Subsystem Report

1. Microcontroller Subsystem Report - Grace Kent

1.1. Subsystem Introduction

The Nvidia Jetson Orin Nano functions as the main brain for PLSS. The microcontroller will connect all peripheral hardware with the web server and app. Furthermore, video recordings are captured using the Jetson. The video stream is processed with the mask, so any video saved has only the user defined area.

1.2. Subsystem Details

The Nvidia Jetson Orin Nano is the main microcontroller of the system. The Jetson is used to communicate with the motion microcontroller through UART, along with an accelerometer and three passive infrared detectors. The Jetson has a forty pin header, these pins are used for UART and I2C communication.

The microcontroller is used to monitor the sensors, and depending on the sensor data, events are triggered. A flow chart of the program is shown in Figure 40. When the program first starts the sensors are written to ensure the desired settings and to check if the sensors are operating properly. Events are triggered using the PIRs and accelerometers. The threshold for the PIR is based on information from the data sheet, and empirical data. The accelerometer's threshold is based on the standard deviation and mean of a baseline measurement.

At the start of the program multiple tasks are started. In a docker container, the human detection script is run so that it can utilize the GPU. The rest of the code runs simultaneously on one script. There are four main categories: server, AI, GPIOs, and image processing. The main script starts the processes using multiprocessing so all tasks can be run concurrently. There are multiple variables shared between each to use for logic.

After a panorama is taken, the basis for the accelerometer is taken to calibrate the accelerometer. The accelerometer's values are dependent on orientation (for instance if lying flat the z value of acceleration is 1g, or 9.81m/s²), and readings will be compared to the basis to see if motion has occurred. The basis is taken by measuring the accelerometer over a certain period of time, and then averaging these values. The thresholds are set using the standard deviation of the basis measurements. Within three standard deviations should account for any noise (around 99%), so anything outside of the three standard deviations of the mean can be considered movement.

Once the accelerometer basis is taken, the gpio loop is started. The accelerometer and PIR sensors are continuously read until camera motion or object motion is detected. Once object motion is detected the recording will start. While recording the sensor data will still be updated.

The recording will last for at least thirty seconds, if object motion is detected while recording, then the recording will continue. The recording only ends ten seconds after any motion is detected and there is no live streaming. Videos are named for the time when the recording started. Once a video ends the video will be added to the AI queue. A queue is used for facial recognition processing because a video can take longer to process than a recording, so if a video is ended while the AI processing is working, then that saved video will not be processed.

The gpio loop is infinite, and will skip reading the pirs, as there has been tampering and the area being recorded could be illegal. Furthermore in the recording logic it is nothing .Once tampering/camera motion is detected any current video is ended. So if recording, the recording will end and be saved.

There are three passive infrared detectors, this is done so that object motion can be detected for the entire range of motion (180°). Depending on which sensor detects motion, the motion system will be commanded to turn to a desired value/angle. This angle value is determined by what PIR is triggered, and if multiple are triggered.

Configuration from the app is also used in logic. For instance if the camera is turned off in the app then video will not occur. Moreover, if camera motion is off then motion tracking using the PIRs will not occur.

Frame capture was separated into a separate process, so that it can be constant. This prevents inconsistencies in the buffer, and reduces lag when starting a video. Serial read write with the motor control also needed to be separate. This is accessed by pano generation, motion tracking, image processing, and the server. Using a queue each can concurrently send move or read commands to the motor, without causing conflicts.

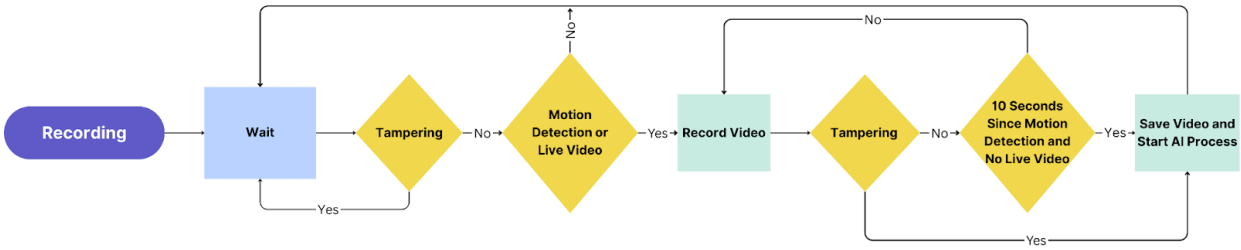


Figure 40. Recording Process

The recording process is set up so that the main state is waiting. If tampering is detected, wait is continued, otherwise the system waits till live streaming or motion detection occurs. Once recording occurs, the video will only be ended if tampering is detected, or once live streaming is false and it has been ten seconds after motion was last recorded.

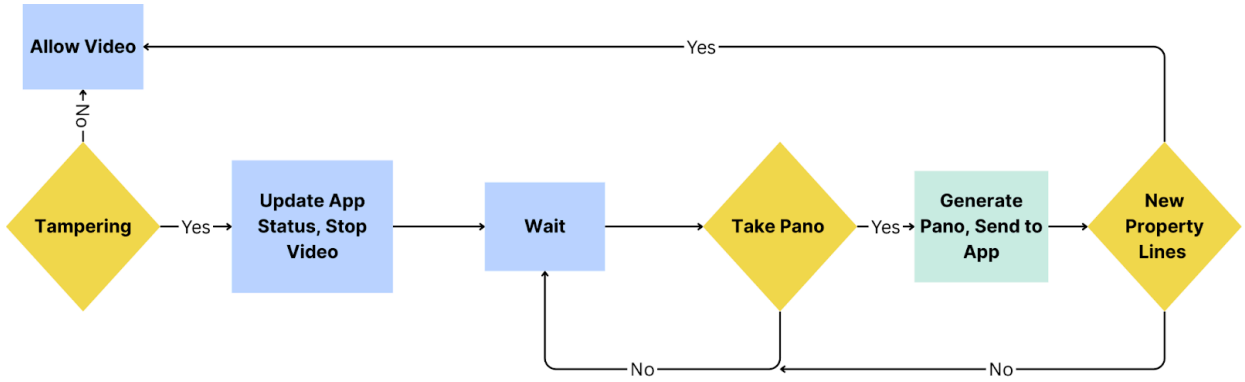


Figure 41. Tamper Detection Process

Tampering detection is monitored so that in case of the camera being shifted we can prevent recording. If the position does not align with the original, the property lines most likely are no longer true, so a recording is now illegal. Once tampering is detected, the tampering flag remains true, and the system will wait for the user to start the pano creation process and upload new property lines.

1.3. Subsystem Validation

Validation for the Jetson is done by running through scenarios and monitoring the response of the system. Furthermore, since the thresholds for events are set in the program, the exact sensitivity can be changed. The system also has some self validation built in as the program will not run if the sensors are not communicating properly.

To validate the system the program is run. At a known time motion is triggered, motion is only triggered once, after ten seconds the video is saved, with the known time of motion as the name. It was also validated that additional motion will extend the video recording, at 10 seconds after the first motion, motion is triggered again. Video will continue to be captured for another thirty seconds. An important function is the infinite loop, so motion is triggered after a video is saved to begin another recording.

The accelerometer/camera motion is validated as well. The accelerometer had some random data, so the threshold had to be increased to accommodate these erroneous readings. Furthermore, acceleration is only seen as true if over three consecutive readings surpass the threshold. It was validated that camera detection will be triggered if the accelerometer is bumped. When bumped the system will wait for a new set of property lines. Once new property lines are uploaded these new property lines are applied to new videos.

Different app inputs are also used to validate the server integration with the code. More validation data can be found in the integrated system validation.

1.4. Subsystem Conclusion

The main microcontroller subsystem works as currently intended and is fully integrated.

2. Peripheral Sensors Subsystem Report - Grace Kent

2.1. Subsystem Introduction

The sensor peripheral hardware is meant to connect multiple sensors with the microcontroller, while also integrating power and motion systems. The peripheral hardware consists of three passive infrared detectors (PIR), an accelerometer and a buck convertor to power the sensors. Connectors are also integrated to interface between power, motion, and the microcontroller. The microcontroller will use these sensors in order to record video, track motion, or prevent video (detection of tampering).

2.2. 3.3V Buck

2.2.1. Operation

The AP63203WU-7 is used as the buck converter. The voltage output from the power system is used as the input to the buck, which in turn outputs 3.3V. The output range of the power system varies as batteries discharge, but the buck has an input range of 3.8V-32V, so the battery voltage is comfortably within the operating range of the buck.

2.2.2. Validation

To validate the buck different loads are applied and the output voltage is measured. The buck should provide a steady output voltage if the input voltage fluctuates. Validation was done using a 9V battery, which is the lowest operating voltage of the microcontroller.

Table 2: Buck Loading and Output Voltage

Load	Voltage Output [V]
No load	3.41
1 Device - PIR	3.41
2 Devices - 2 PIR	3.4
3 Devices - 3 PIR	3.4
4 Devices - 3 PIR, 1 Accelerometer	3.4

The output voltage is mostly stable, with a slight drop as load increases.

2.3. Passive Infrared Detector

2.3.1. Operation

The system consists of three passive infrared detectors (PIR), which are used to provide a 180° field of view. The PIRs are used to trigger camera recording and the motion subsystem. This provides a wider field of view with one camera, while reducing power consumption. Video increases power consumption, so by reducing the time video is being recorded, the power consumption is lessened.

The EKMC2606111K was chosen, it provides a max range of 12 meters and field of view of 62°. PIRs are used to detect object movement, though are only reliable at farther distances with human sized targets (700x250mm) moving at speeds of at least 1m/s. The PIR chosen is an analog device, so an 8-bit ADC (ADC081C021CIMM/NOPB) is used to convert that analog value to a binary one.

Since the PIR is analog the voltage when no motion is detected rests around 1.1V. When an object enters the positively polarized detection zone the voltage should increase, and when the object enters the negatively polarized detection zone the voltage decreases. Sensor sensitivity can be adjusted, to do so thresholds can be set in the microcontroller code to determine when object movement has actually occurred.

A PIR operates by detecting a change in the infrared intensity/luminosity using the pyroelectric effect. Intensity of light has an inverse relationship with distance, so a close object will have a greater intensity or power. The intensity of light is affected by the heat of the target, as well as the speed and distance of the target. Heat is radiated as waves within the infrared wavelength range. The infrared light is absorbed by the detector, and due to the pyroelectric effect a voltage and current are induced. However, current only actually flows when there is a difference in temperature between an object and the surroundings. Therefore, due to the working principle of the passive infrared detector the strength of the response is reduced when the target is smaller, moving slowly, farther from the detector, or the outside temperature is higher.

2.3.2. Validation

Validation is done by recording the voltage output of the PIR detectors using the microcontroller. The voltage output of the PIRs are read by ADCs, and are converted to an 8-bit binary value. A value of 3.3V the corresponding value is 11111111, or 255, this is sent to the microcontroller and can be converted into a voltage value. At distances of 5ft to 30ft human walks and the response is recorded.

