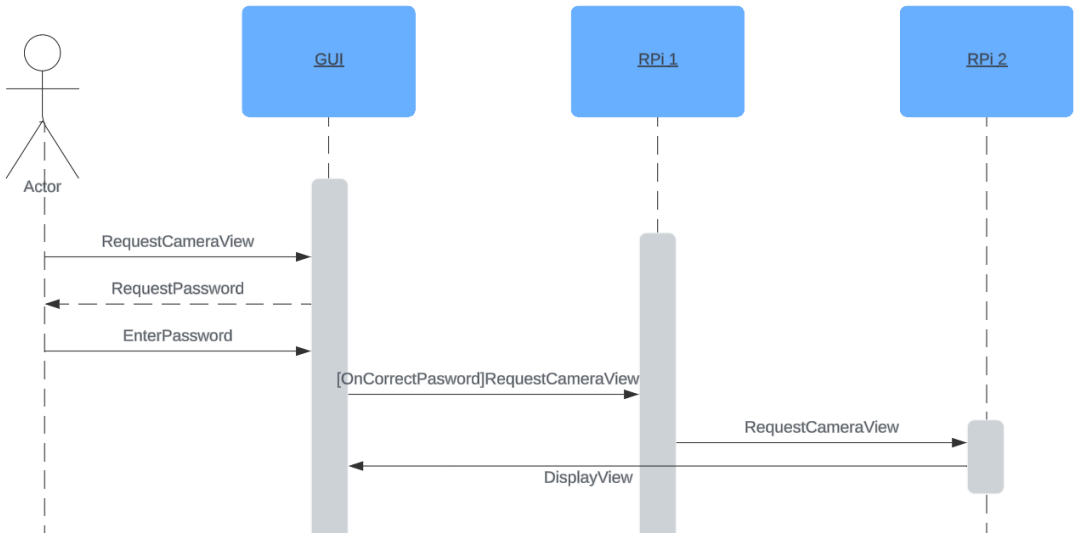

Figure 8: Message Sequence Diagram for Live Camera Feed

2.3.6 Message Sequence Diagram 6: Temperature Control

The temperature control of the printer enclosure is done through the use of a PWM fan. This allows for variable speed on a fan to control the temperature easily. The fan will be controlled by an Arduino from the data sent by Raspberry Pi 1. The RPi determines if the temperature is too high or too low from the data recorded on the DHT22 temperature sensors. The RPi will tell the Arduino that the temperature is too high or too low and the Arduino will adjust the fan speed accordingly. The Arduino will then return the fan's RPM to be put in the database.

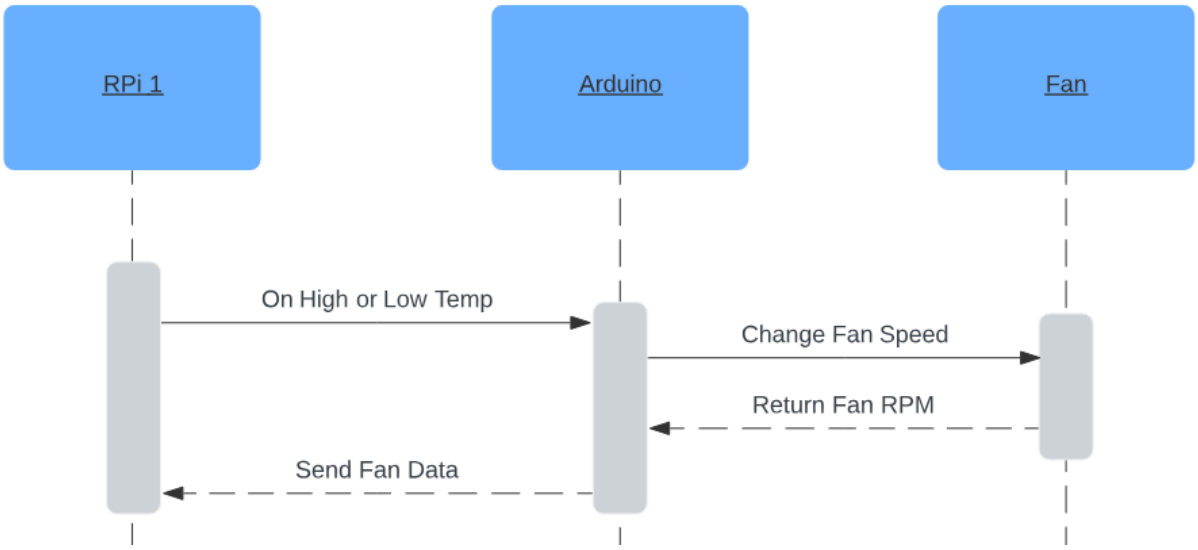

Figure 9: Message Sequence Diagram for Temperature Control

2.4 Database Table Design/Schema

The Print Guard system will be using two separate databases, a local SQLite database and a remote Firebase database. This will allow for the wifi connection to be dropped and all the data recorded during that time is still stored locally and can be pushed to the Firebase when a connection is re-established. The databases will be broken down into three different tables, the first table will hold the internal DHT22 sensor data, the external DHT22 sensor data, the current fan speed, and the date and time. The second table will hold the 3D printer's nozzle and bed temperature, the percentage through the print, the captured images of the print in progress, as well as the date and time. The final table will hold any logs that have been created, this would be something like a high temp warning or a crash detection. This table will hold the date and time of the log, the type of error (high nozzle temp, low nozzle temp, possible crash, etc…), the recorded value that the system thought was an error, and a captured image at the time of the log. The design of the database can be found below in figure 9.

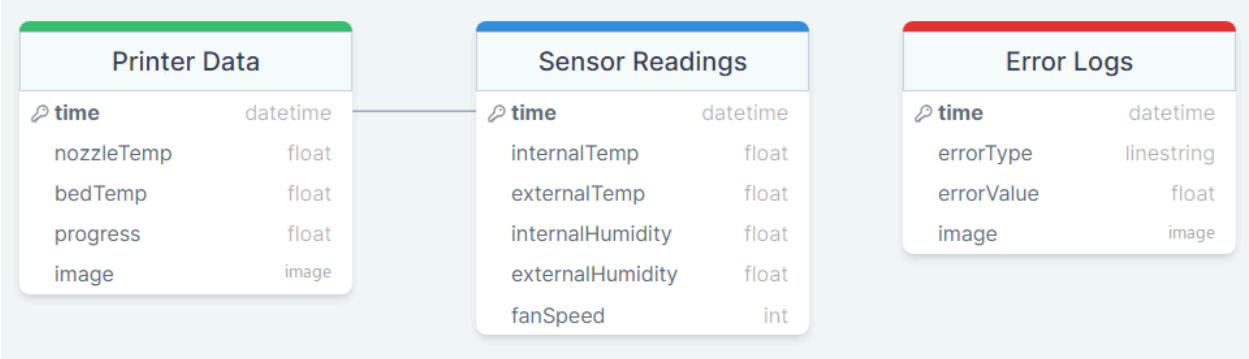

Figure 10: Database Design/Schema

3 Software Design

The UML diagram shown in Figure 11 visualizes a Print Guard system's modular design, illustrating the relationships and data flow between various components responsible for overseeing and managing printing operations.