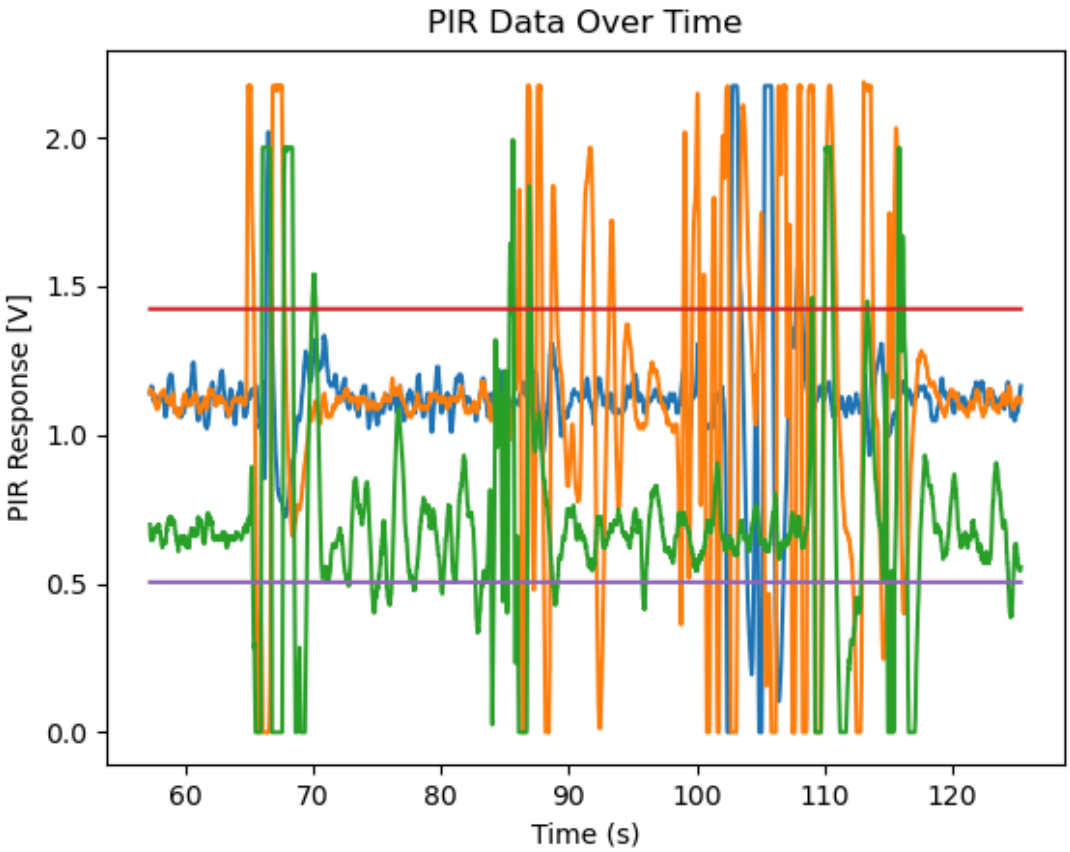


Figure 42: PIR Voltage Response at 5 Feet

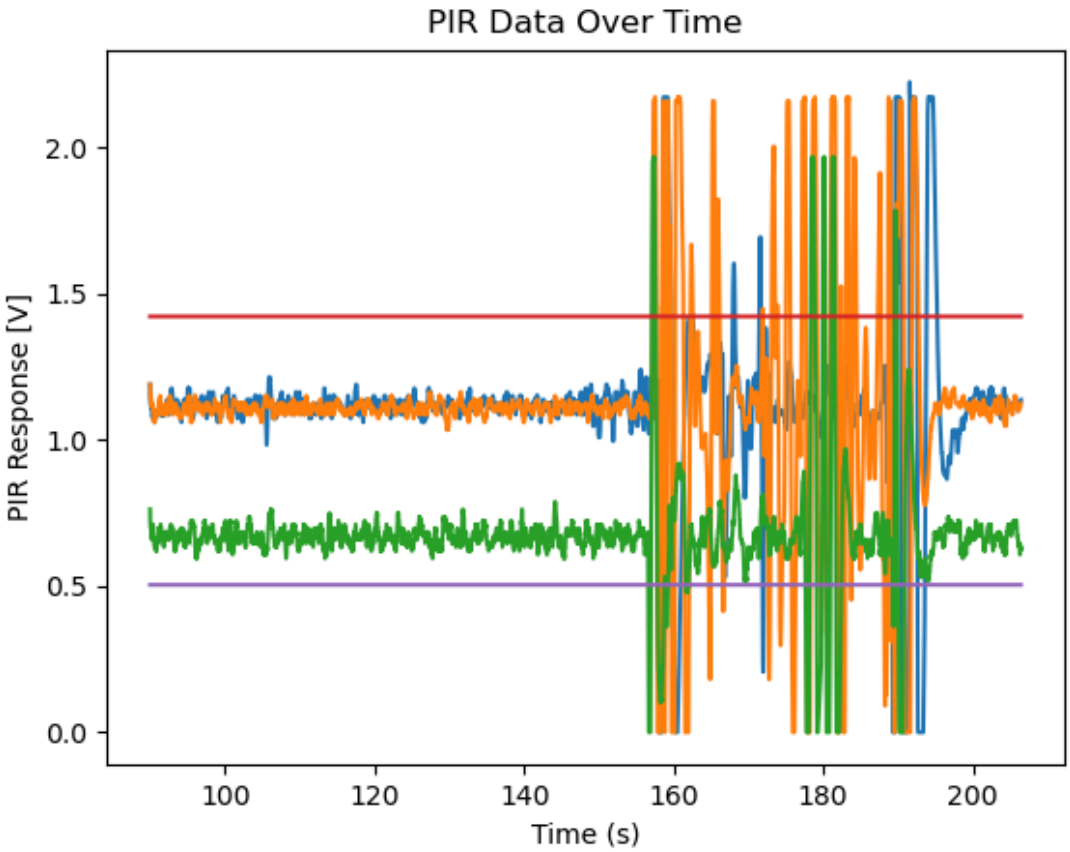


Figure 43: PIR Voltage Response at 10 Feet

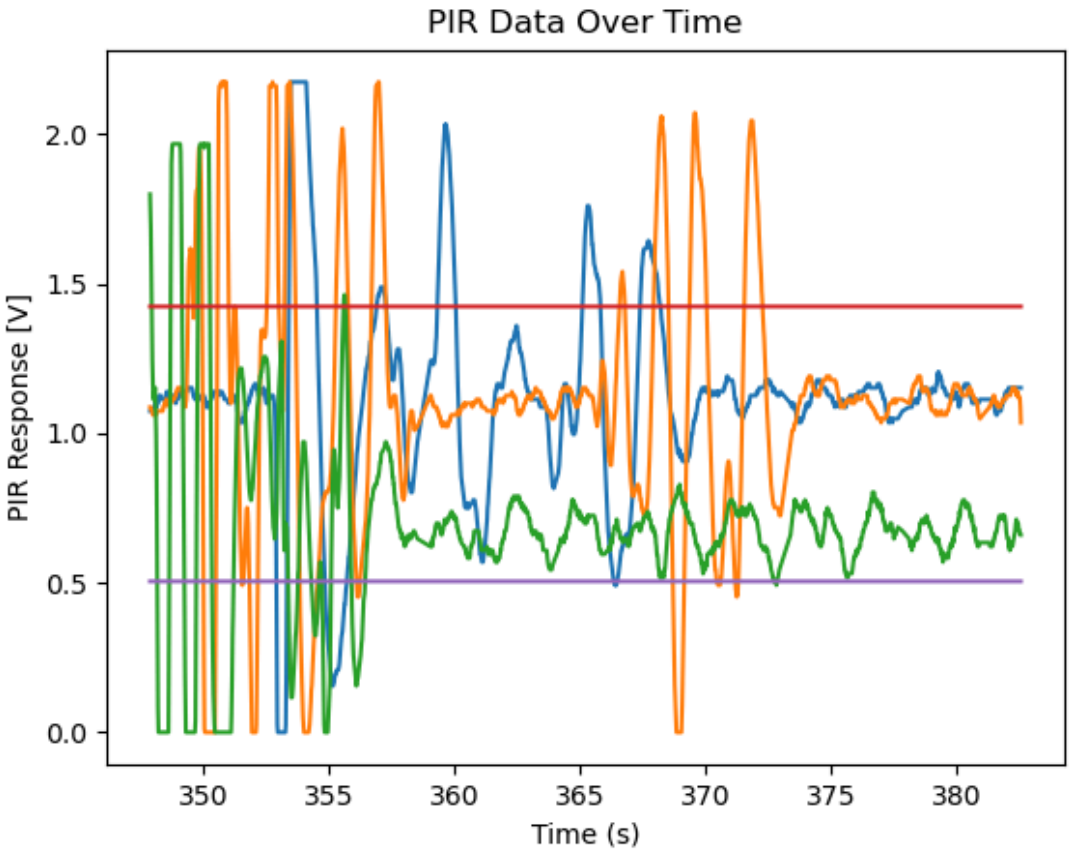


Figure 44: PIR Voltage Response at 15 Feet, with Threshold

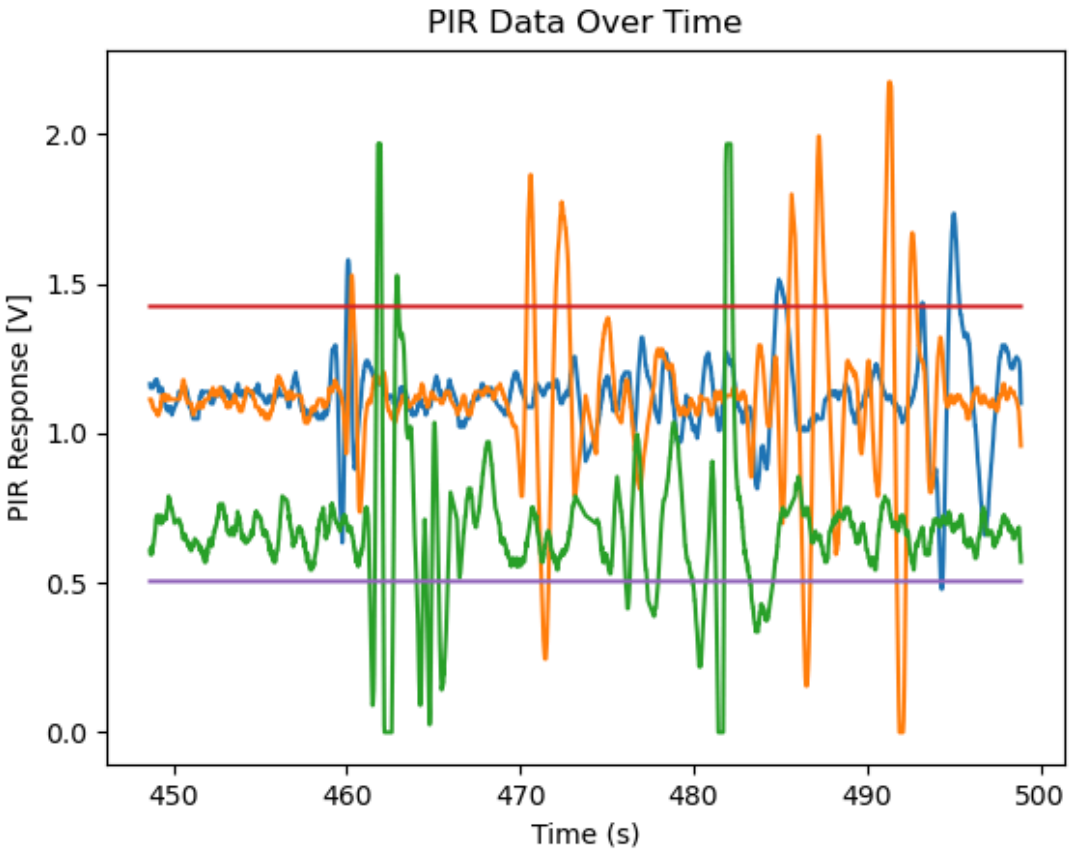


Figure 45: PIR Voltage Response at 20 Feet, with Threshold

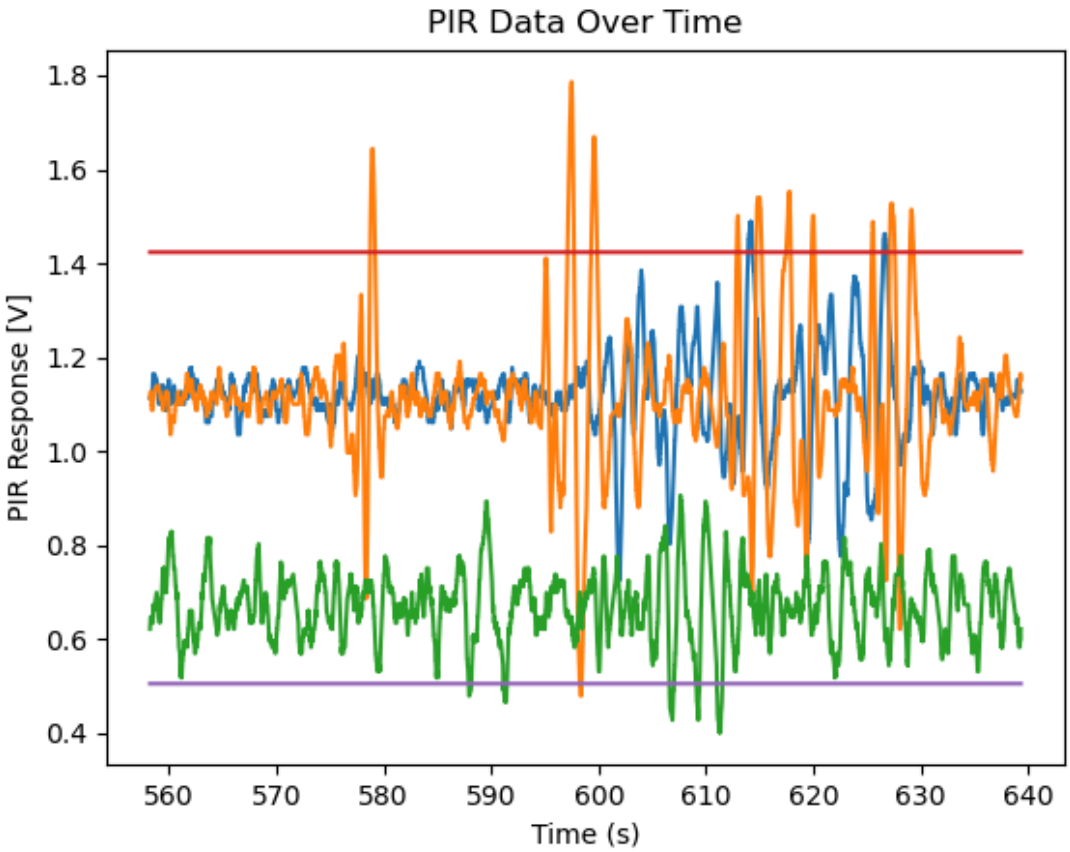


Figure 46: PIR Voltage Response at 25 Feet, with Threshold

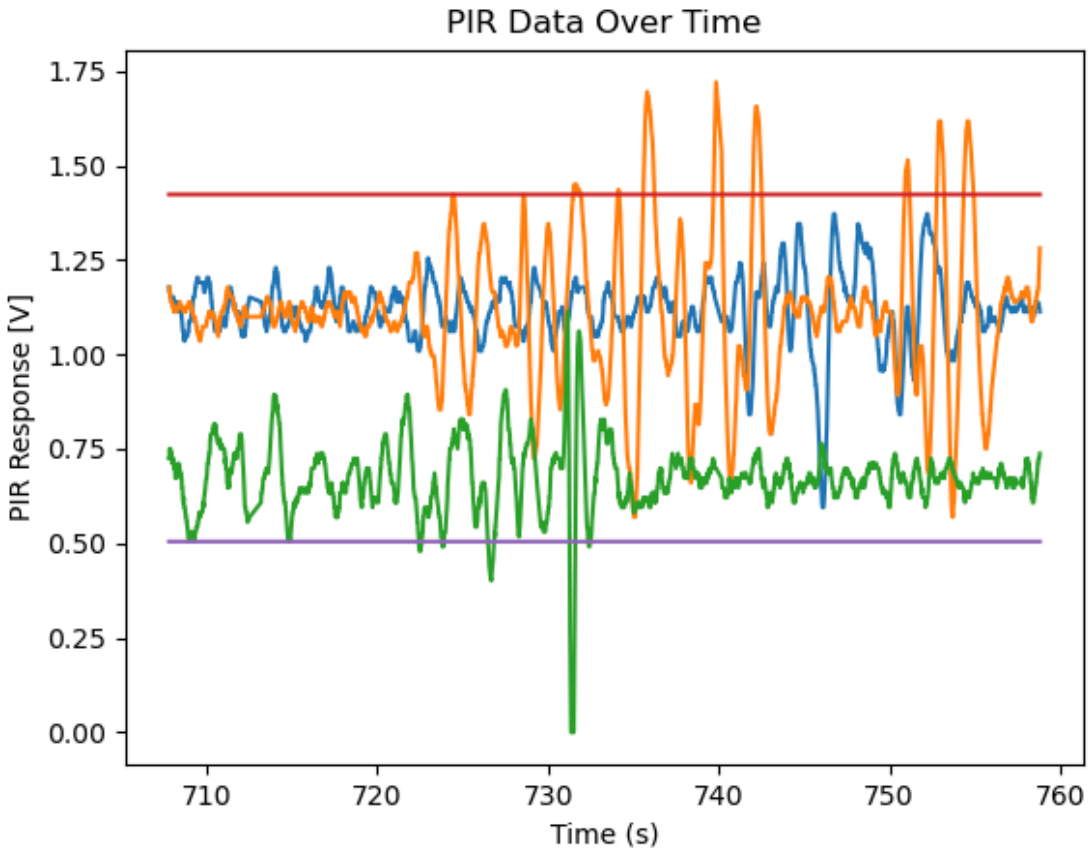


Figure 47: PIR Voltage Response at 30 Feet, with Threshold

In figures 43 through 48 the thresholds set in the code are displayed. These thresholds are 1.4V and 0.5V. It can be seen that there is a clearly saturated response at distances of at least ten feet. After twenty feet the negative response is greater than the threshold value, but above the positive response crosses its threshold. A distance of 27ft to detect movement should be adequate, and past that the threshold can be adjusted if needed since the threshold is arbitrary. The range of the PIR is 36ft, so adjusting the threshold should extend the range.

It is important to note the conditions in which the data was taken. The data was taken on an April day after 3pm, while there was a UV index of 8, and temperature around 87°. The operation of infrared sensors is based on temperature differences. Therefore, there is a clear degradation in PIR response when temperature is close to skin temperature (typically 92°F to 98°F), and as incident sunlight increases. Furthermore, there was a large bush in the area that one PIR was measuring (green line), which is why its response data is lower than the PIRs.

The PIR response can clearly be seen to be reliable up to 30ft. The manufacturer states reliability up to 36ft. It has been noticed that objects can reliably seen at 60ft. Multiple cars at a range of 60ft were tracked. The PIR response depends on the size, speed, and temperature of

an object. A human walking at 60ft is not able to create a great enough response, but a car at 30mph can.

Using the signal readings from the PIR motion tracking is implemented. However, due to the nature of the output signals of the PIRs, the resulting camera motion was jittery. To smooth out this jitter, additional logic was added in the embedded code to transform this into a digital signal.

Figure 48: PIRs Response with Additional Logic to Create a Digital Signal

The digital signal can be seen in the figure above, and means that camera motion was smoother. Further validation of PIR response and position can be found in validation of the overall system.

2.4. Accelerometer

2.4.1. Operation

The accelerometer is used to detect unauthorized movement. Since it is illegal to record other properties, and the boundaries are set, if the camera is moved then it is likely that the recording will include other properties. To prevent illegal recording the accelerometer is used to detect if the camera is tampered with.

Comparing readings to previous values, it can be determined if the camera is moved. The values must be compared as the accelerometer will read a non-zero value when the accelerometer is still.

2.4.2. Validation

The accelerometer was tested by comparing the value read on another accelerometer, to the camera's accelerometer. Furthermore, the accelerometer can also be moved at the known accelerations. The key importance is that a bump should be detected.

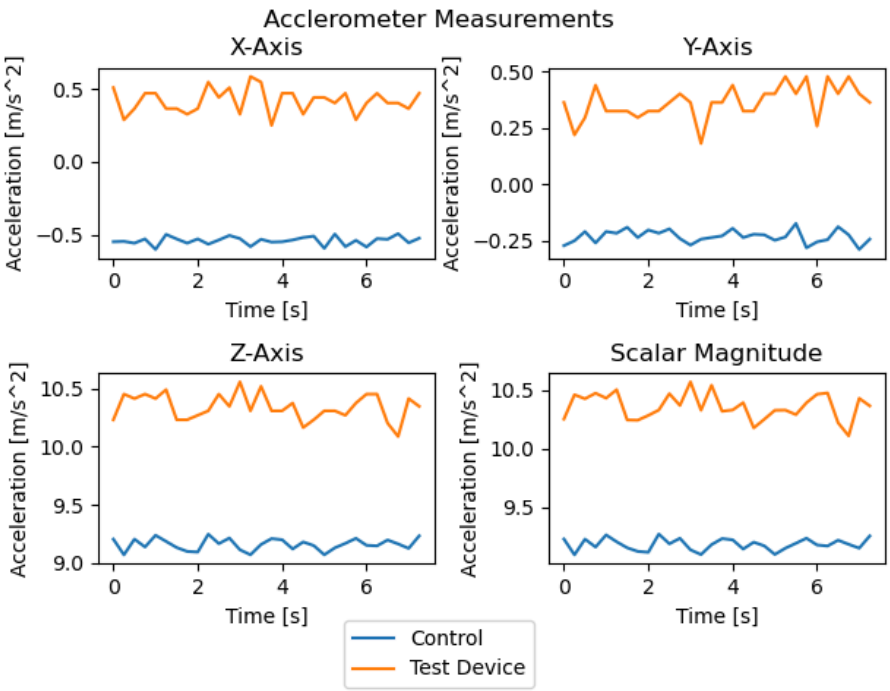


Figure 49: Accelerometers at Rest

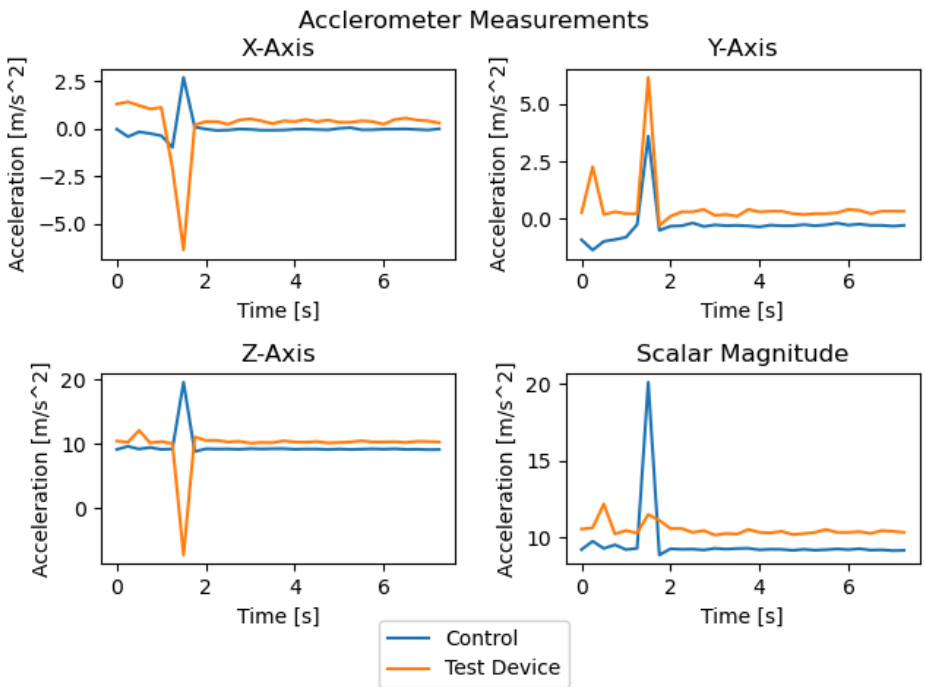


Figure 50: Accelerometers Dropped

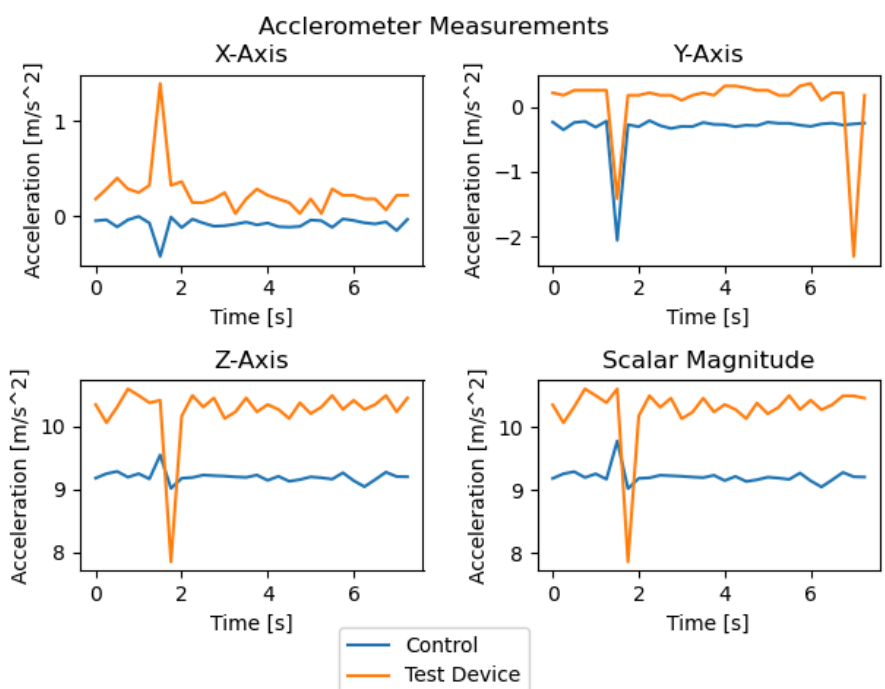


Figure 51: Accelerometers Bumped, 1

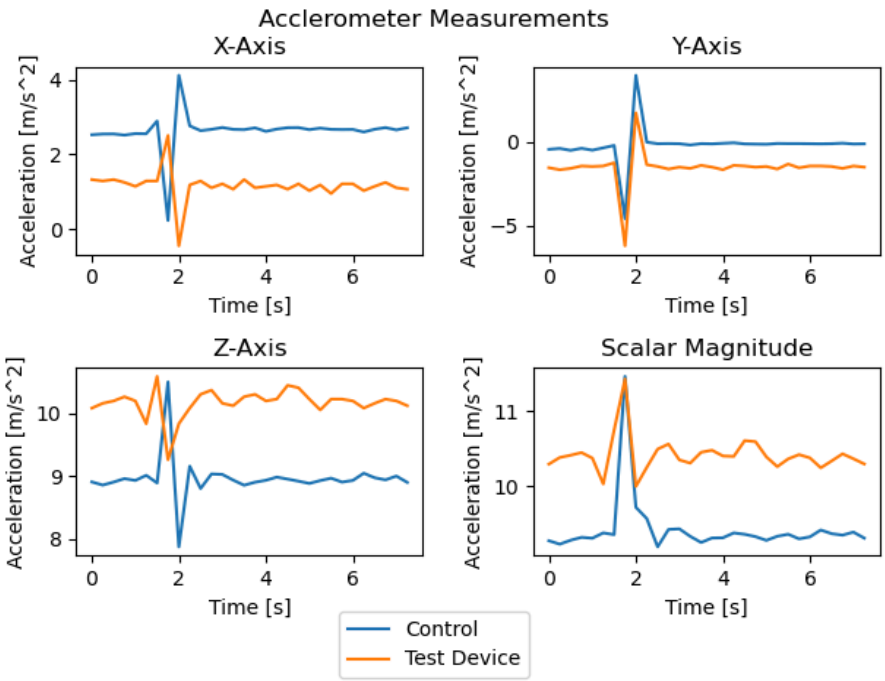


Figure 52: Accelerometers Bumped, 2

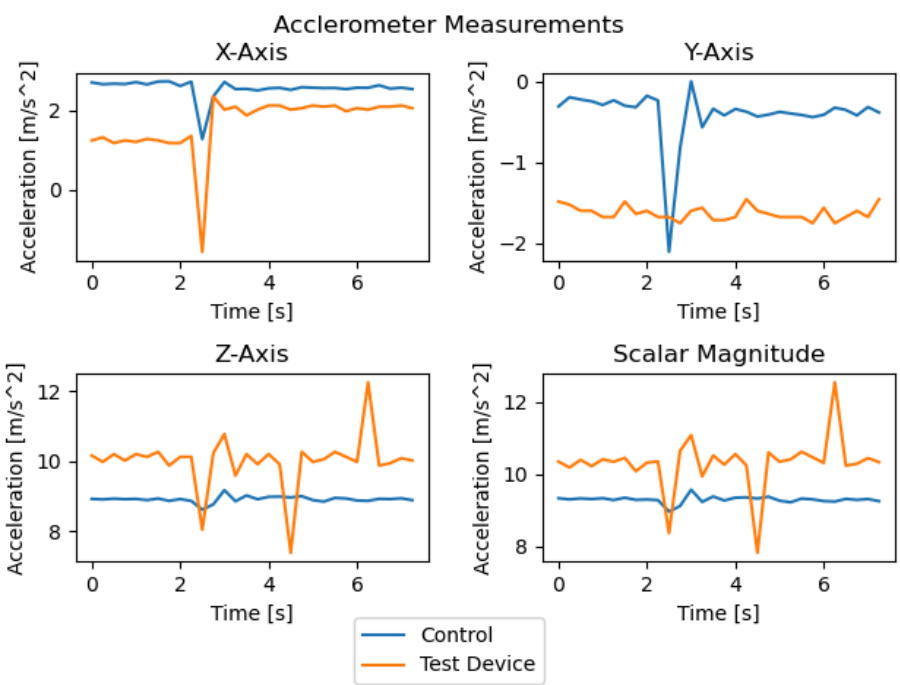


Figure 53: Accelerometers Bumped, 3

The accelerometers were measured at rest, dropped, and bumped. Both devices have different average acceleration values at rest, but this can be ignored because it is the difference between the at rest value and value at a certain time that matters, not the actual value. In each figure the

control device is generally less sensitive than the test device. However, the test device also has greater noise. The test device has greater variance, and even false positives.

The noise affecting the accelerometer became greater as integration progressed. This noise would consistently trigger tampering.

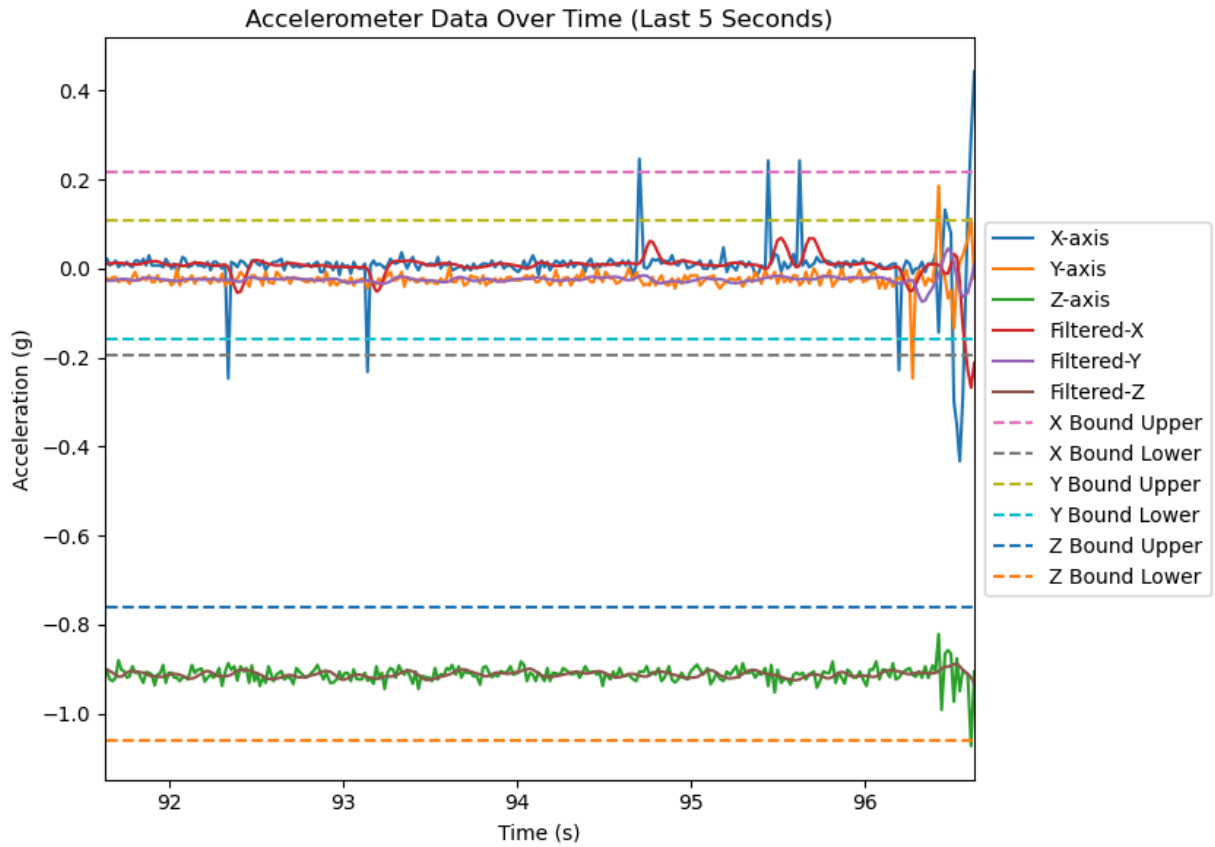


Figure 54: Accelerometer Data over Time, Raw Data, Filtered Data, and Bounds

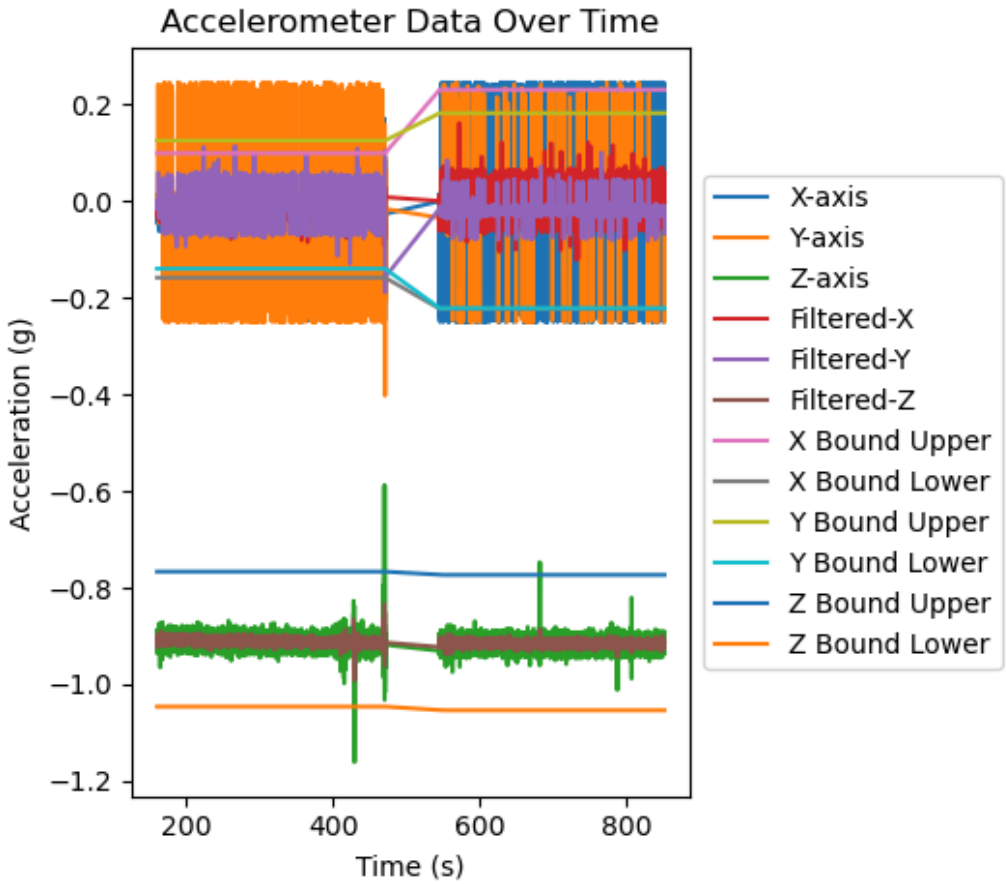


Figure 55. Accelerometer Data

The noise is the most significant on the X-axis. To mitigate this a digital filter was implemented, reducing the noise and eliminating false positives. This allowed for tampering to be accurately detected.

The noise and high sensitivity of the accelerometer is mitigated by implementing a digital filter, meaning the system works as intended, and accurately detects unauthorized motion.

2.5. Connector Board

2.5.1. Operation

The connector board allows for all peripherals to interact with the Nvidia Jetson Orin Nano. Furthermore, this board connects power from the battery board to all peripherals.