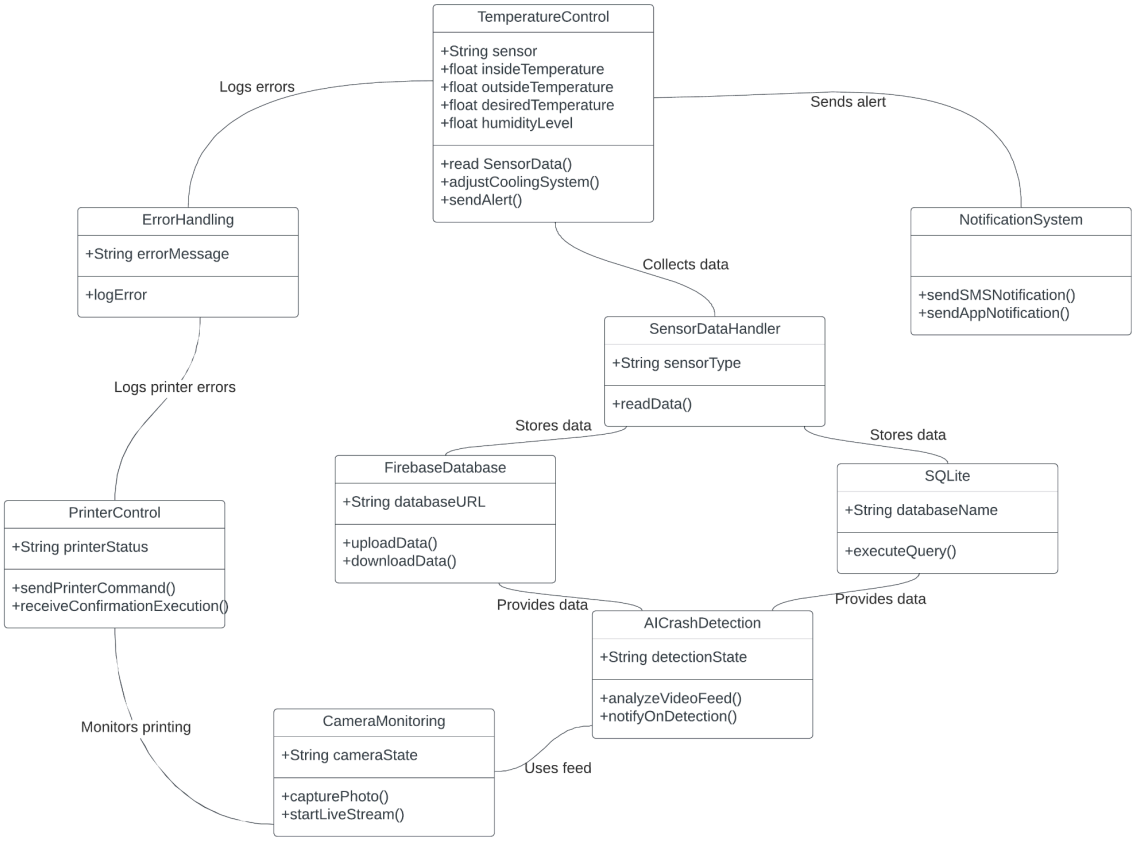

Figure 11: UML Class Diagram of Software Design

3.1 Software Design for Node 1: Temperature Control

The temperature control sensor node focuses on optimizing 3D printing operations through precise temperature and humidity monitoring. Python is selected as the primary programming language due to its versatility and robust libraries. The software components consist of data acquisition scripts responsible for continuously reading temperature and humidity data from DHT22 sensors placed inside and outside the 3D printer's enclosure. These scripts ensure seamless acquisition and processing of sensor data, allowing for real-time monitoring of the printing environment. The core of the node lies in the temperature control algorithm, which dynamically adjusts PWM signals based on real-time temperature readings to regulate cooling mechanisms effectively. This algorithm ensures stable print quality, prevents filament warping, and enhances energy efficiency by activating cooling mechanisms only when necessary. Additionally, the node incorporates a notification system triggered by temperature deviations. Python scripts handle notifications, sending alerts via email, SMS, or mobile app notifications to keep users informed about changes in temperature conditions.

Error handling and logging mechanisms are integrated into the software design to manage exceptions and record system status for analysis and troubleshooting. The temperature control sensor node seamlessly integrates with the broader system architecture, facilitating communication with other components such as the main control unit and user interface. Clear communication protocols and interfaces ensure smooth interaction between system elements, enhancing reliability and scalability. Through comprehensive testing and validation procedures, including unit tests and system tests, the software design ensures the robustness and performance of the temperature control sensor node, contributing to efficient and safe 3D printing operations.

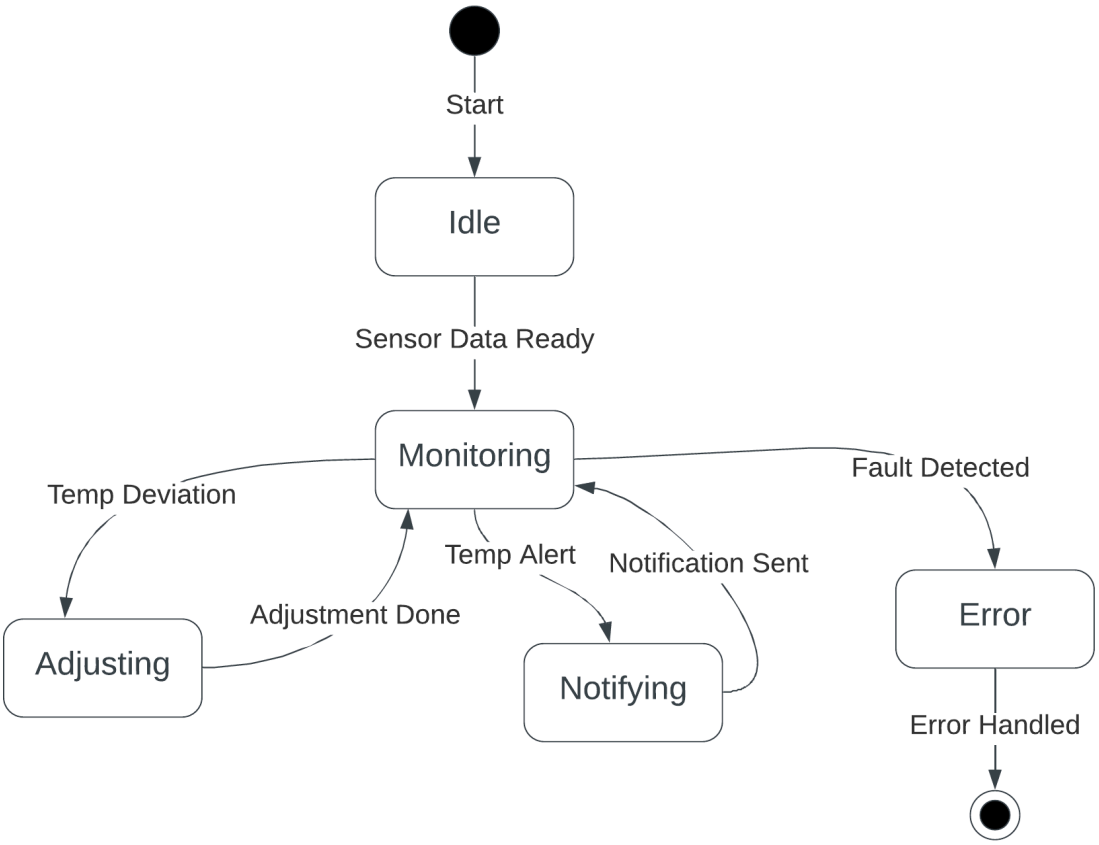

Figure 12: State Machine of Node 1 Temperature Control

3.2 Software Design for Node 2: Databases

The databases such as SQLite and Firebase store the data and the system will constantly be waiting for any new data. This is why the system is always waiting. The data that is being stored in the system consists of errors that have been found. As well as data such as fan speed, sensor data and printer data. As seen below, the system will always begin in idle and will wait until data is received. Data is received every 15 seconds as it gets recorded by all the sensors. If there is no internet connection, the system will not be able to upload the data to Firebase. The system will make a note of the data point that was not uploaded, and will upload all the data from that point when it re-establishes connection.

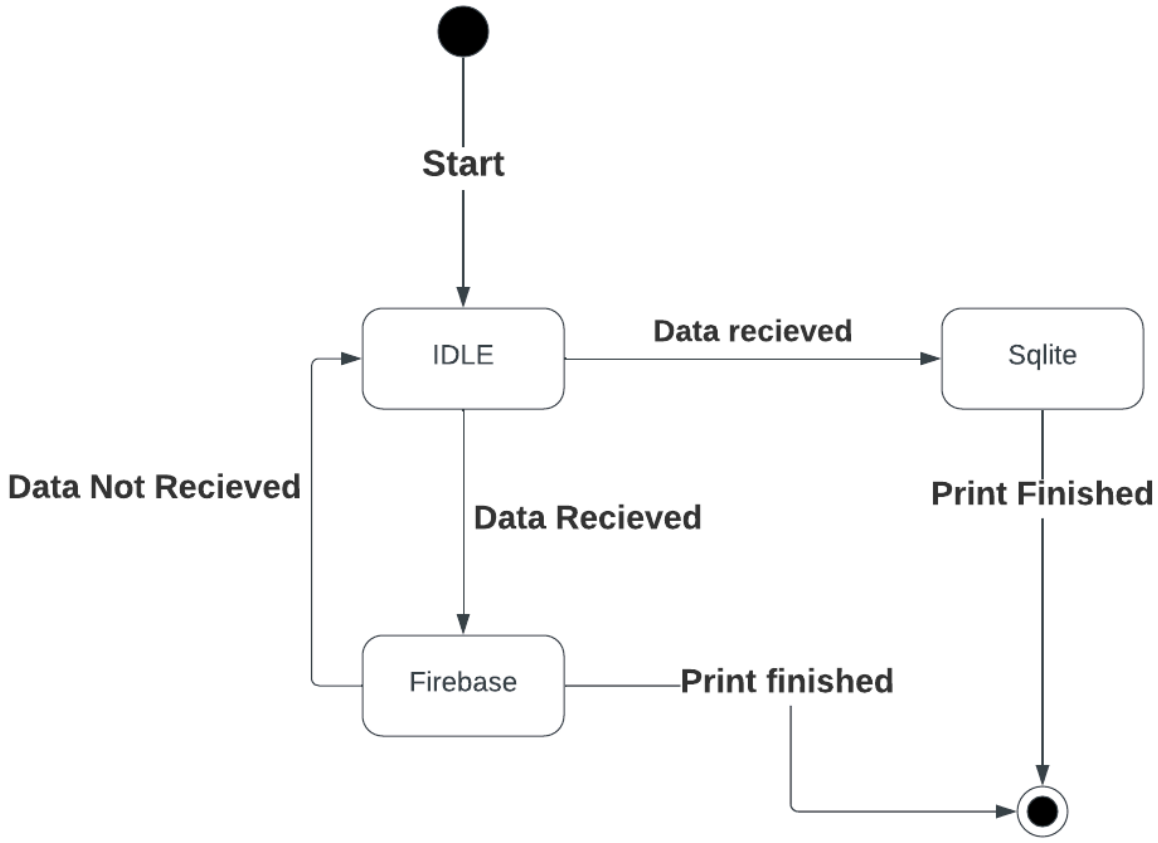

Figure 13: State Machine for Software Design of Databases Node

3.3 Software Design for Node 3: Camera Monitoring

The camera monitoring node will allow the user to have a sense of security and access to the item that is being printed. The figure below shows how the camera monitoring subsystem will be implemented in state machine form. The system has 2 states. State 2 is where the printer is printing an item, meaning that the system is taking photos periodically. State 2 is where a live stream feed of the item has been requested. RPi 2 (the camera that is capturing photos and the livestream), is constantly waiting for feedback from RPi 1, which will inform RPi 2 on whether to capture photos or livestream the item. If RPi 2 is live streaming, RPi 1, requests a web socket, where the user will have to enter security details to access the livestream. If the request is wrong, the system goes back to State 1, where photos are periodically taken, State 2 and enters an idle state waiting for a message.

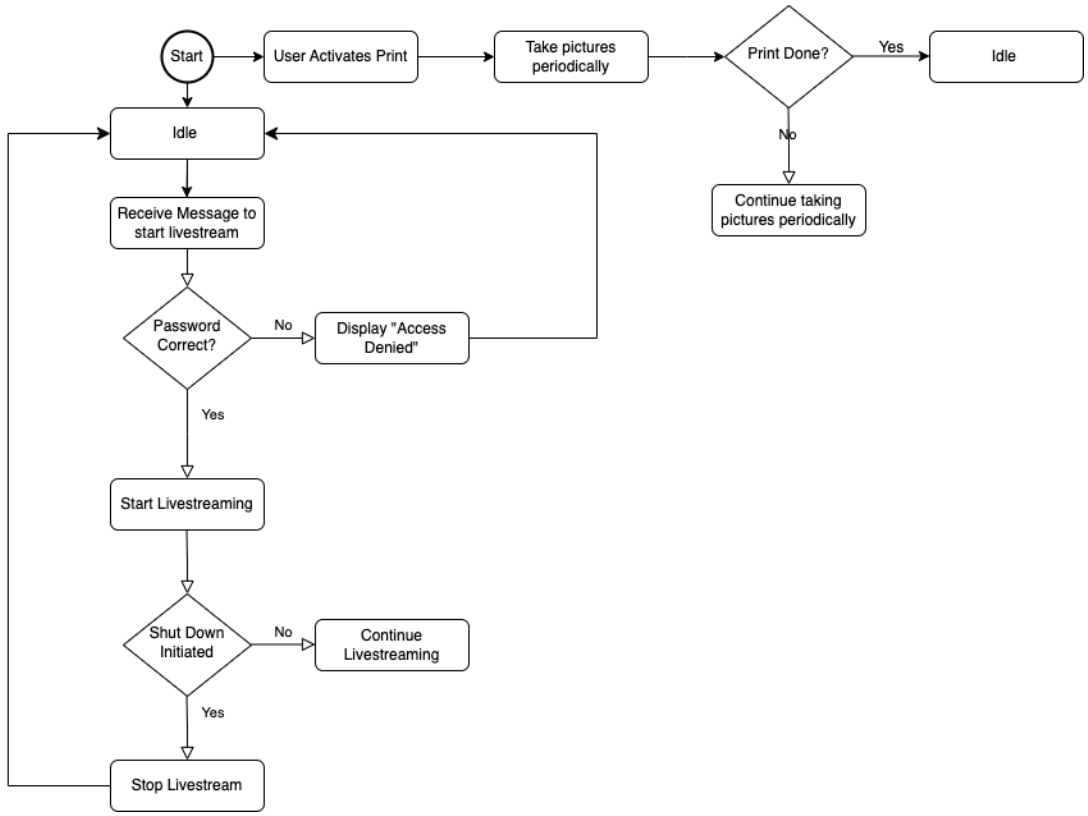

Figure 14: State Machine for Software Design of Camera Monitoring

3.4 Software Design for Node 4: Printer Control

The printer control node for the Print Guard system will allow the user to remotely control the 3D printer. This is key if an error is detected and the user wants to stop the print. The system will be waiting idle for a command to be requested. This will be done through the GUI that is always waiting for an input. When an input is received it will look up if the command needs confirmation and ask for it. When it receives confirmation, the GUI will relay the command to the RPi, which will then relay it to the 3D printer.