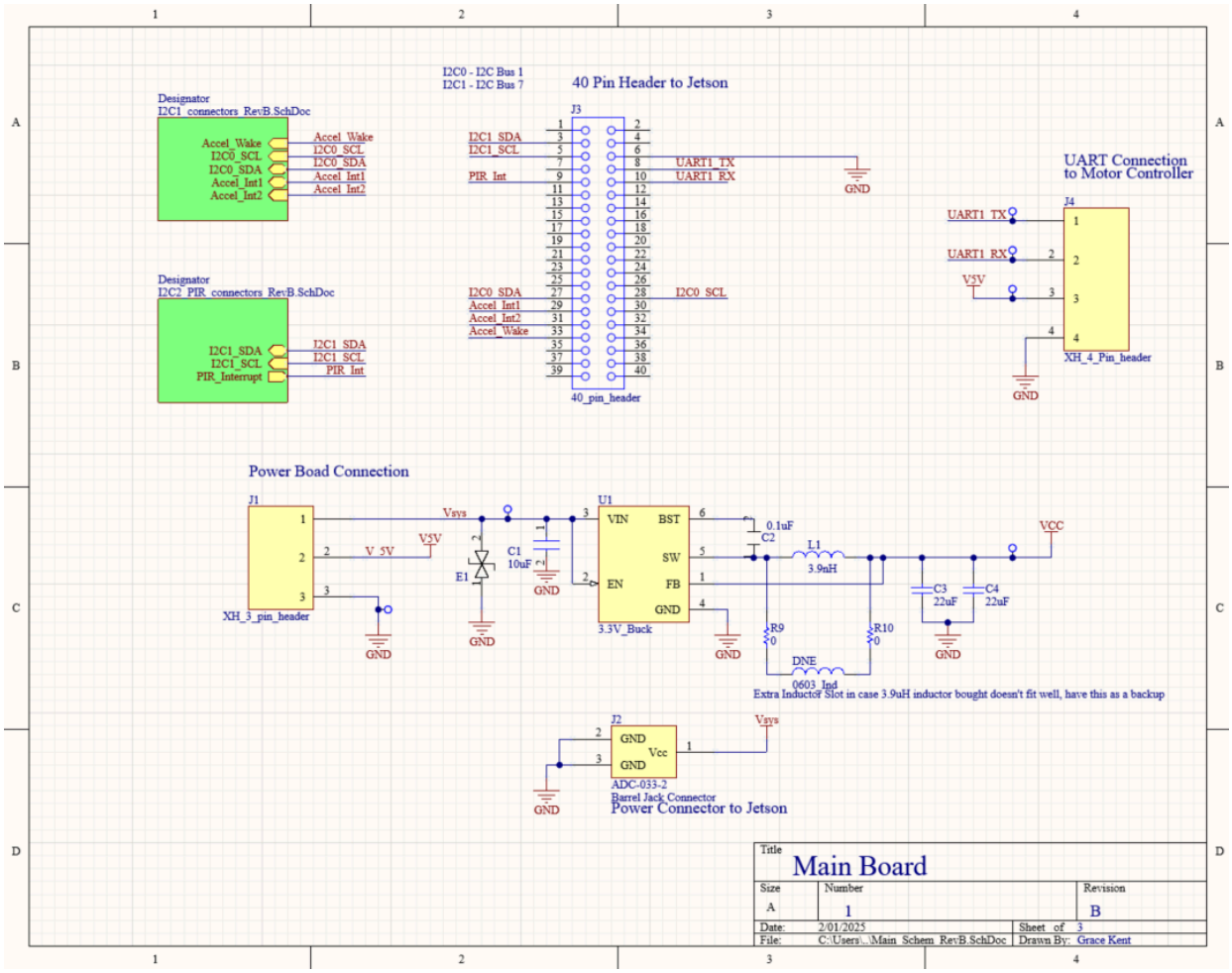


Figure 56: Connector Board Schematic

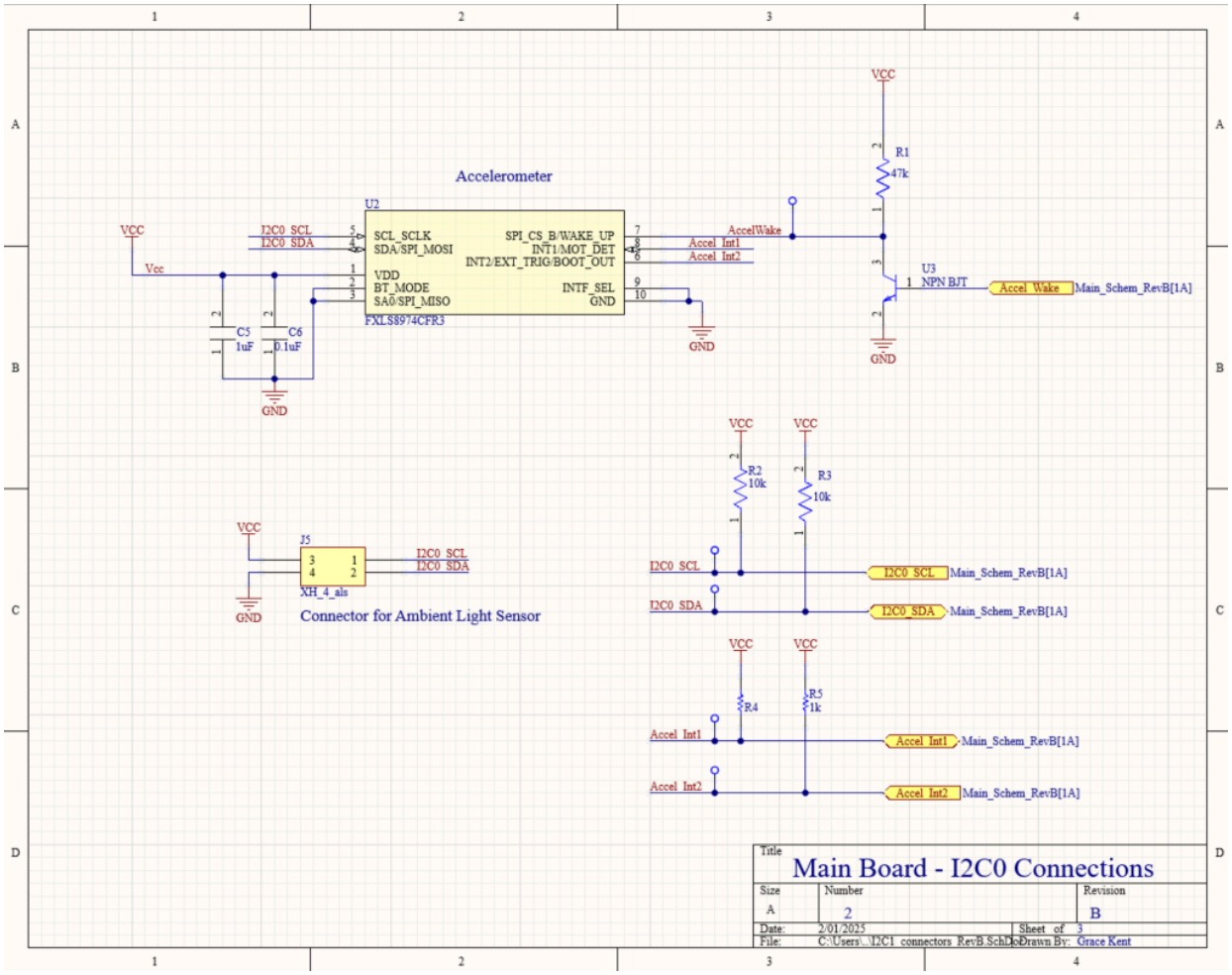


Figure 57: Connector Board Schematic, Sheet 2

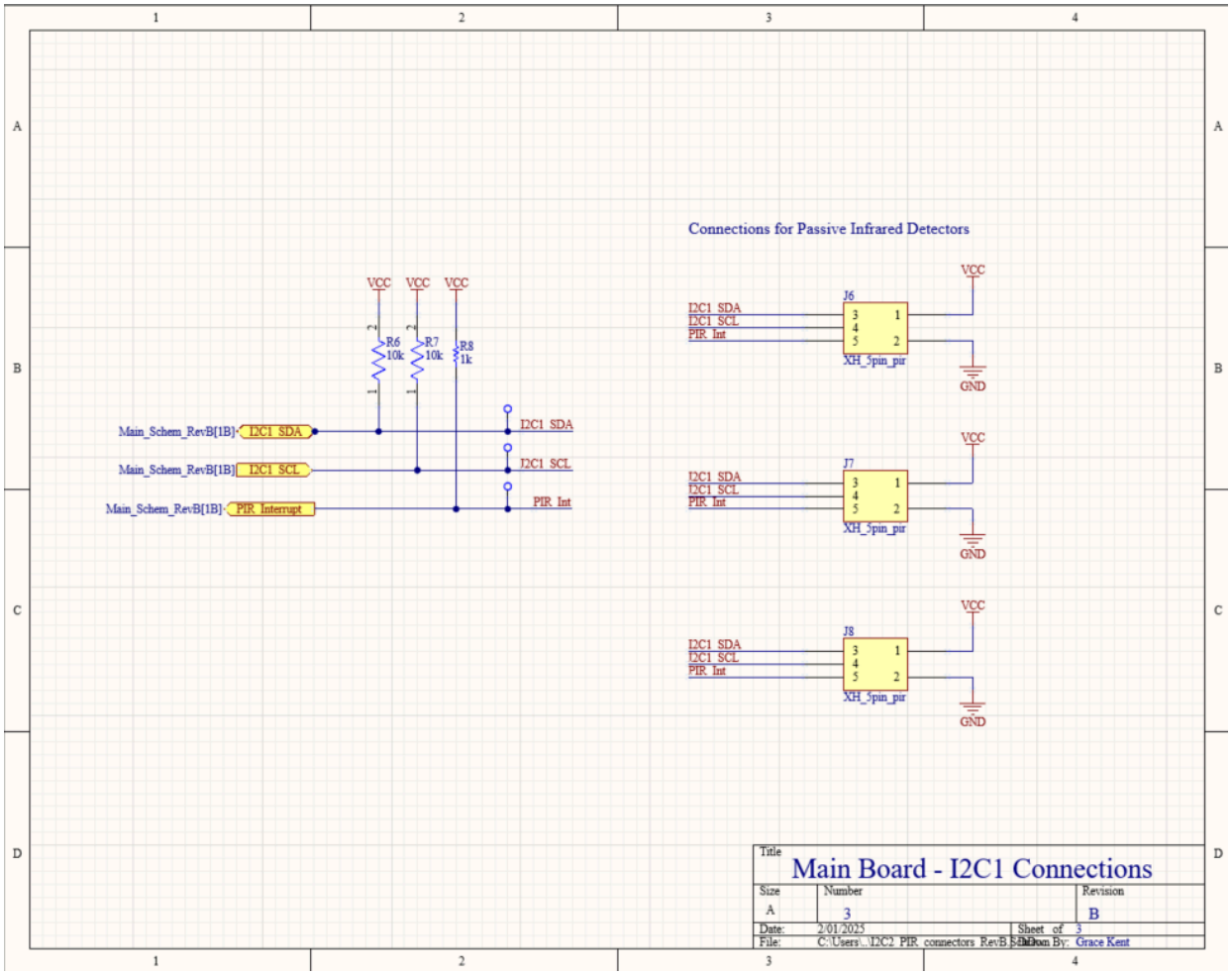


Figure 58: Connector Board Schematic, Sheet 3

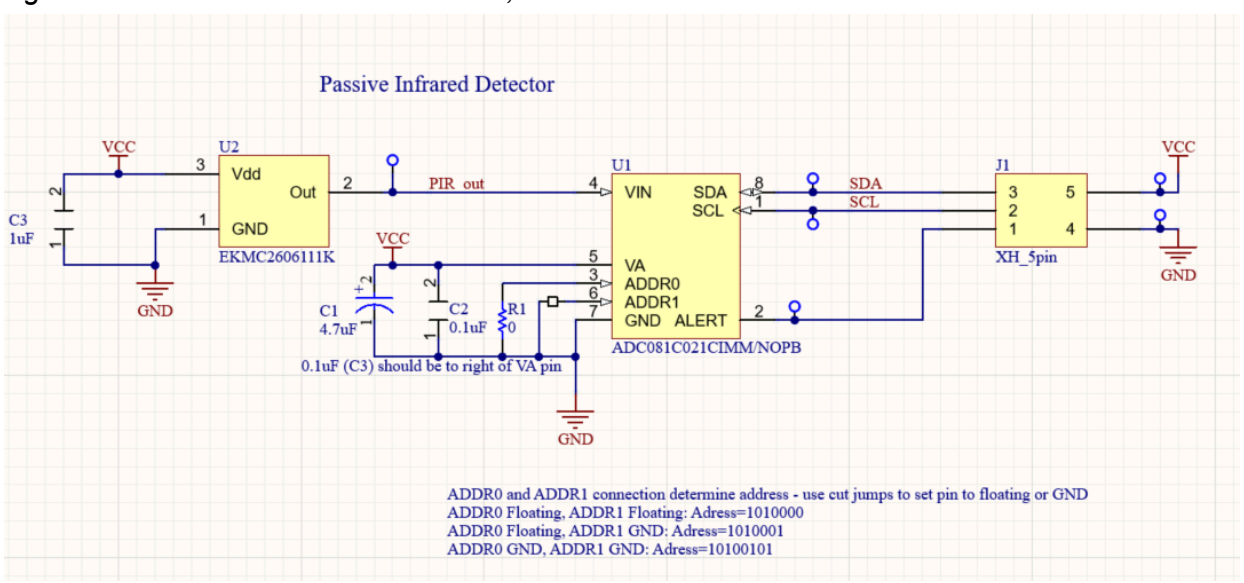


Figure 59: PIR Schematic

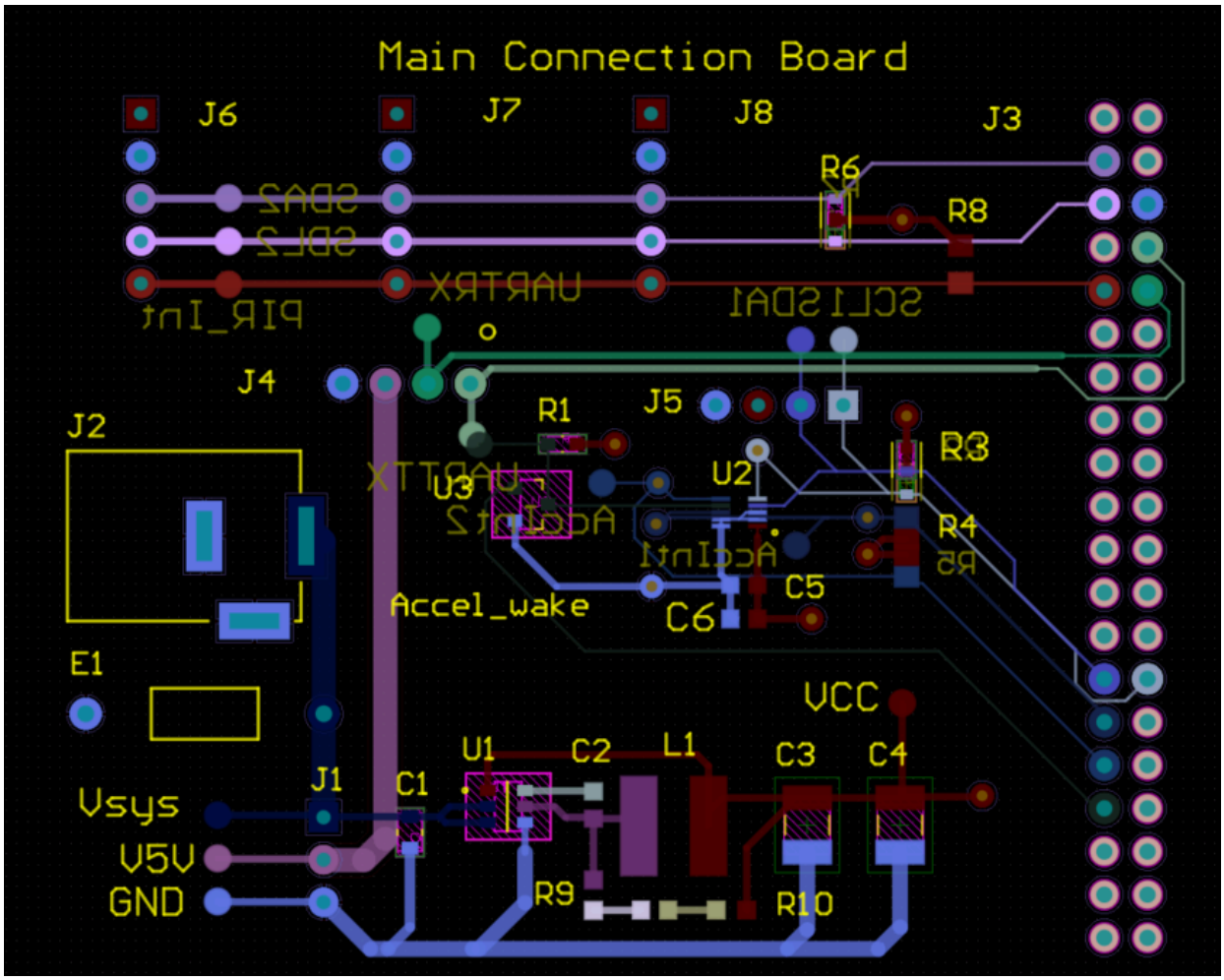


Figure 60: Connector Board PCB

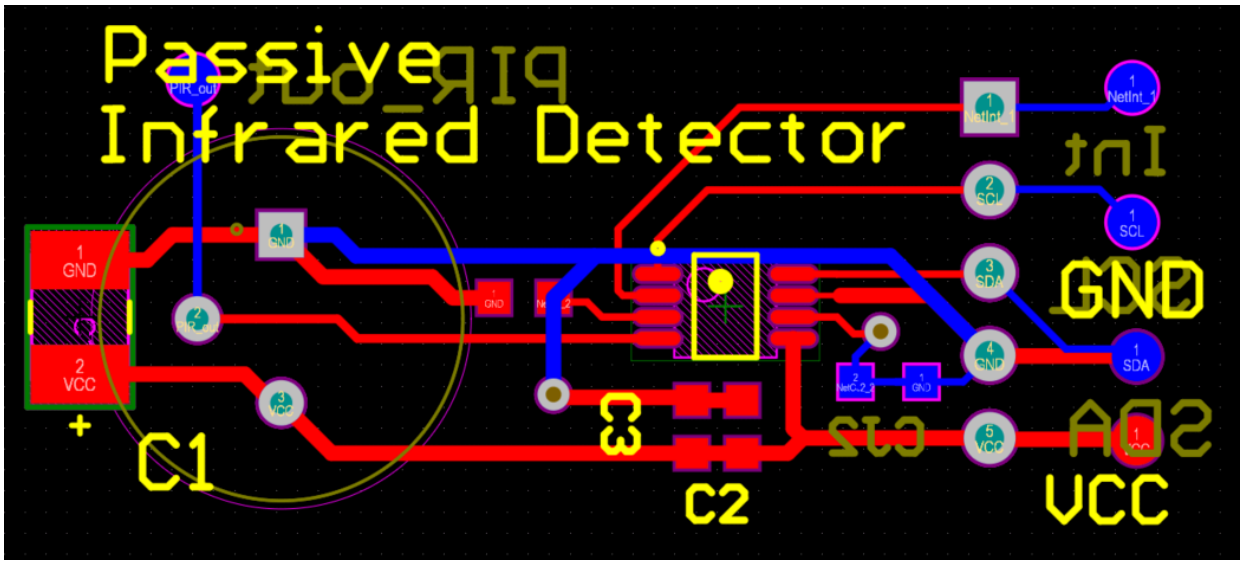


Figure 61: PIR PCB

2.5.2. Validation

Validation was done by reading all power levels on the interfacing boards, and seeing that Nvidia Jetson Orin Nano could communicate with the sensors.

2.6. Subsystem Conclusion

The hardware subsystem is validated, and will be easily integrated into the system. All sensors operate as intended. There are some discrepancies in performance as compared to the manufacturer's, but the thresholds are set in the program of the microcontroller, so the sensor performance can be compensated for. These sensors are necessary to reduce power consumption, detect tampering, and allow for motion detection.

More work should be conducted to calibrate the sensor thresholds. For instance a dynamic threshold may be useful for the PIRs, when different environmental conditions are present.

Integrated System

1. Overview

The Property Line Surveillance Systems allows for recording of one's property within a definable area. The system consists of a Sensors, Microcontroller, Pan Motion, Facial Recognition, and PC App subsystems. These substems were integrated together to form a completed product. Validation ensued to ensure that the subsystems interacted properly.

2. Design Plan and Execution

2.1. Design Plan

To integrate the systems the first step was the creation of the server. The server would allow the PC app to communicate with the microcontroller, allowing for important data to be accessed by the user. The server was simultaneously integrated into the app and the microcontroller. The microcontroller also needed to be able to communicate with each sensor and the camera. Furthermore, communication between the control of the pan motion, live blocking of property lines, and facial recognition processing of the recorded videos needed to occur.

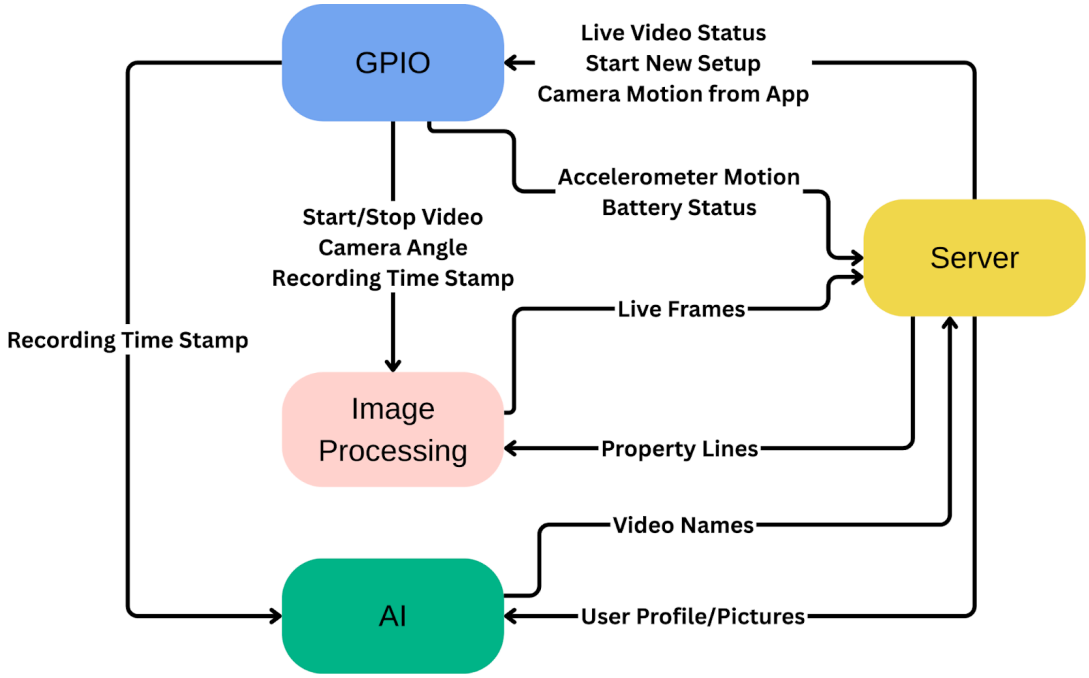


Figure 1: I/O Diagram

The code on the microcontroller (Jetson) was separated into four categories: server, GPIO, image processing, and AI. Due to live blocking being needed, and requiring a great deal of computational power, the image processing was separated into its own category. Each category is largely independent of each other, but there is a great deal of communication between them. However, each category sends and receives data from the server. The server is used as the interface to control the system and retrieve data.

The PC application used ‘get’ and ‘post’ requests to send the status updates of the user inputs in the PC app to the server and retrieve information from the server that was posted by the Jetson. The PC application also used a timer to check every 2 seconds if the panorama was done being recorded, and then load the panorama into the application, as well as checking every 10 seconds to see if there was a change of status in the server for the battery and accelerometer sensor statuses.

The microcontroller code is started using a bash script. This bash script starts the main code and a docker in which the Yolo script is run. The Yolo code is separated out so that its dependencies do not interfere with that of the main code, specifically the model. The docker allows human detection using Yolo to be run on the GPU, which significantly reduces processing time compared to using the CPU.