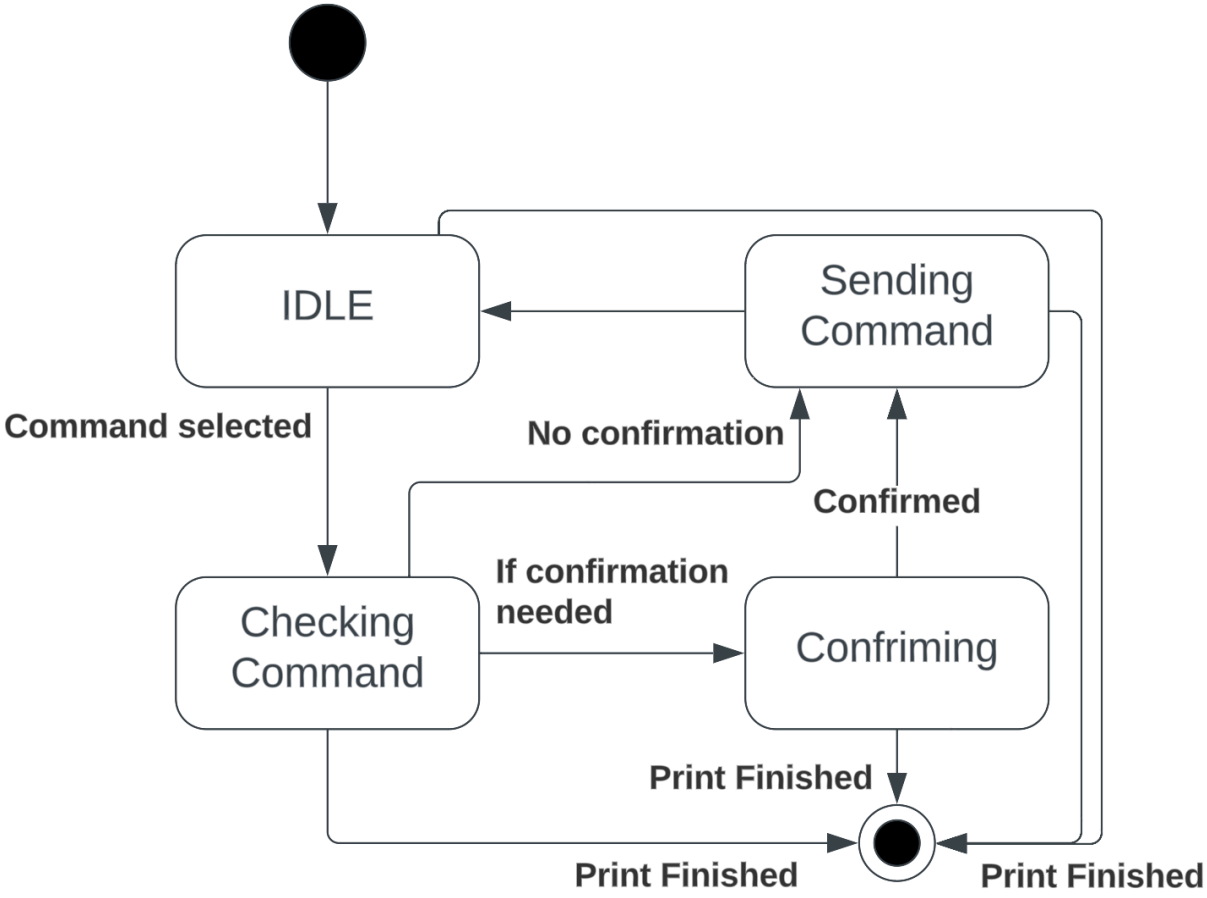

Figure 15: State Machine for Software Design of 3D Printer Control

3.5 Software Design for Node 5: AI Crash Detection

The AI crash detection node will be a great addition to the Print Guard system. This will allow the user peace of mind and allow them to not have to be constantly checking the website for a camera view. The AI system will be constantly looking at the video feed of the RPi camera. This will allow the AI to detect possible faults in the print based on visuals. This system may need to be tuned to get consistent results with minimal false positives. The system will have three different states of crash detection. The first state will be no crash detected, this will be the default state and would get increased to the next state if something starts looking wrong. The next state will be possible crash detected, this state shouldn't give a notification as it is possible to get more false positives in this state. The final state will be crash detected which should notify the user to check the video feed. Further down the line with refinements to the AI system, this state could also shutdown the printer to instantly avoid wasting resources over a failed print.

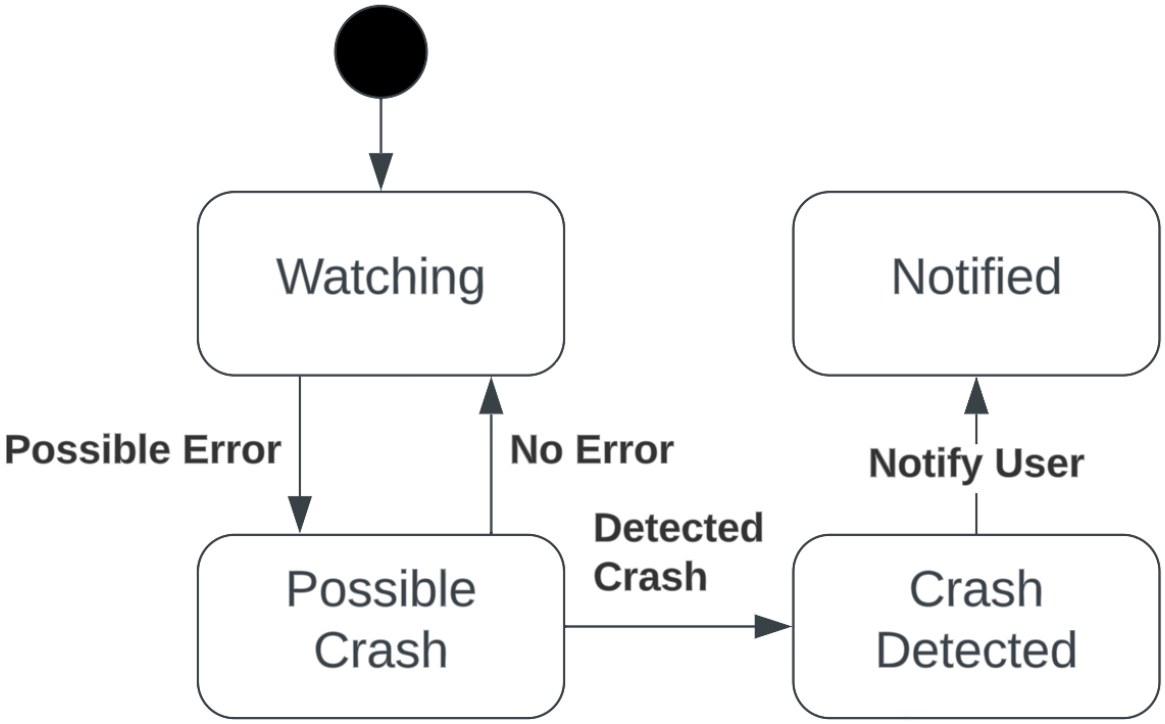

Figure 16: State Machine for Software Design of Ai Crash Detection

4 Hardware Design

4.1 Hardware Design for Component 1: Sensor Monitoring

The temperature monitoring for the Print Guard system is done through the use of two DHT22 sensors. These sensors allow for precise recording of both temperature and humidity. One of these sensors will be placed inside the enclosure for temperature control, while the other will be placed on the outside of the enclosure for comparison. These sensors are connected to an Arduino Nano and communicate with it through serial communication. The sensors require three pins to run, the 5V Vcc pin, Gnd pin, and Data pin. The Arduino will relay the data through the I^2C communication protocol to the Raspberry Pi which will use the data.

The two figures below show the Arduino connected to the Raspberry Pi through the SDA and SCL pins. The data pins for the sensors will be connected to D3 and D2.

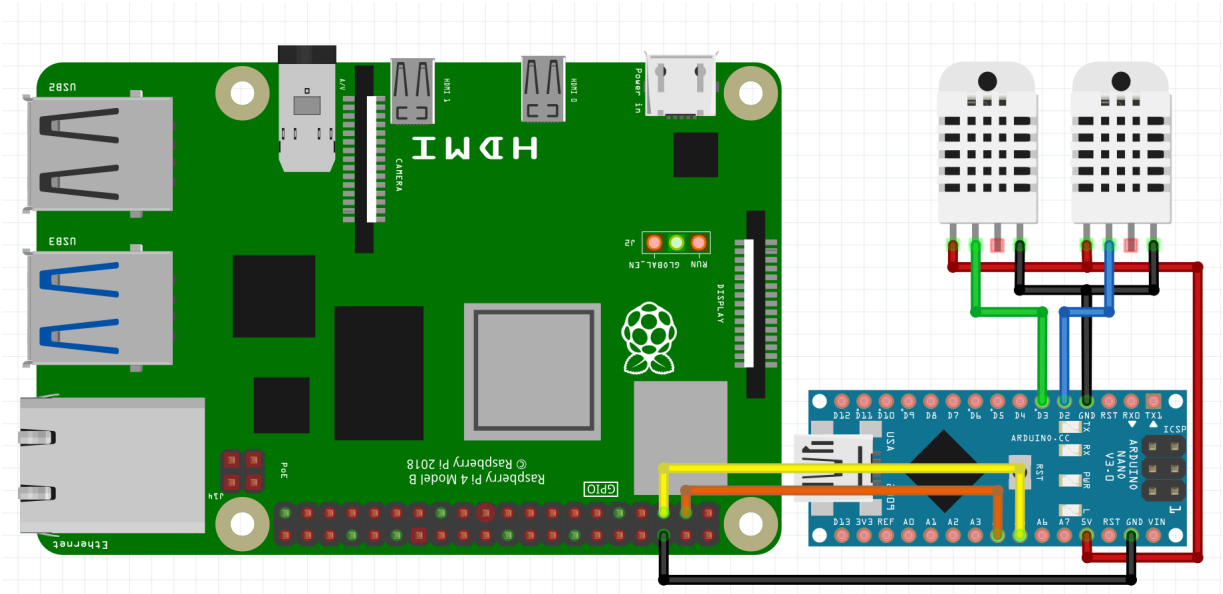

Figure 17: Breadboard View of Sensors

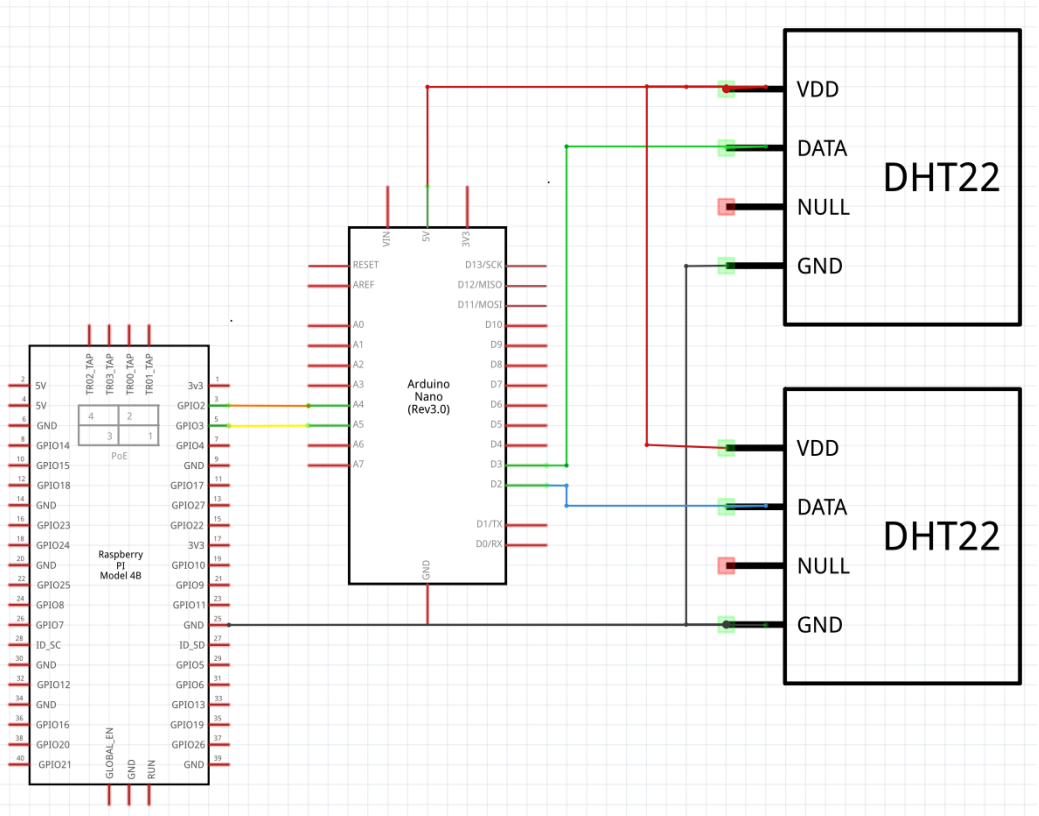

Figure 18: Schematic View of Sensors

4.2 Hardware Design for Component 2: Fan Control

To effectively manage temperature control within the Print Guard system, a fan control mechanism is designed to regulate the internal environment of the 3D printer enclosure. The schematic view illustrates the connection between an Arduino Nano and the fan, utilizing a PWM (Pulse Width Modulation) signal to adjust the fan speed dynamically based on the temperature readings from the DHT22 sensors.

The PWM signal allows for a variable fan speed control, where the Arduino generates a signal with a duty cycle adjusted according to the temperature feedback. If the internal temperature of the enclosure exceeds the predefined threshold, the Arduino increases the PWM duty cycle, thus increasing the fan speed to cool down the enclosure. Conversely, if the temperature falls below the threshold, the duty cycle decreases, slowing down the fan to maintain the desired temperature range.

The figures below show the connections between the RPi and the Arduino Nano as well as the Nano to the Fan. The fan is connected to the Nano through two wires, one for PWM control, and one for RPM sensor reading. The fan gets powered via 12V power source.