When the main code is run multiple processes are started using multiprocessing and threading. This was needed to ensure non-blocking operation, allowing separate pieces of code to function at the same time. Multiprocessing specifically allows the processes to be spread across multiple

cores (threading is done using one core), which increases processing speed since one core is not overloaded. In the final revision of the code there are seven multiprocessing processes: serial, hardware manager, image capture, image processing, recording logic, web server, and auto delete. Furthermore, facial recognition is run as a thread, there were issues when this was run using multiprocessing. Each process/thread is an infinite loop to guarantee operation.

The serial worker is used to communicate with the motor controller for pan motion and angle. A command queue is set up so that any concurrent commands sent can be stored and not forgotten. The response from the motor controller is then placed into a response queue, allowing each accessing task to get its correct response. These queues are shared with pano setup, hardware manager, image processing, and the server. Without this method the wrong response was being seen, which was detrimental to image processing. After a motion command the motor controller response is “0”, and since motion and image processing occur simultaneously the image processing was consistently reading a 0 for angle, causing misaligned blocking.

The hardware manager is used to read all of the sensor values and report them to the other processes. The accelerometer, pirs, and battery status are all monitored. The accelerometer and PIRs are only read if the tampering flag is false. If motion is allowed (value in the configuration/app settings), then a command may be sent to the serial process to move based on the PIR responses. Furthermore, there is a queue for the accelerometer basis to be taken. When a pano is taken a command is placed in the queue, and the accelerometer basis is taken. Once the accelerometer basis is taken the tampering flag is set to false. Due to irregularities with GPIO inputs to the Jetson, the battery status line does not read at zero when the USB-C is not plugged in. A buffer stage was added to try to keep this voltage line pulled low, but it still reads at a non-zero value, so the reading in the code would fluctuate between a 1 and 0. However, when USB-C is plugged in the reading is consistently high/1. So a list of the last twenty pin readings is monitored, if there is a zero in the list it can be concluded that there is only battery power. Otherwise, USB-C is plugged in.

The image capture process is separated so that frame capture is constant. It was noticed that starting the image pipeline each time pictures (for pano) or a video is taken caused inconsistencies with the frame buffer. This inconsistency led to changes in blocking as the frame could have been taken at an unknowable number of seconds after the angle was read. Furthermore, there is no lag when video is started since the pipeline has already been created.

The image processing is separate from capture as it should not run constantly. The image processing should only occur when recording. A read angle command is sent to the serial process, then a frame is taken from the image capture. Based on the angle the property lines are selected and then applied to the frame. The frame is then written to the video, whose time stamp is determined in recording logic. The image processing starts and stops based on the image processing event variable, which is set by the recording logic process. When a video is stopped the name of the video is put in a queue for the facial recognition process.

Recording logic is put into its own process so that it can constantly monitor multiple different inputs, while also not interfering with the processing time of the image processing loop. Recording logic monitors the gpios and values from the server. When tampering is true or the camera is turned off via the app, then the loop should continuously skip over the logic and set the image processing event to false. When recording is allowed the loop will check for object motion based on the PIRs and live streaming. At the start of a video motion/live streaming is true, but the image processing event variable has not been set to true. A name is then set for the video and image processing event is set. Video should be recorded until video is not allowed, or there is no live streaming and it has been ten seconds after the last motion detection. At that point the image processing event is cleared.

The AI/facial recognition process is continuously waiting for a video name to be placed in its queue. A queue is needed so that if a video is finished being recorded while a previous video is processing, it will be processed as well.

The auto delete process is used for deleting stored videos. Every 24 hours auto delete will run, deleting any videos that are past the video keep time.

The server process is what connects the microcontroller to the app. Based on impetus from the app events can be triggered, and the configuration changed. When any setting/status is changed in the app, it is then updated in the Jetson. If live streaming occurs, then the video will start (if allowed), and the frames will be sent through the server. When property line set up is clicked, then the pano generation will occur, and the accelerometer basis will be taken again. Once property lines are updated the new ones will be used for subsequent videos. When motion is allowed, the app motion command will send a command to the serial process through its queue. This process achieves all functionalities that are needed for the application features.

3. Validation

2.4.2. Drawing Property Lines

To validate property line drawing the “Property Line Setup” button was pressed and a new property line was drawn and confirmed to be applied to the live video. The live video and recorded videos were then checked to ensure the property lines functioned as expected.

Figures 7 to 11 show the process of drawing a polygon and figures 11 to 13 show the live and recorded videos to validate that the property lines were property applied.

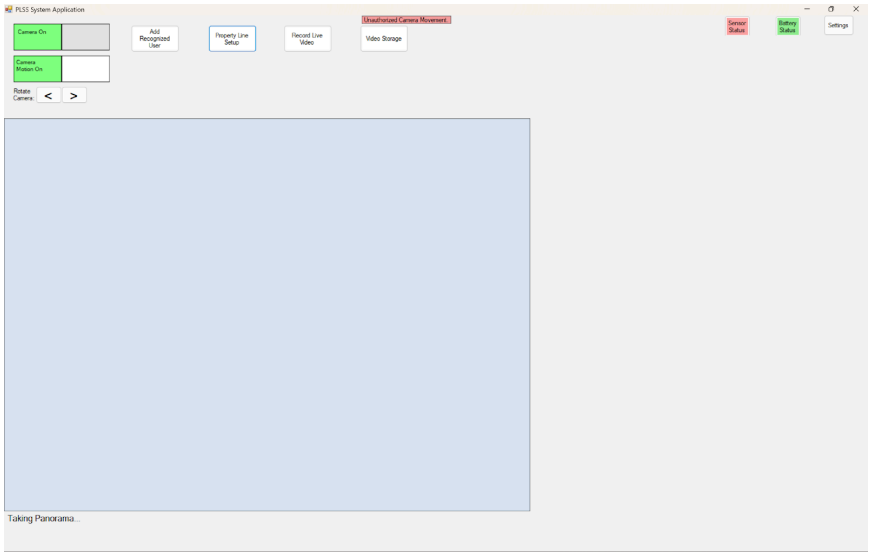


Figure 7: Property Line Drawing task while panorama is being recorded

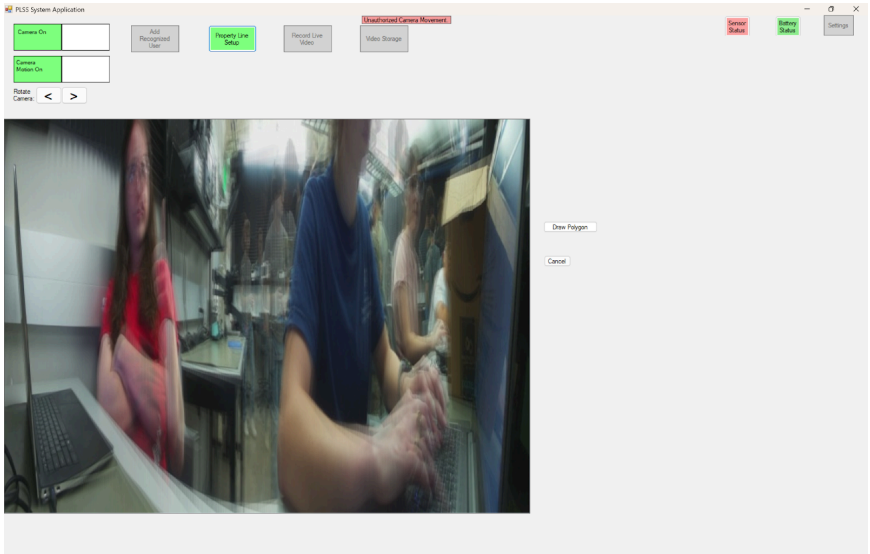


Figure 8: Property Line Drawing task with new panorama loaded into app

The downloading photos from the server into the PC application portion of the task for the FSR paragraph 3.2.1.3 is validated by the panorama photo taken by the camera and uploaded to the server then being downloaded and shown in the application in figure 8.

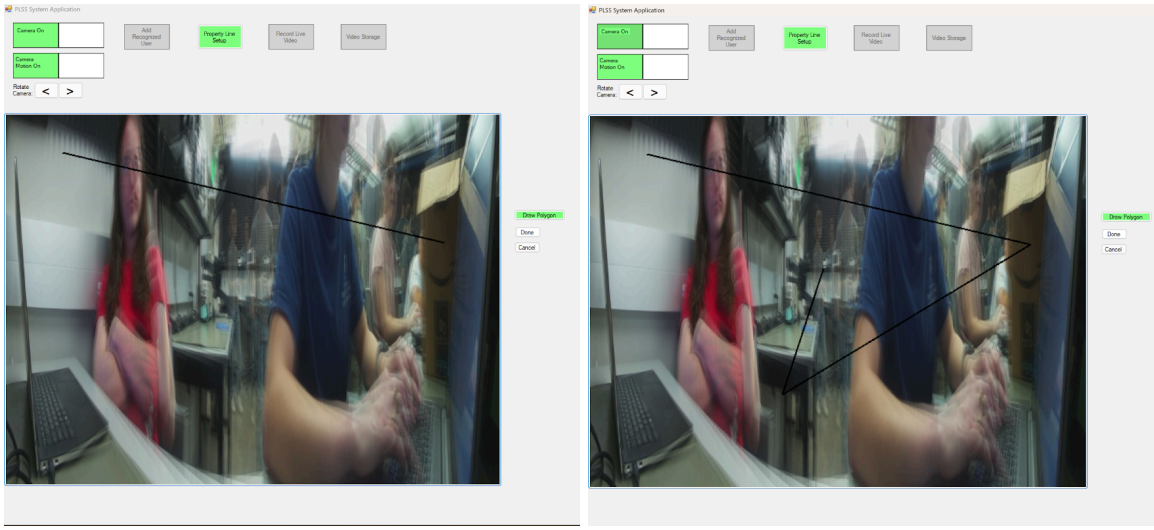


Figure 9: Draw Polygon button pressed and the user is drawing lines on the panorama.

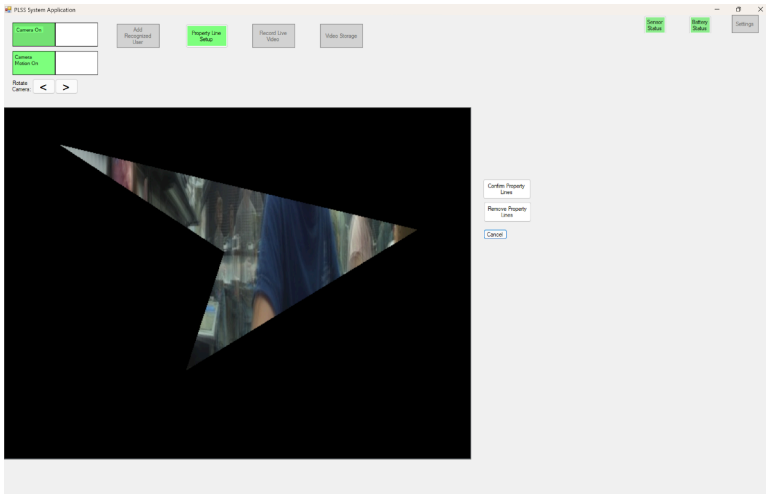


Figure 10: Done button pressed in figure 6 and a black mask applied outside of polygon.

The Remove Property Lines button was pressed and the panorama returned to the blank panorama in figure 8.

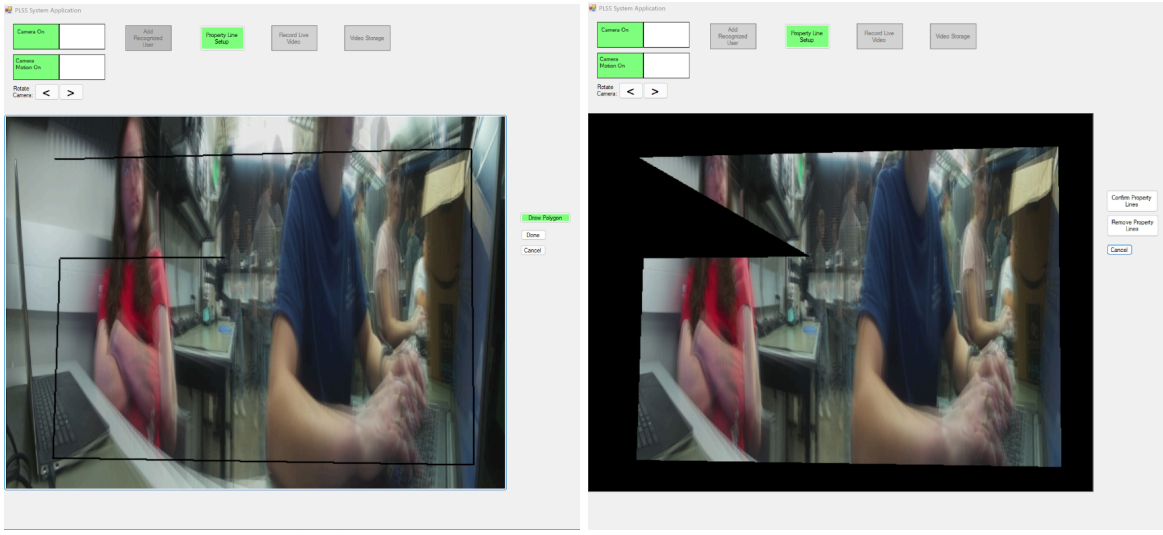


Figure 11: New property lines are drawn after pressing "Remove Property Lines"

The Confirm Property Lines button is pressed to apply the new property lines to the live video.

2.4.3. Live Video and Recorded Videos

The live video function was validated by pressing the Record Live Video button in the PC application and checking that the video feed loads into the application and is live.

Figure 12 (left) shows the live video with the new applied property lines from figure 11, and then shows the recording of that live video on the right of figure 12 after it has been processed by the facial recognition model.

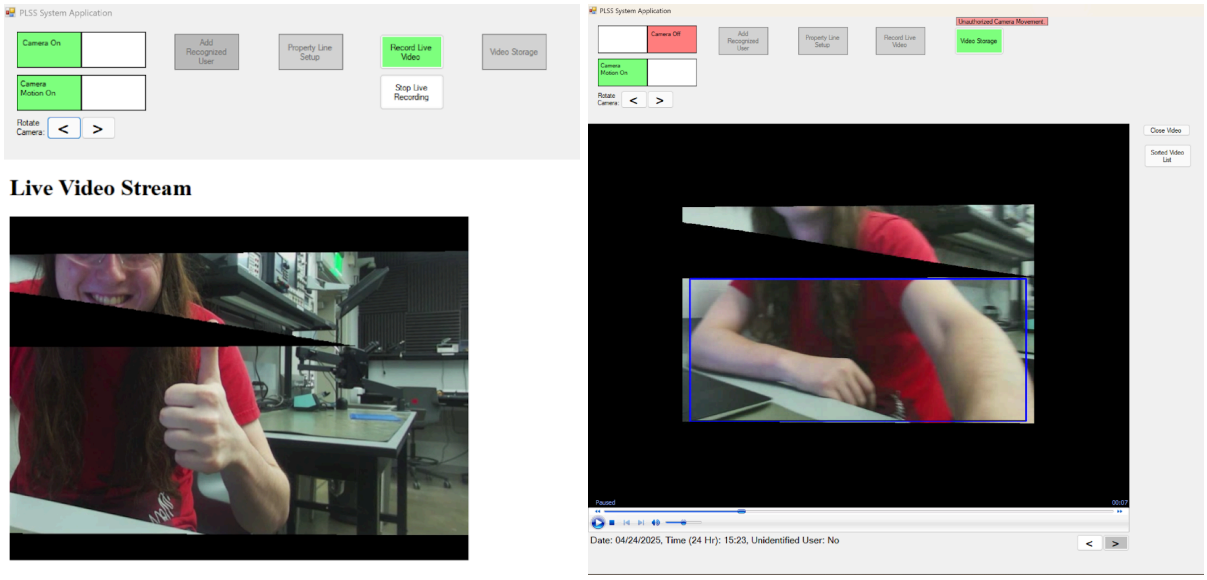


Figure 12: A live camera view with new applied property lines is shown (left), and the recording of that live video is then shown (right) in the video storage section.

The downloading/viewing of videos from the server to the PC application portion of the task for the FSR paragraph 3.2.1.3 is validated by the live video loading into the application for live video viewing from the server and the recorded video then being accessible from the server and loaded into the recorded video window in the application.

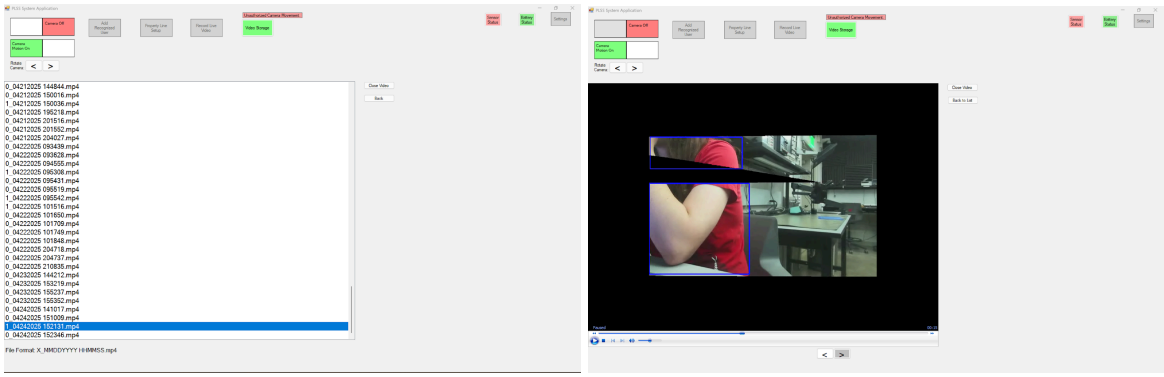
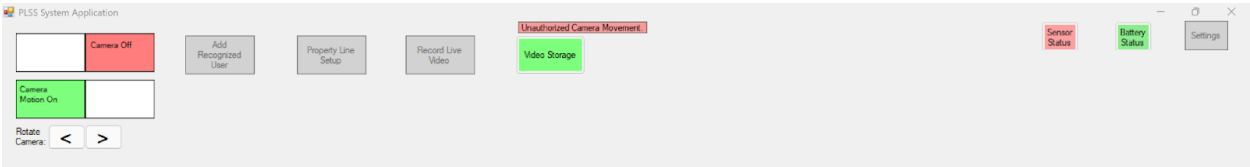


Figure 13: Shows list of record videos

The list of recorded videos was validated by clicking on the latest video and ensuring that the date, time, and video content are the correct corresponding video to the one that was clicked.

2.4.4. Unauthorized Movement Notification

To validate the accelerometer sensor status (unauthorized movement) notification the camera was shaken so the camera system would turn off the live video and the new status would be loaded into the application. To ensure this happens the application would show the battery status notification at the top of the application and the battery status button would be red when the server status of the accelerometer was changed to show there was unauthorized motion.



2.4.8. Video Blocking Based on Property Lines

Figure 18 shows a panorama produced by the system with the user mask applied. It also demonstrates how this mask is applied to the live frame from different camera angles.

Figure 18: Top image shows pano with user mask applied. Bottom images show subsequent blocking

2.4.9. App Movement

Using the buttons in the app, the camera system will move right or left. There are bounds placed on the movement range so that it does not exceed bound based on angles the panorama was taken at.

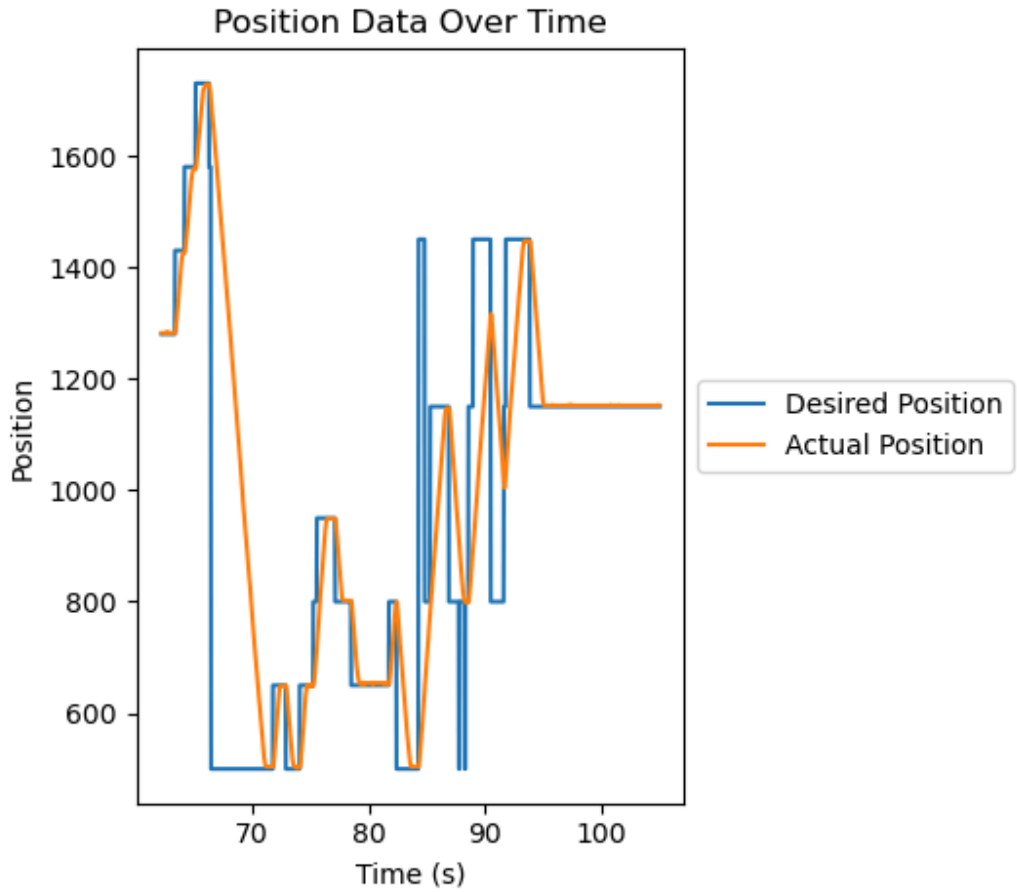


Figure 19: Position Plot with Clear App Movement

In Fig. 18 the app movement can be seen when steps occur. One button click in the app corresponds to a 150 change in position (180° is 2048).

2.4.10. Configuration - Movement On/Off

Movement of the camera system can be turned on/off in the app. To validate this the PIR response, position based on PIR response, and the actual position should be monitored.

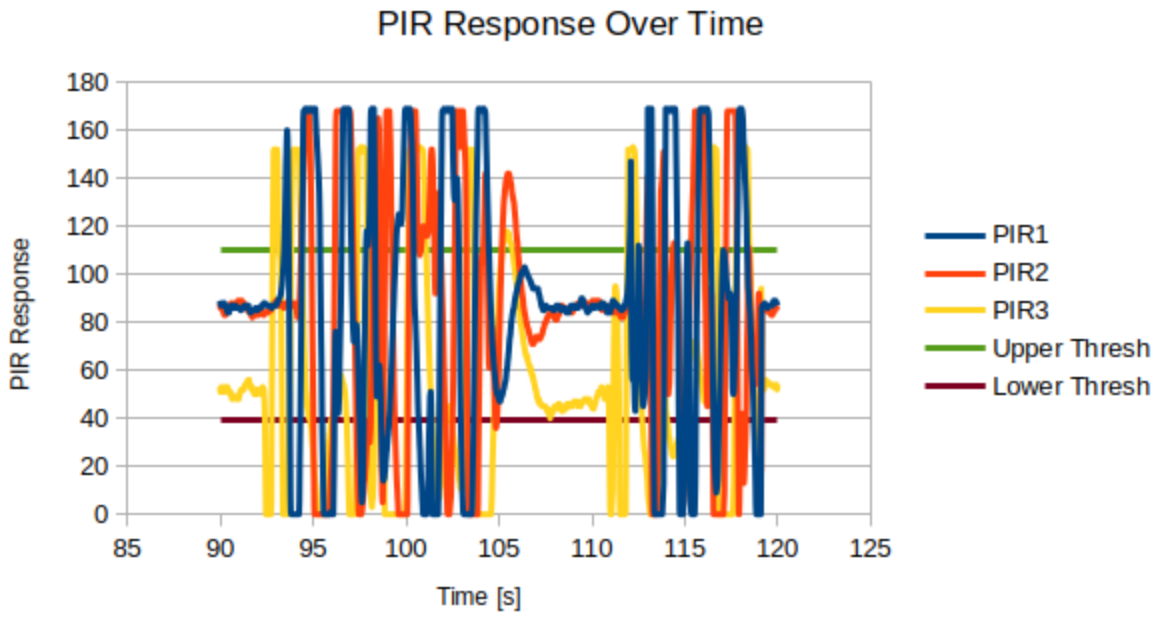


Figure 20: PIR Response Over Time for Movement On/Off Validation

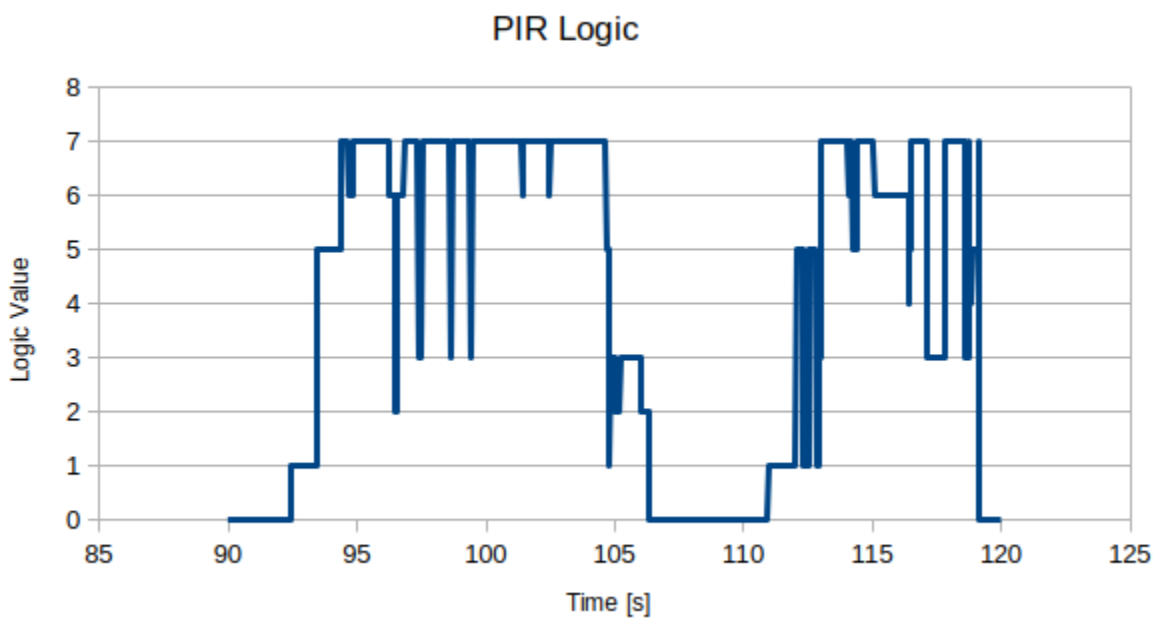


Figure 21: PIR Logic Over Time for Movement On/Off Validation

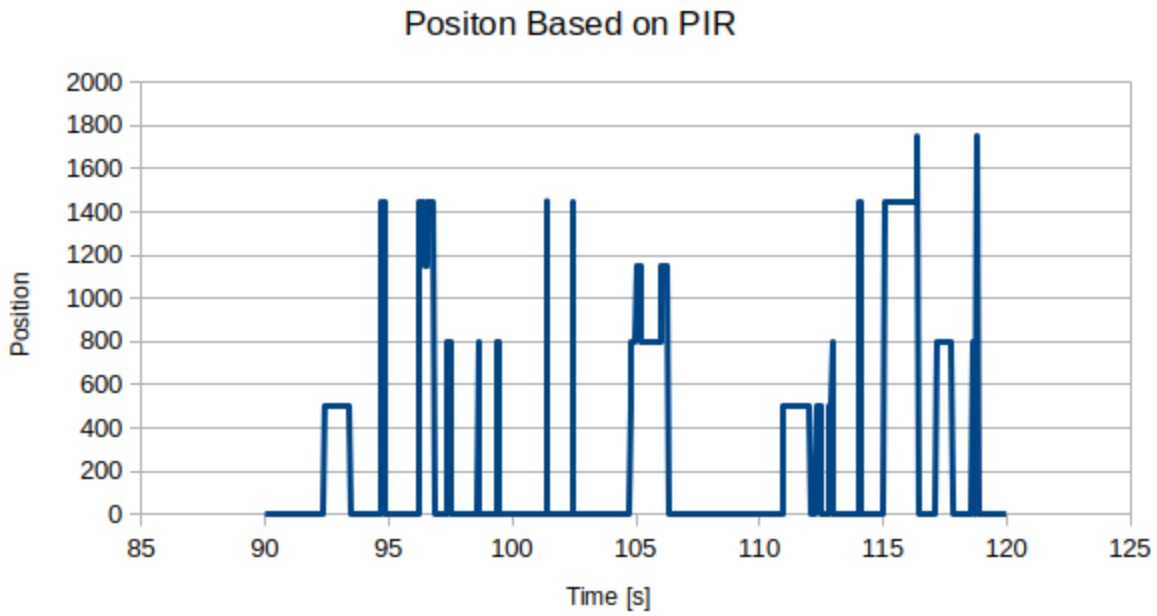


Figure 22. Desired Position Based on PIR Response Over Time for Movement On/Off Validation

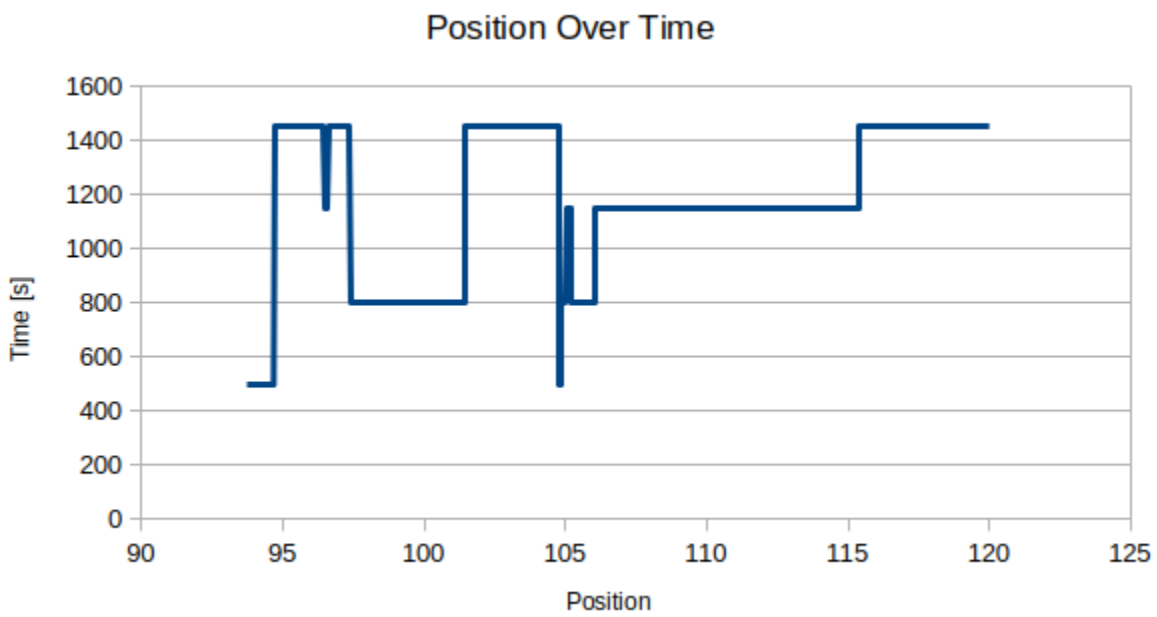


Figure 23. Actual Position Over Time for Movement On/Off Validation

The figures above illustrate motion being on, and the response of the system as the PIRs detect motion. The motion is then turned off, so while the PIRs do detect motion, the actual position never changes. Once motion is allowed again the camera system once again moves according to the PIR response.

2.5.4. Auto Delete

Auto delete runs every 24 hours. Videos are kept for 45 days automatically, but the user can choose the video storage time. The following table shows a condensed version of the auto delete log.

Table 8: Condensed Auto Delete Log

Last Keep Date	Video Date	Video Name	Delete Status
2025-03-17	2025-02-23	0_02232025 125943.mp4	Delete
2025-03-17	2025-03-23	1_03232025 180452.mp4	Keep
2025-03-17	2025-03-28	0_03282025 150313.mp4	Keep
2025-03-17	2025-04-01	0_04012025 160255.mp4	Keep
2025-03-17	2025-04-08	1_04082025 164108.mp4	Keep
2025-03-17	2025-04-16	1_04162025 145906.mp4	Keep
2025-03-17	2025-04-25	0_04252025 131148.mp4	Keep

2.5.5. PIR Response and Position

Based on the value of the PIRs, a position is chosen. The correct operation of this needs to be monitored.

Table 9: PIR Flags, Logic Value, and Corresponding Position

PIR 1	PIR 2	PIR 3	PIR Logic Value	Position
0	0	0	0	0/Do Nothing
0	0	1	1	500
0	1	0	2	1150
0	1	1	3	800
1	0	0	4	1750

1	0	1	5	0/Do Nothing
1	1	0	6	1450
1	1	1	7	0/Do Nothing

To monitor the response of each, movement at certain distances can be recorded and the resulting data.