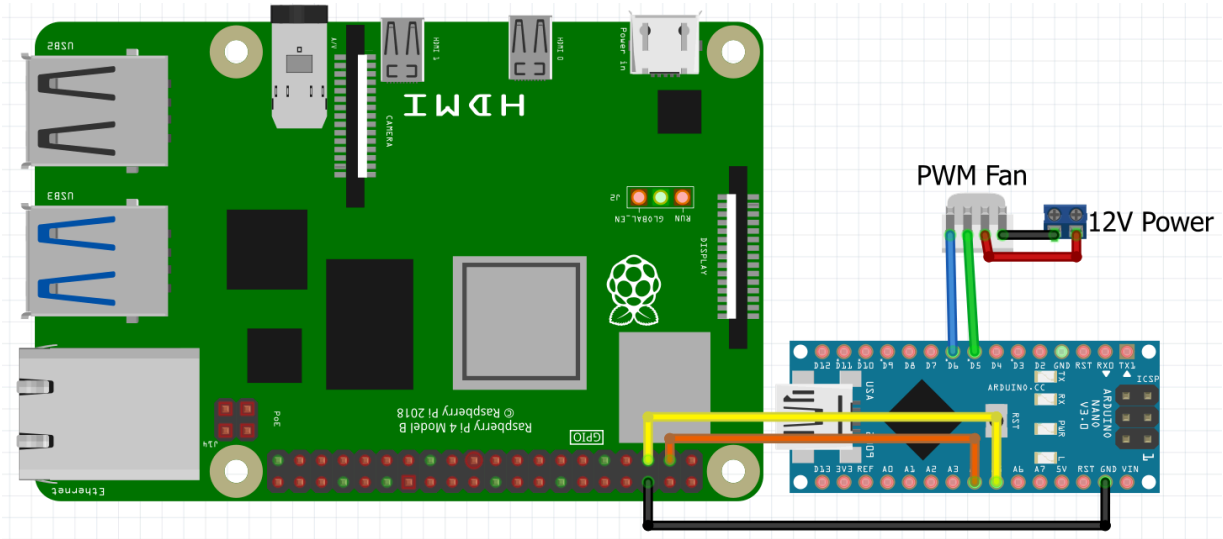

Figure 19: Breadboard View of Fan Control

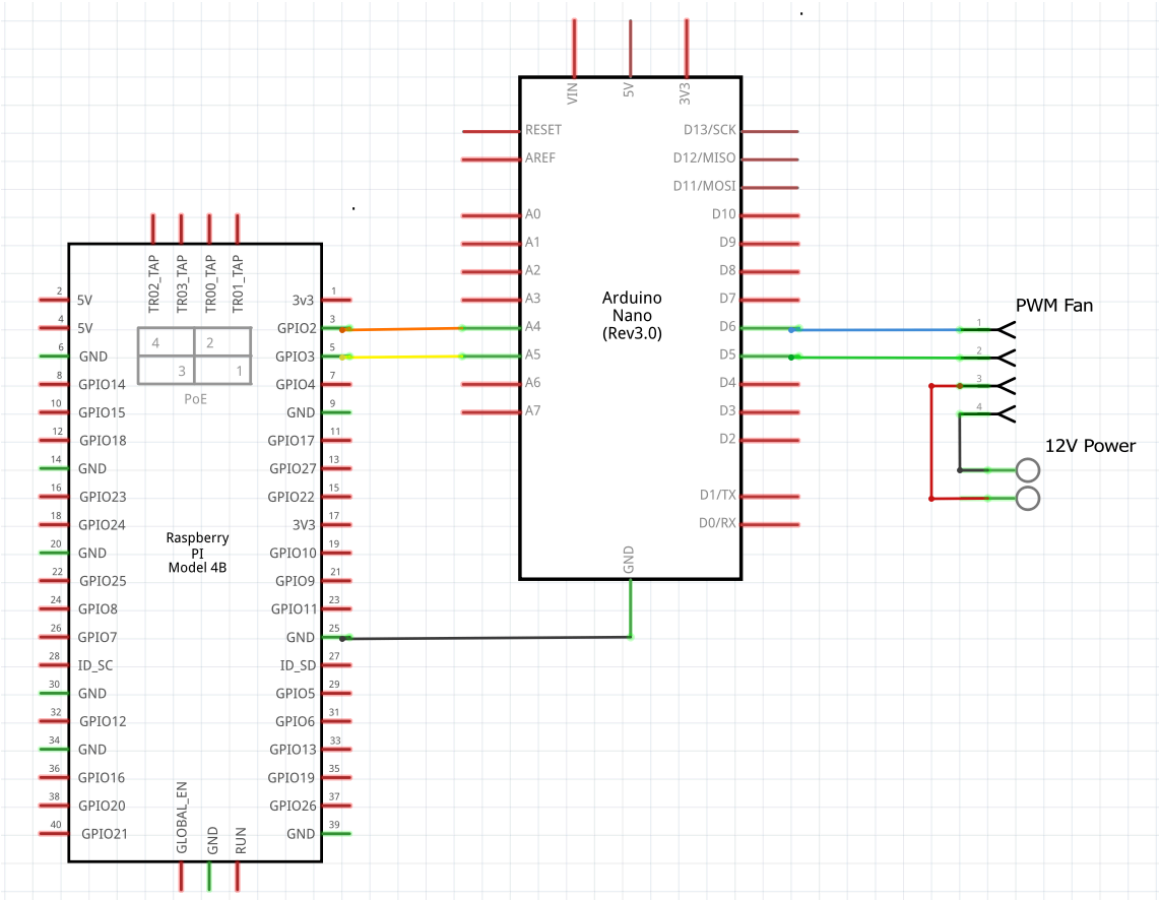

Figure 20: Schematic View of Fan Control

4.3 Hardware Design for Component 3: Camera

The RPi has a specific port for connecting the respective camera. This is located near the bottom middle just to the right of the “HDMI” text on the board. This connection will allow the RPi to control the camera to take photos and record video. This is done through the use of the picamera library.

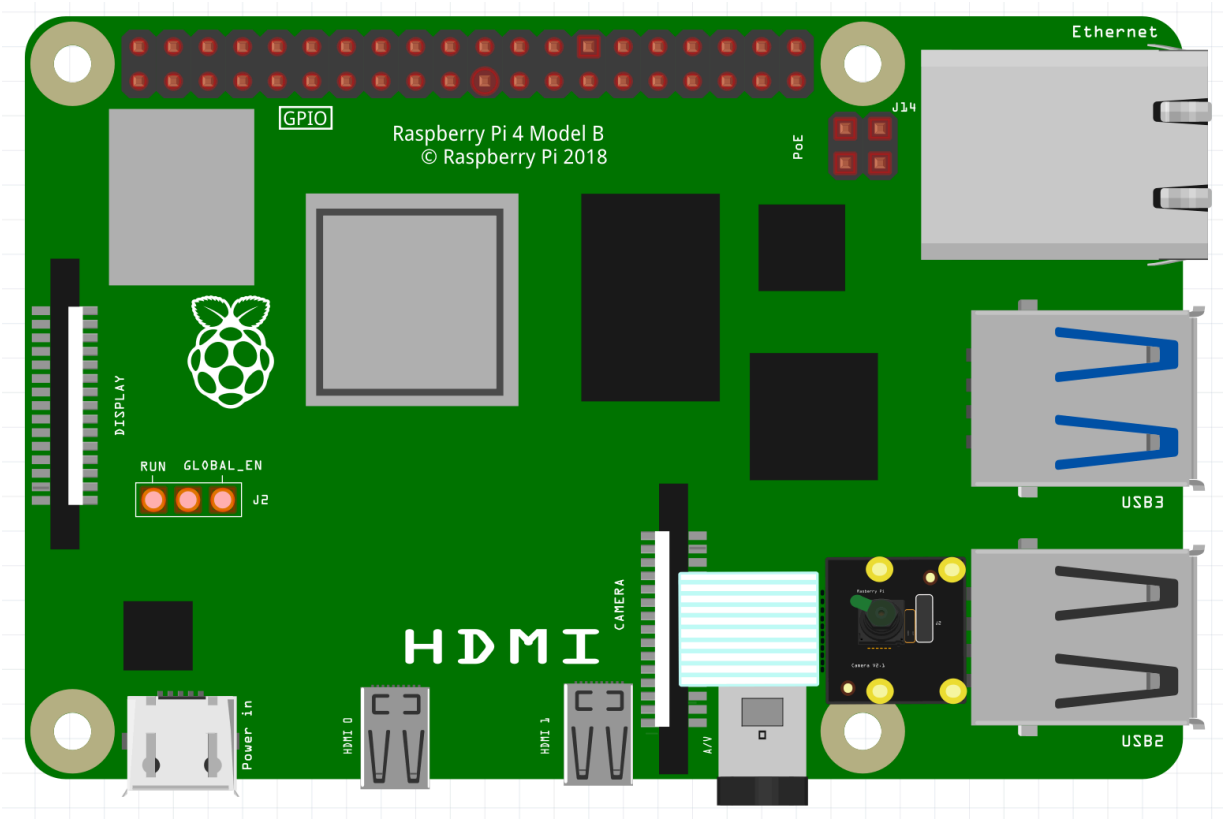

Figure 21: Raspberry Pi with Camera Module

5 GUI Design

The GUI Design for Print Guard aims to provide an intuitive and user-friendly interface for monitoring and controlling 3D printers remotely. It includes the following key features:

- **Dashboard**: A central hub displaying real-time data on printer status, temperature graphs, and ongoing print jobs.
- **Camera Feed**: Live streaming of the 3D printing process, allowing users to visually monitor their print jobs from any location.
- **Print Job Management**: Users can start, pause, or stop print jobs, and adjust printer settings through the GUI.
- **Notification Center**: Alerts and notifications for critical events (e.g., print job completion, errors, or temperature anomalies) are displayed, ensuring users are promptly informed.
- **User Management**: Admin users can manage access rights, adding or removing users and assigning roles to control who can view or control the 3D printer.

5.1 Users and Roles

- **Admin**: Full access to all system features, including user management, print job control , view camera feeds, and system settings.
- **User**: Can monitor print sensors, and print status but cannot change system settings or manage other users.

Time	Nozzle Temp	Bed Temp	Internal Temp	External Temp
12:05:00	111	30	22	21
12:05:15	134	34	23	21
12:05:30	156	37	23	21
12:05:45	187	41	25	21
12:06:00	200	43	26	21
12:06:15	215	46	25	21
12:06:30	217	50	26	21
12:06:45	214	51	27	22
12:07:00	218	57	29	22
12:07:15	215	59	30	22
12:07:30	216	58	31	22
12:07:45	214	60	33	22
12:08:00	218	61	34	23
12:08:15	213	60	33	21
12:08:30	215	60	32	21
12:08:45	215	59	31	21
12:09:00	216	60	31	21
12:09:15	214	60	32	21
12:09:30	215	60	33	21

Figure 22: Example GUI Design

6 Test Plans

6.1 End-to-end Communication Demo Test Plan

The Print Guard system is built up of multiple computers which will require communication that has been determined in Section 2.2. The main communication protocols for connecting the computers together will be I^2C, ssh, the Firebase API, and the prusaLink API.

The communication between the devices is key and that is why they must be tested to make sure the outputs are correct. These tests should cover all aspects of the communication functions. The test plan has been broken up into four parts consisting of testing the RPi to Arduino connection, RPi to RPi connection, RPi to Firebase connection, and the RPi to 3D Printer connection.

6.1.1 RPi to Arduino Test

1. This test starts with one RPi connected to two Arduino Nanos. Arduino 1 will be simulating sensor data through the use of an integer array.
2. The RPi will request data from Arduino 1 every 3 seconds through the I^2C connection protocol.
3. The RPi will then determine if the temperature is steady, increasing or decreasing.
4. Based on the result of the data collected, the RPi will send a message to Arduino 2 to either increase the speed, or decrease the speed using I^2C.
5. To visualize the message the Arduino received, a RGB LED will be connected to the Arduino, based on the message sent, the Arduino will change the color to either blue (too cold, decrease speed), green (perfect temp, keep speed), or red (too hot, increase speed).

6.1.2 RPi to RPi Test

1. Testing the connection between the two RPis will use both the SSH and I^2C protocols.
2. RPi 1 will send a command for RPi 2 to capture a photo with the camera.
3. RPi 2 should capture the photo and save it locally.
4. RPi 2 should then send the photo to RPi 1 through the use of the SCP command.
5. Once that is finished, the photo on both devices should have the same name and be the exact same.

6.1.3 RPi to Firebase Test

1. Testing the Firebase portion for end to end testing will confirm that the system can receive data from the sensors, and upload it to firebase.
2. The test will work by having RPi 1 receive the values from the Arduino code and pushing it to Firebase using the Pyrebase python library.
3. Once on Firebase, the data should be the same as the one the system will print out in the terminal window.

6.1.4 RPi to 3D Printer Test

1. Testing the RPi on the 3D printer will use the PrusaLink API connection.
2. The RPi will connect to the 3D printer and send a command.
3. The 3D printer will receive the command, and the change should be able to be visualized as the 3D printer executes that command.

6.2 Unit Test Demo Test Plan

To make sure the Print Guard system is operating correctly, the unit test for each component will run on startup. This will allow the user to ensure that the system is fully operational without any

faults. The unit tests have been broken up into two parts, the hardware tests, and the software tests

6.2.1 Hardware Tests

Unit Under Test	Description
DHT22 Sensors	The DHT22 sensors are the main sensors in monitoring the temperature. To test that these are operational, the system will send out a print statement with the current temperature and humidity. The sensors can also be tested through heating or cooling them to determine that the temperature is following that direction.
Fan	The fan will be set off to start, the system will confirm this through the use of the fan's RPM pin. The fan will then ramp up to max speed which can be confirmed by the RPM pin again.
Camera	The camera should be able to take photos and show a video feed. To test these features, on startup the camera will take a test photo and display the current video feed for a while.
3D Printer	The 3D printer can be tested by making a call to the system and testing that the call works visually on the 3D printer.

6.2.2 Software Tests

Unit Under Test	Description
Database Control	On startup the system will send any unit test data to both databases with a special symbol to separate the data from normal data.
Website Control	On startup the website will launch. This website will allow for monitoring of the Print Guard system as well as the control of the 3D printer. To verify that the website is operational, the website will prompt the user to verify that the system is operational. When the button is clicked, the website should display all the normal information.

Notifications	The notifications will be very important for remote monitoring of the 3D printer. On startup the system should send out a notification saying if all unit tests have passed or failed.

6.3 Final Demo Test Plan

The Print Guard system will be able to be tested through this final test demo. These tests should be able to validate all the functional requirements of the system.