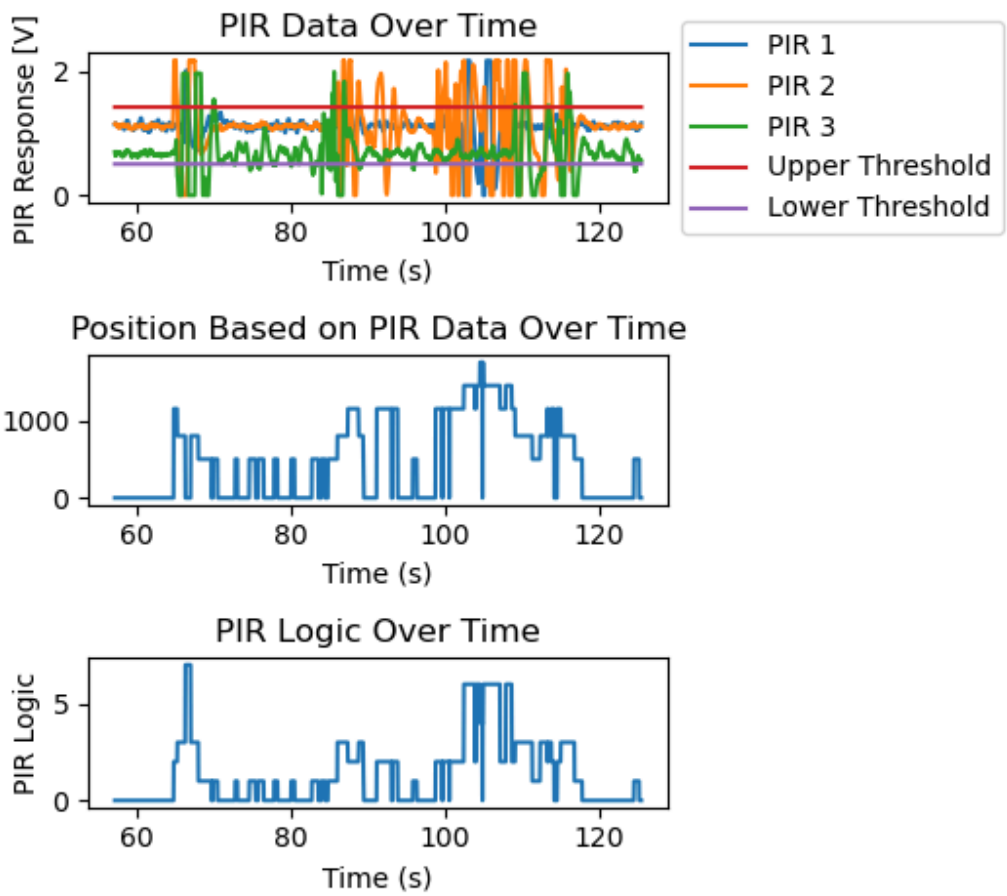


Figure 26: PIR Response, PIR Logic, and Position at 5ft

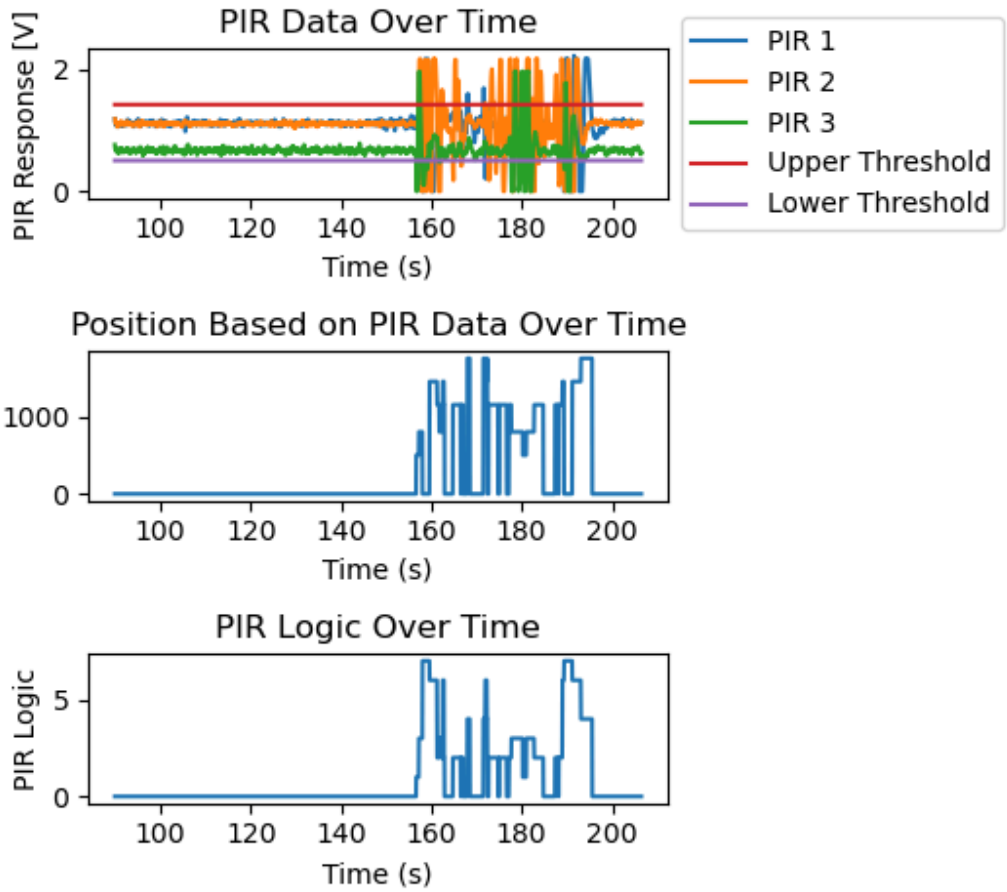


Figure 27: PIR Response, PIR Logic, and Position at 10ft

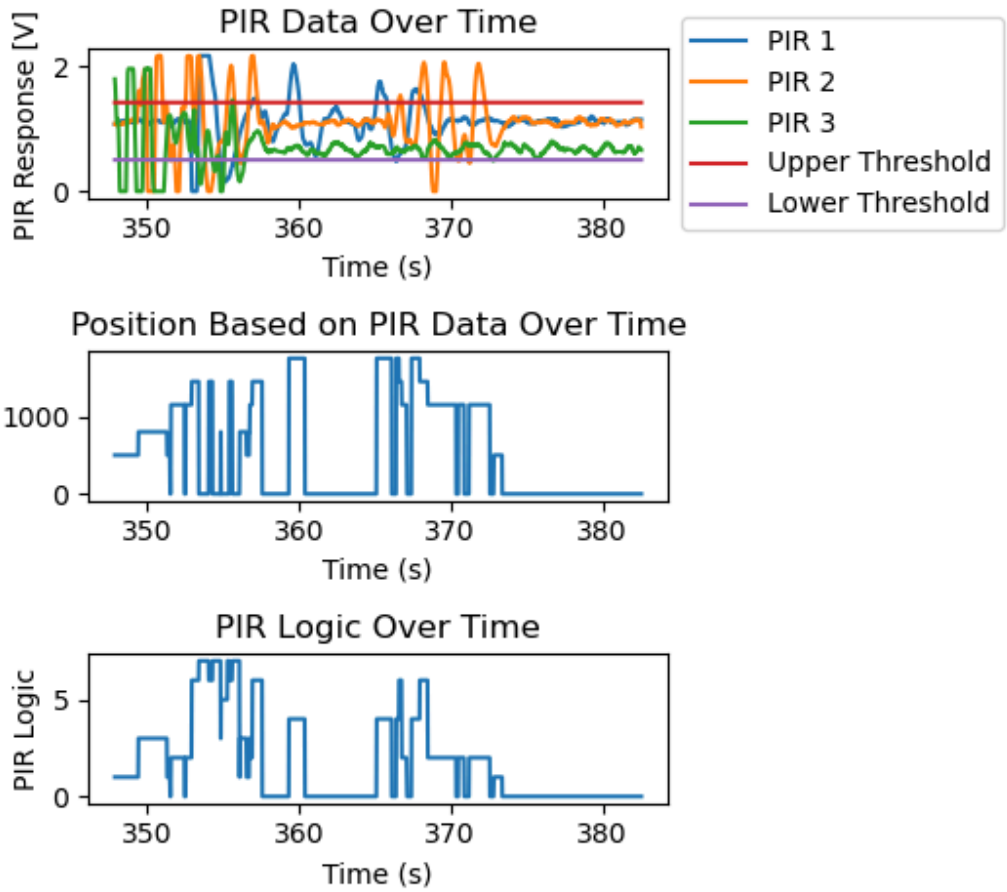


Figure 28: PIR Response, PIR Logic, and Position at 15ft

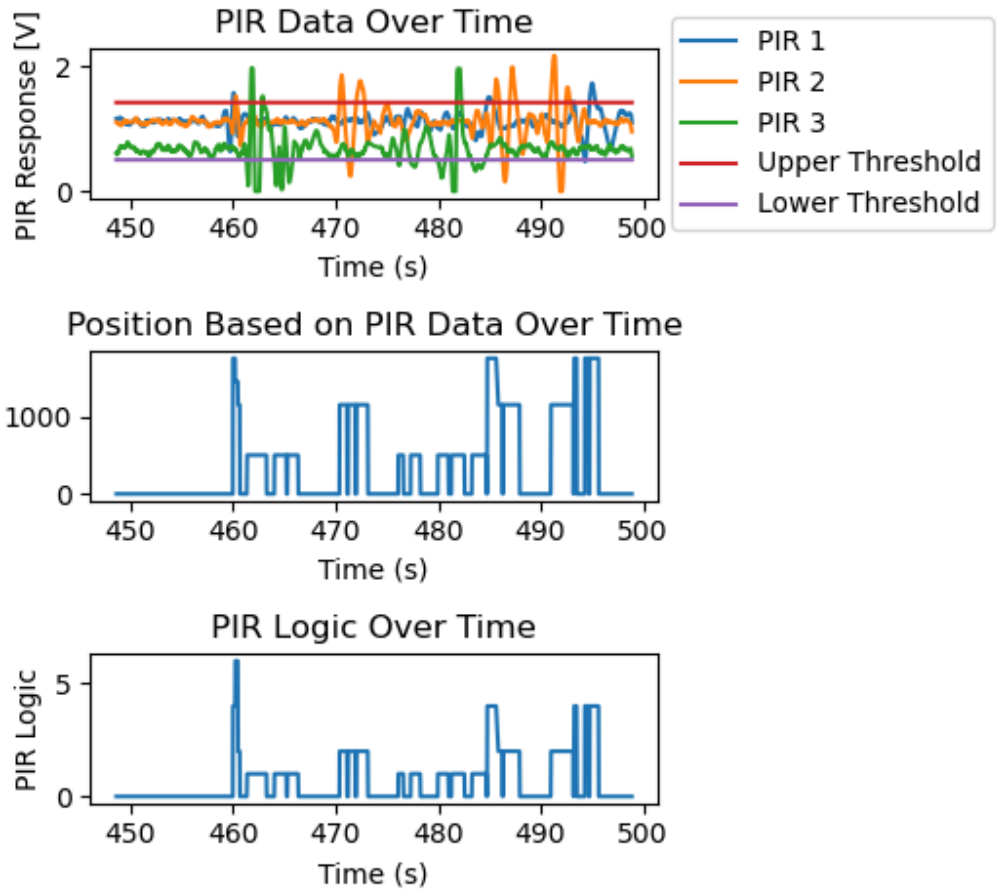


Figure 29: PIR Response, PIR Logic, and Position at 20ft

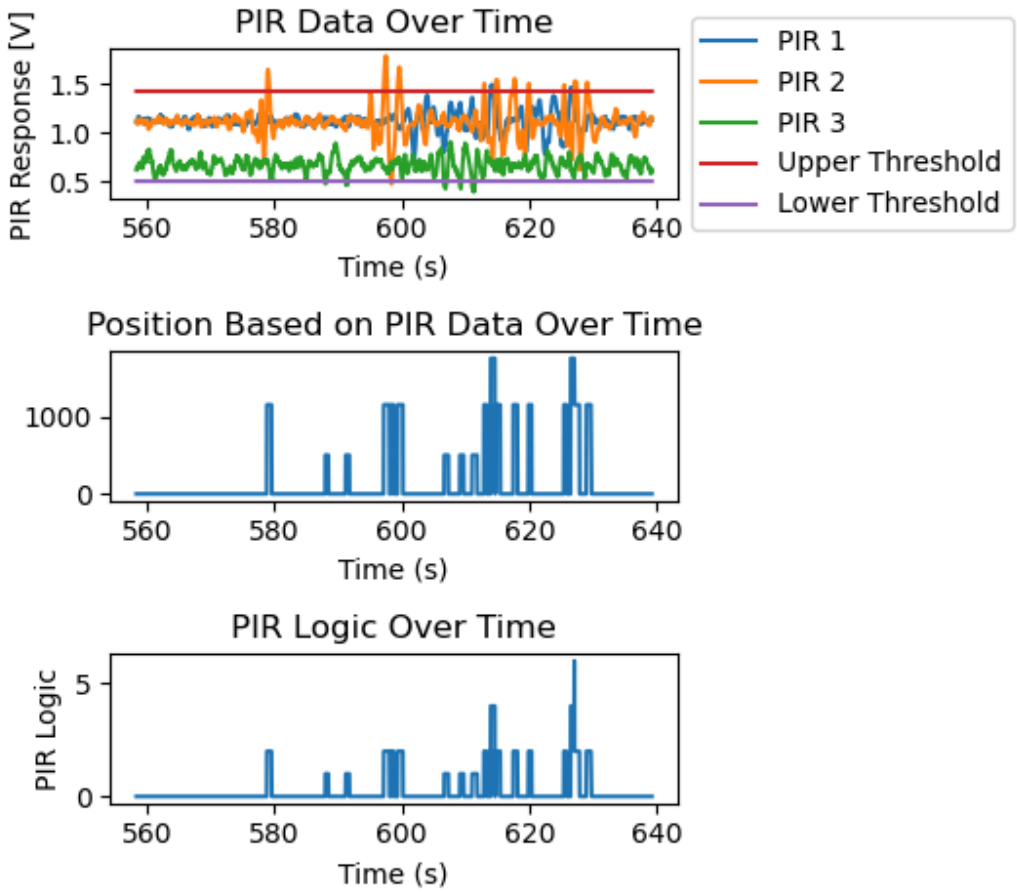


Figure 30: PIR Response, PIR Logic, and Position at 25ft

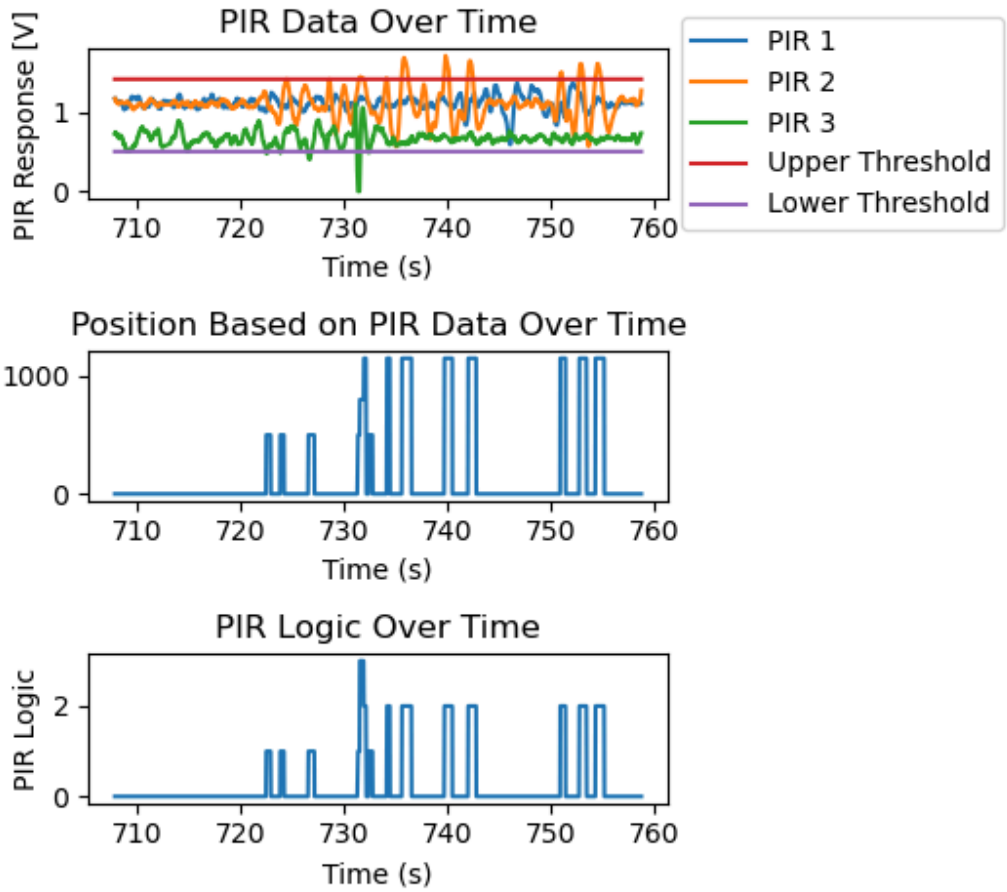


Figure 31: PIR Response, PIR Logic, and Position at 30ft

The correct motion based on PIR response is validated. While the operation is correct, it can be improved through refining of the logic, or using the camera to help track human motion. It has validated detection even during adverse conditions to 30ft. Detection as far as 60ft has been observed under more favorable conditions.

2.5.6. Recording Logic

A further step in validating the correct operation is by monitoring all of the sensor flags and the corresponding recording logic. This can be done for multiple use cases as well.

This is a condensed set of data, but the full tables can be seen in Table 14 in the appendix.

Recording should only be allowed when property lines are uploaded. For recording logic, if configuration status is true if the camera is on and there are correct property lines. PIR readings happen once the accelerometer/tampering flag is false, but video will not start until after configuration status is true. So for recording to occur configuration status must be true, accelerometer/tampering flag false, and then once either live streaming or motion flag is true video will start. The recording status will be “Not Allowed” if the accelerometer flag is true, and

will be “None” if recording is allowed, but there is no live streaming/motion. Video status is “Recording” when actual video occurs. “Start” will occur only at the very beginning of a recording.

Table 10: Live stream starts video, then ends 10 seconds after last motion

Time	Accel Flag	Configuration Flag	Motion Time	Motion Flag	Live Stream Flag	Recording Status
142.314736843109	False	False	0	False	False	Not Allowed
143.250711917877	False	True	0	False	False	None
144.21665096283	False	True	0	False	True	Start
145.171357154846	False	True	0	False	True	Recording
157.259848833084	False	True	1745698715.66032	True	True	Recording
185.35408616066	False	True	1745698743.05867	True	False	Recording
206.434041500092	False	True	1745698753.84117	False	False	None

The table above shows where the accelerometer basis/bounds is reset, then there is no motion. The video is then started by live stream, motion occurs during, and the video ends ten seconds after motion.

Table 11: Only Live Streaming

Time	Accel Flag	Configuration Flag	Motion Time	Motion Flag	Live Stream Flag	Recording Status

207.411942481995	False	True	1745698753.84117	False	True	Start
208.507237195969	False	True	1745698753.84117	False	True	Recording
270.775648593903	False	True	1745698753.84117	False	True	Recording
271.764537096024	False	True	1745698753.84117	False	False	None

This table displays a video where only live streaming happens. Video starts on live streaming, and ends when it ends.

Table 12: Video Start on Motion, Ends on Tampering

Time	Accel Flag	Configuration Flag	Motion Time	Motion Flag	Live Stream Flag	Recording Status
348.103960037231	False	True	1745698906.4421	True	False	Start
349.436425447464	False	True	1745698907.83913	True	False	Recording
381.175274848938	False	True	1745698931.33721	True	False	Recording
383.187419652939	True	False	1745698931.33721	False	False	Not Allowed

This table displays an example video where the video starts on object motion detection, and ends when tampering is detected.

Table 13: More Complex Recording Logic

Time	Accel Flag	Configuration Flag	Motion Time	Motion Flag	Live Stream Flag	Recording Status
448.420065641403	True	False	1745698931.33721	False	False	Not Allowed

450.22155046463	False	False	1745698931.33721	False	False	Not Allowed
461.405915737152	False	False	1745699019.77191	True	False	Not Allowed
470.542803049088	False	True	1745699028.79914	True	False	Start
471.752681255341	False	True	1745699030.07971	True	False	Recording
498.610296726227	False	True	1745699053.93118	True	False	Recording
501.467321872711	True	False	1745699053.93118	True	False	Not Allowed

This table displays the accel flag changing from true to false, and then the motion flag turns true. However, the configuration flag is false as property lines have not been uploaded yet so video is not allowed. Once the property lines have been uploaded the video starts. The video ends on tampering detection, even though the motion flag is still true.

2.5.7. Overall

The overall operation can be validated. This includes the recording log, PIR data, accelerometer data, and position data, all recorded for the same period of time. The recording log can be seen in *Table 14*.