6.3.1 Temperature Control

The Print Guard system will be periodically fetching sensor data to update the fan speed for temperature control. The system will also be able to determine if the values are in a good range to determine if there is an error. The system should be able to measure the temperature and humidity inside and outside of the enclosure, as well as being able to monitor the 3D printer's nozzle and bed temperature. Based on the internal temperature the system will monitor and adjust the fan speed.

6.3.2 Remote Camera View

The remote monitoring feature in the Print Guard system will not just allow for remote sensor monitoring but also allow for remote view. This view will be able to be accessed through the website under a password for security purposes. The user will also get a photo every time there is a notification sent. The system will also store photos on the databases periodically so the user can go back and view the print photos.

6.3.3 3D Printer Control

The Print Guard system will not only allow for monitoring but also allow the user to control the 3D printer in the event of an error. The user will be able to remotely shutdown the 3D printer when they can see that there is an error with the print.

6.3.4 Notification System

The Print Guard system will help to notify the user of potential errors in the system. These errors can be different sensors reading too high or too low temps, the fan speed not increasing when the system is telling it to, or potentially messages about a crash. The system will also notify the user of prints finishing. These messages should be sent with a message saying what is happening with an accompanying photo.

7 Project Update

The progression for the Print Guard system is moving well. The enclosure that surrounds the 3D printer is almost finished. The last part needed is adding the walls and making a mount for the fan and DHT22 sensors. The connections between each computer is being tested to make sure they are all working as intended. The individual hardware components like the DHT22 sensors, fan, and camera are all being worked on. The SQLite and Firebase database templates have been created and are ready for information. The base of the notification system is working and we are able to send notifications to Discord from the RPi.

7.1 Project Milestones

The milestones for the project have been moved around a bit as we had to wait for components to arrive to start working. The milestone table can be located below.

Milestone	Week
Read DHT 22 sensor data	Week 4 -> 6
Control Fan using PWM	Week 4 -> 6
Stream Video Feed over Internet	Week 5 -> 7
Send data between Arduino and RPi (completed)	Week 5
Send 3D printer Data to RPi (completed)	Week 5
Send All data to Firebase	Week 6
Send Notification from RPi (completed)	Week 7
Create Simple GUI to display data	Week 7
Setup crash detection server	Week 8
Refine GUI with graphs and live video feed	Week 8
Implement AI crash detection	Week 9
Implement security for camera on GUI	Week 9
Assemble basic product	Week 9
Test basic product	Week 10
Implement AI on product	Week 10
Assemble Final Product and test	Week 11

7.2 Schedule of Activities

The schedule of activities from the proposal can be found below. The only changes to the timeline would be the changes located above in the milestone table. These changes would be moving the DHT22 sensor data, fan control to week 6, and streaming the video feed to week 7.

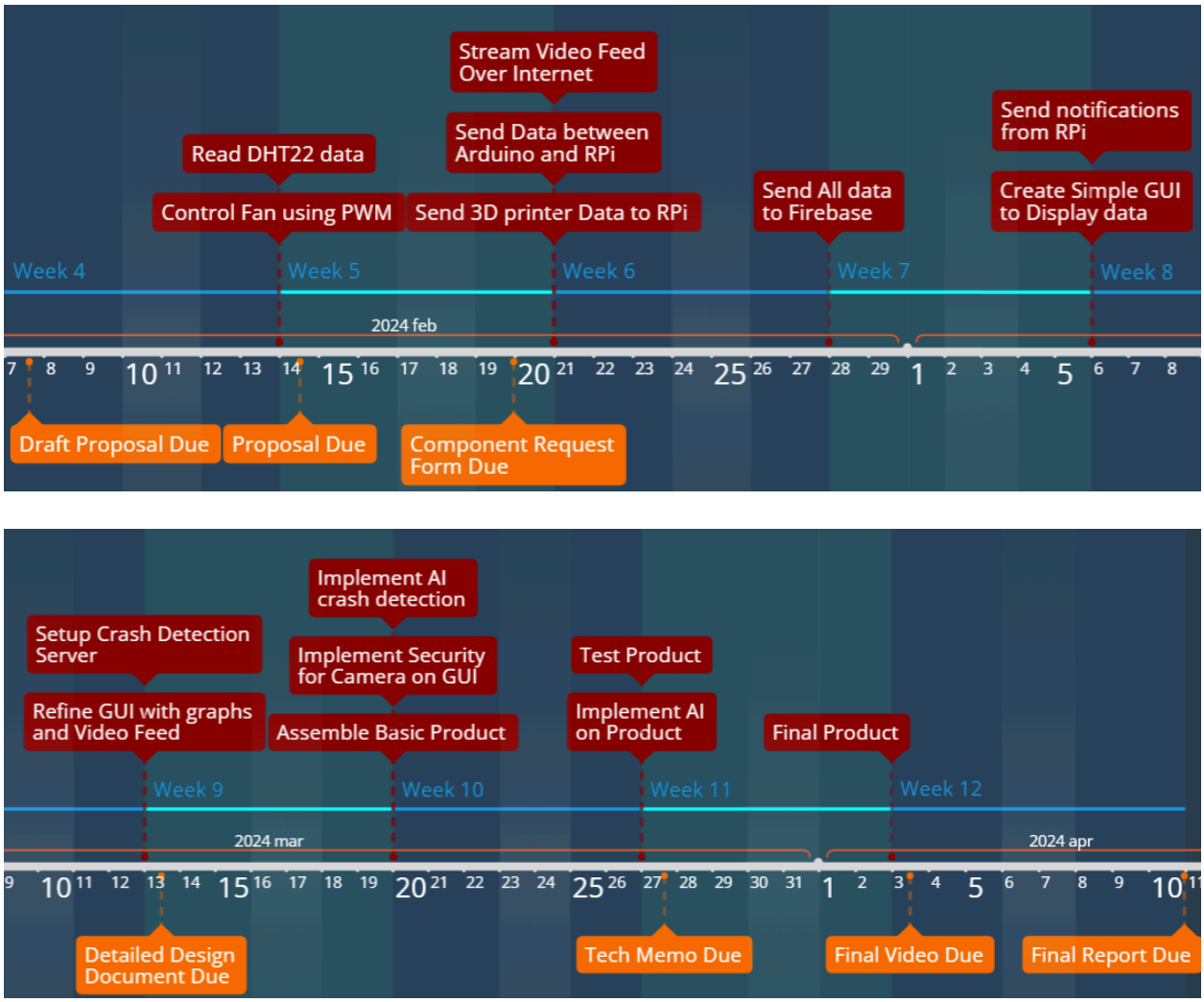

Figure 23: Project Timeline

References

[1] Bourell, D. L. (2016). Perspectives on additive manufacturing. *Annual Review of Materials Research*, *46*. https://www.annualreviews.org/matsci/doi/10.1146/annurev-matsci-070115-031606

Breakdown of Tasks

Name	Tasks Completed
Joseph Vretenar	Sections 1, 2.1, 2.2, 2.3.3, 2.4, 3.4, 3.5, 4.2, 4.3, 6.1.2,4, 6.2, 6.3.2,3, 7
Samuel Mauricla	Sections 2.2, 2.3.4, 2.2.3, 2.4, 3.2, 6.1.3, 6.3.4, 7
Basma Aboushaer	Sections 2.2, 2.2.3, 2.3.1, 3, 3.1, 4.1, 6.1.1, 6.3.1, 7
Raunak Singh Soi	Sections 2.3.6, 5
Cholen Premjeyanth	Sections 2.3.2, 2.3.5, 3.3