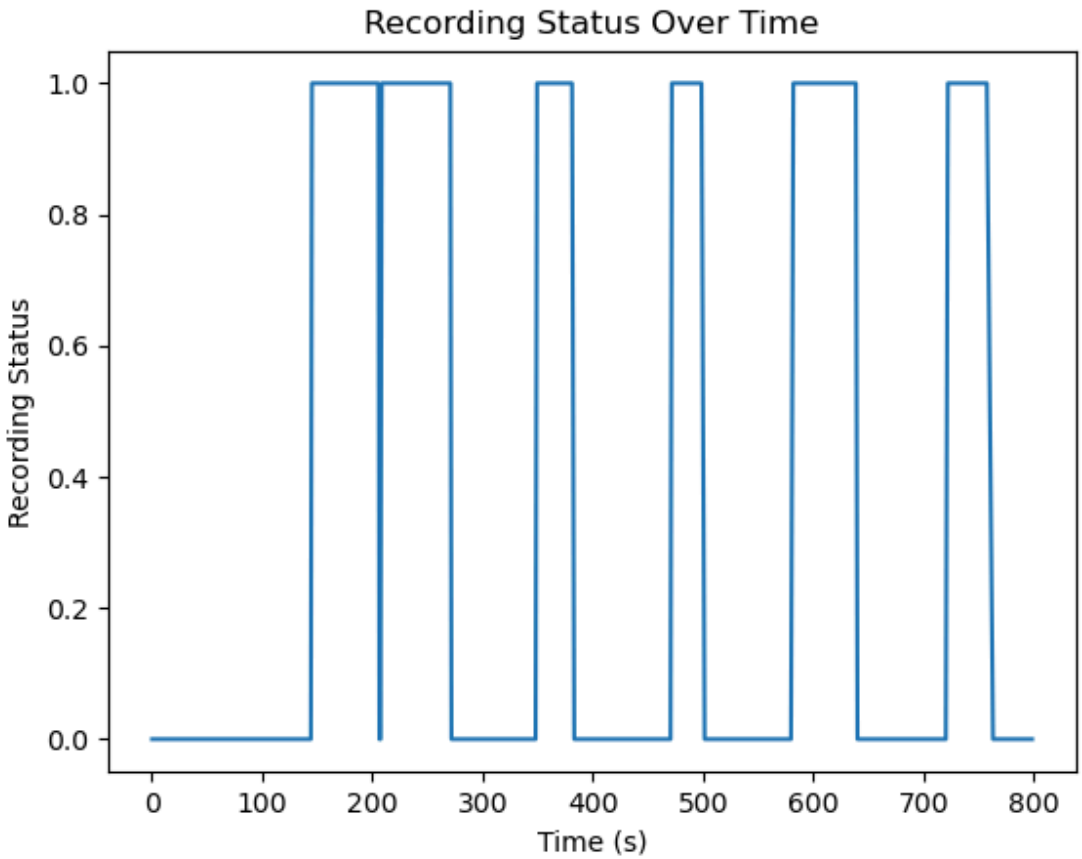


Figure 32: Overall Recording Status

The overall recording status is zero is not recording video, and one while recording.

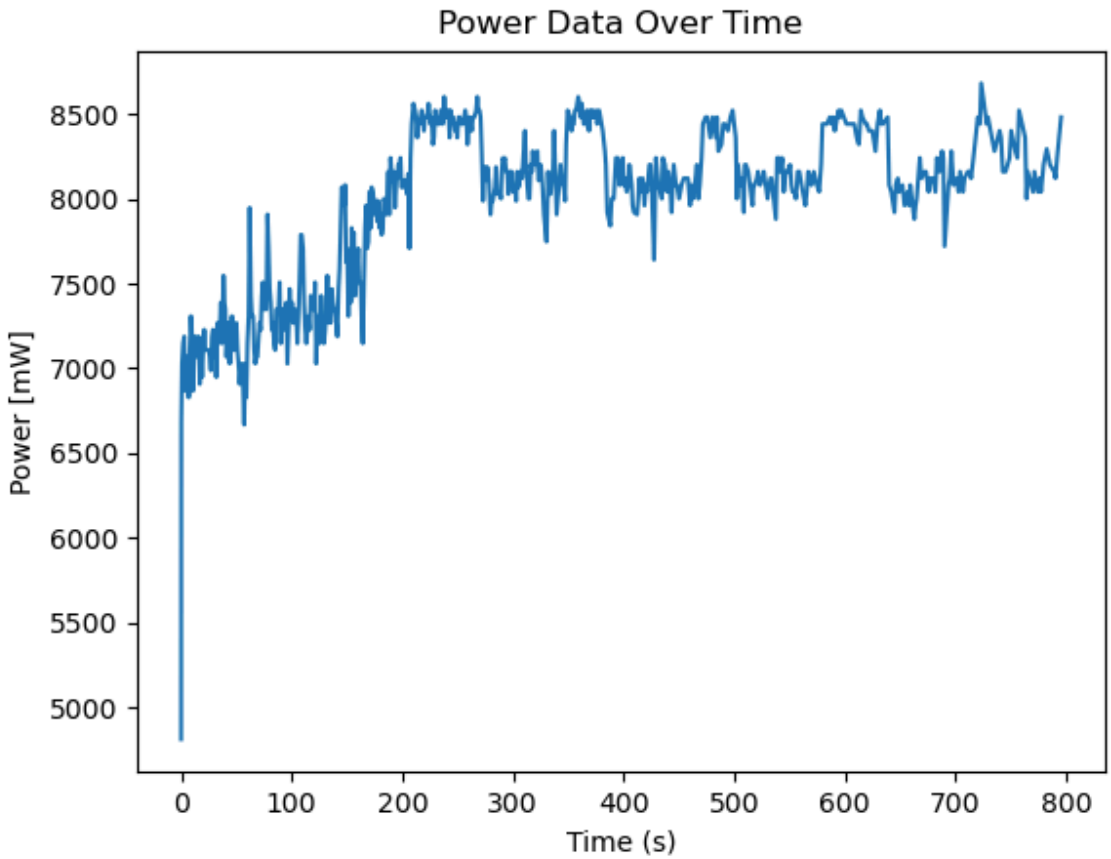


Figure 33. Overall Power Data over Time

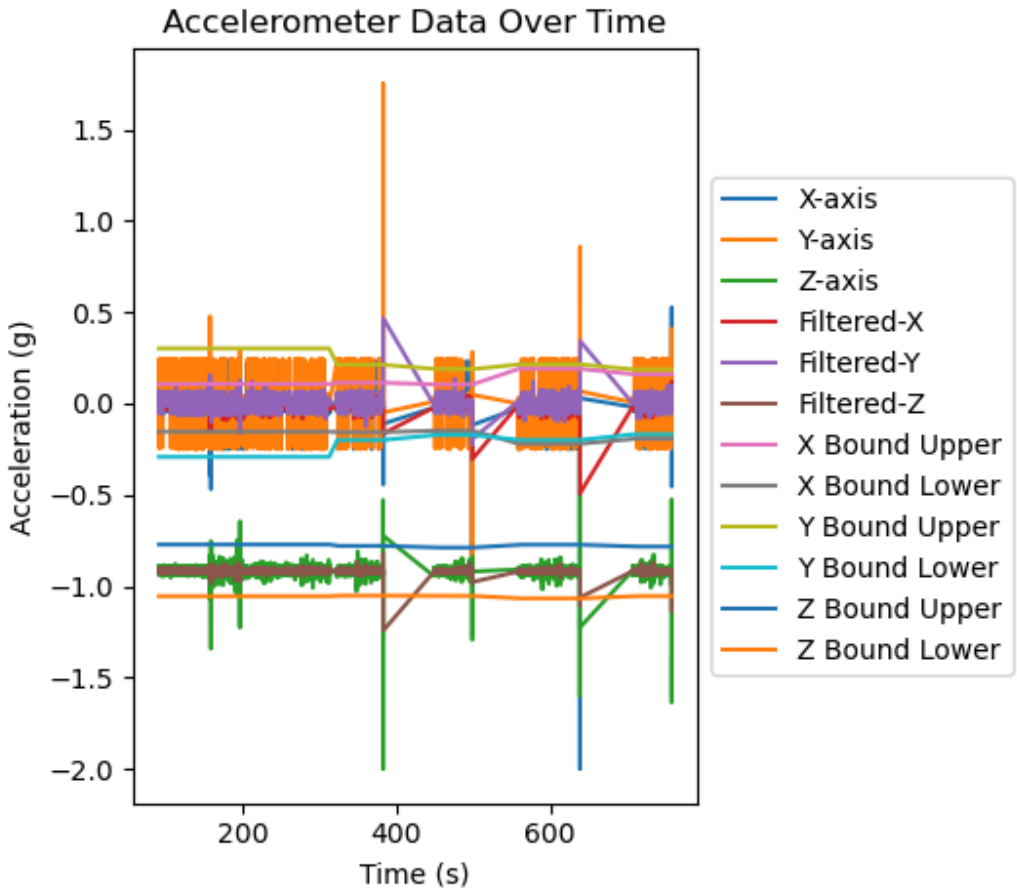


Figure 34: Overall Accelerometer Data Over Time

The accelerometer detects tampering four times. Each time there is a gap it is because a new pan/basis is being taken, or it is after tampering has been detected. Once a new basis has been taken data is collected about the accelerometer again.

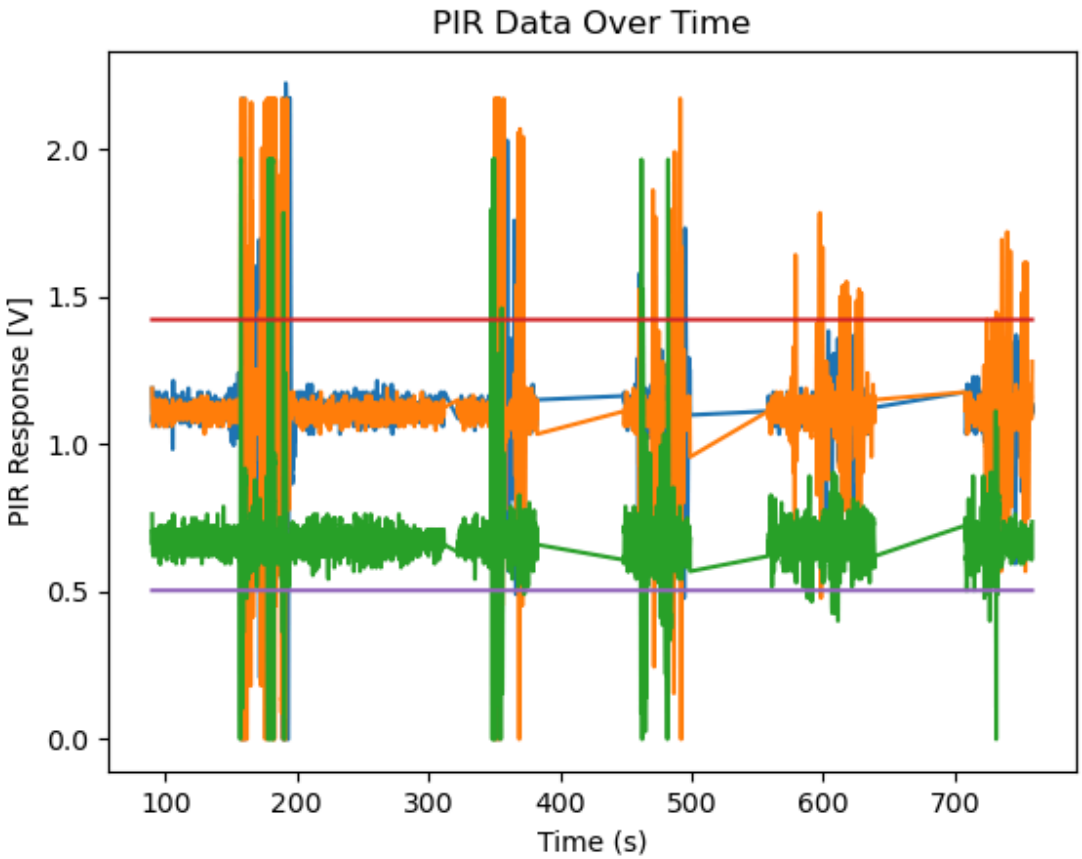


Figure 35: Overall PIR Data

The PIR response over time should be monitored so that it can be ensured that it accurately lines up with the recording logic.

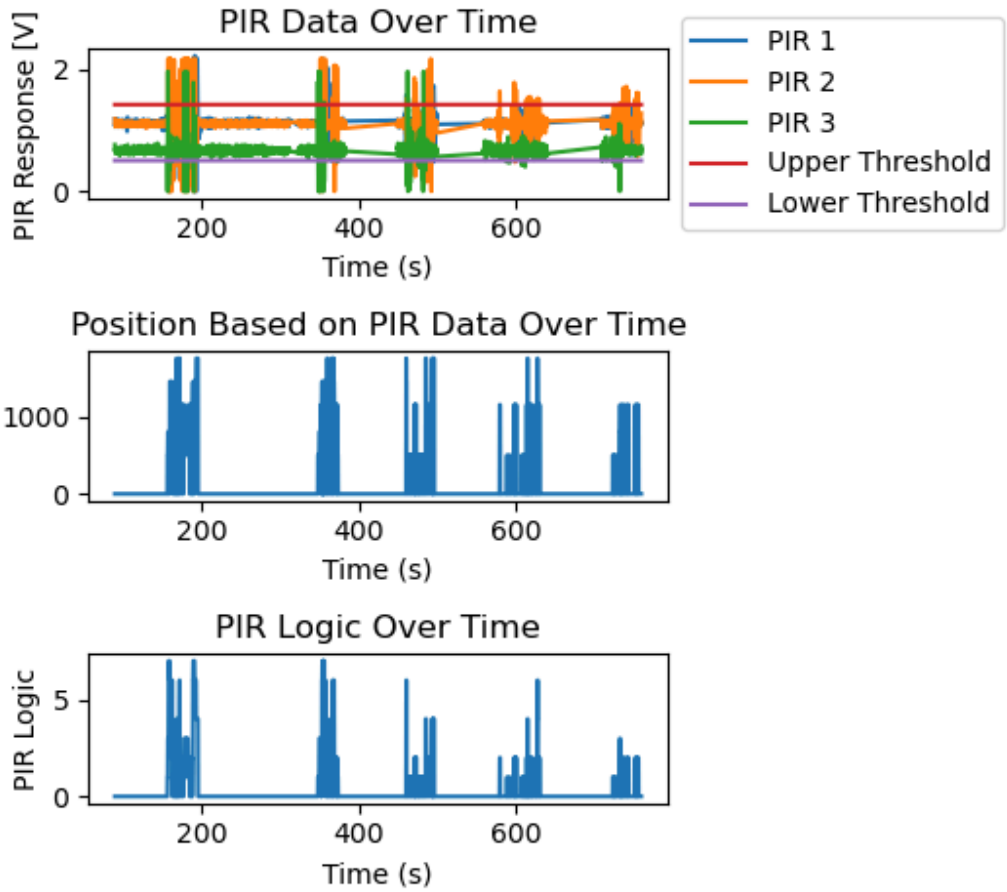


Figure 36: Overall PIR Response, Position, and Logic

The position based on PIR response is also important to compare to the actual position recorded.

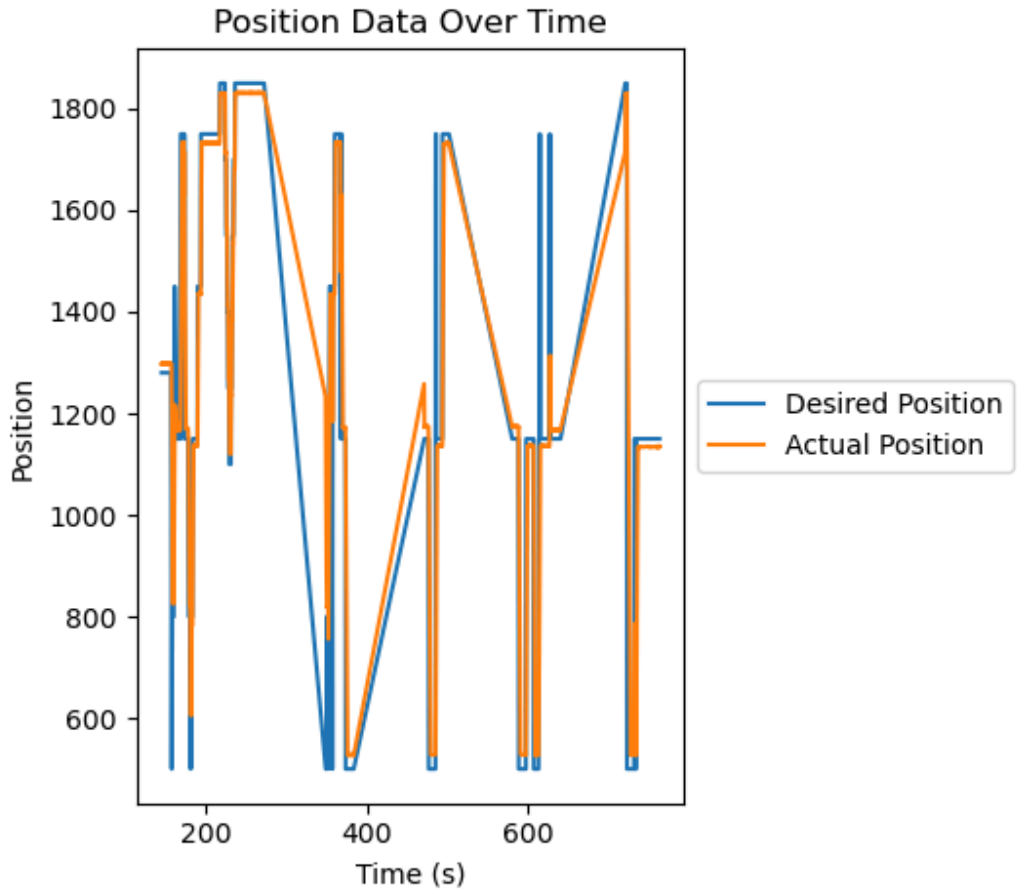


Figure 37: Overall Desired vs Actual Position

The position over time needs to be monitored as well. The actual position should try to track the actual position as much as possible.