A Low-Cost Camera-Lidar UAV Subsystem for Accessible Terrain-Surveying and Data Analytics

a project presented to
The Faculty of the Department of Aerospace Engineering
San José State University

in partial fulfillment of the requirements for the degree
Master of Science in Aerospace Engineering

by

Michael J. Korens

May 2026

approved by

Ali Guarneros Luna

Faculty Advisor



© 2026
Michael J. Korens
ALL RIGHTS RESERVED

ACKNOWLEDGEMENTS

I acknowledge the use of AI tools to assist with Overleaf/LaTeX formatting via ChatGPT and Grammarly, addressing grammatical, flow, and spelling errors. The content presented in this paper is human-verified and written. AI is only used as a sanity-check tool after the content has been hand-typed in its entirety.

Table Of Contents

Acknowledgements **ii**
List of Figures v
List of Tables vi

Symbols **1**

1 Introduction **2**
1.1 Motivation 2
1.2 Literature Review 3
1.2.1 Research Applications of UAVs 3
1.2.2 Research Applications of Lidars 5
1.2.3 Research Applications of UAVs with Camera-Lidar Sensors 6
1.3 Project Proposal 8
1.3.1 Project Goals 8
1.3.2 Requirements 9
1.4 Methodology 9

2 Hardware Consideration and Implementation **11**
2.1 Introduction 11
2.2 Camera 11
2.3 Lidar 14
2.4 Microcontroller 16
2.4.1 Arduino Uno Q 17
2.4.2 Raspberry Pi 5 18
2.5 Additional Considerations 19
2.5.1 Multispectral Camera 19
2.5.2 Thermal Imaging Camera 23
2.6 Battery System 24

3 Subsystem Housing and Integration **30**
3.1 Subsystem Housing 30
3.2 Electronic Integration 37

4 Software Environment **39**
4.1 Operating System and Installation 39
4.2 Accessing the Raspberry Pi 39
4.2.1 Micro-HDMI method 40
4.2.2 Raspberry Pi Connect - Desktop Streaming 41
4.2.3 Raspberry Pi Connect - Remote Shell 43
4.3 Camera-Lidar Software 44
4.3.1 GUI-based Software 44
4.3.2 Command Line Software 45
4.3.3 Data Retrieval 48

5 Subsystem Testing **54**
5.0.1 Camera Testing 54
5.0.2 Remote Access Testing 56
5.0.3 Lidar Testing 60
5.0.4 Battery Testing 61

6 Conclusion and Future Work **62**
6.1 Conclusion 62
6.2 Future Work 62

A Current Hardware Specifications **70**

B Comparison of Components **73**

C Bill of Materials **75**

List of Figures

1.1 Multi-camera and microcontroller UAV mounting system [19]. 7
1.2 Multi-camera, microprocessor, and lidar mounting configuration on a UAV drone [19]. 7

2.1 Raspberry pi high quality camera [21]. 13
2.2 Arducam owlsight 64mp camera with a prebuilt case [22]. 13
2.3 DJI Zenmuse L2 lidar integrated onto a UAV drone [25]. 14
2.4 Garmin rangefinder v3hp lite [29]. 16
2.5 Arduino uno q [32]. 18
2.6 8gb raspberry pi 5 board [34]. 19
2.7 Micasense rededge-p multispectral sensor kit [36]. 20
2.8 3D printed housing for four multispectral cameras and one thermal camera [37]. . . 22
2.9 Multispectral system through single camera with diffraction grating [39]. 23
2.10 Sentera 6X thermal pro sensor kit [40]. 23
2.11 MosaicCORE 320x240 and 200x150 resolution [41]. 24
2.12 Anker Zulu 10,000 mAh power bank [42]. 25
2.13 INIU 240W 1ft cable [**?**]. 26
2.14 Samsung INR18650-25R 2500 mAh battery cell [43]. 27
2.15 16.8V 1A power adapter li-ion battery charger with LED indicator from Amazon [45]. 28
2.16 USB-PD 2.0 / 3.0 to 5V 5A converter for Raspberry Pi 5 [46]. 28
2.17 Block diagram of the battery system for the camera-lidar system. 29

3.1 SolidWorks model of the sensor plate. 31
3.2 Dimensions of the sensor plate. 31
3.3 SolidWorks model of the battery cage. 32
3.4 Dimensions of the battery cage. 33
3.5 SolidWorks model of the cage door. 33
3.6 Dimensions of the cage door. 34
3.7 Final assembly of the housing unit to hold the components and battery in SolidWorks. 35
3.8 (Left) Top view with battery bank. (Right) Bottom view with lidar and camera. . . 35
3.9 Overall caption for the 2x2 image grid. 36
3.10 Wiring block diagram schematic. 38
3.11 Final assembly of the lidar to GPIO interface circuit. 38

4.1 (Left) Pi Imager: selecting Pi model. (Right) Pi Imager: selecting Trixie 64-bit OS. 39
4.2 Visual demonstration of the micro-HDMI method of accessing the Pi. 41
4.3 User interface of the Raspberry Connect portal. 42
4.4 Streaming feed of the Raspberry Pi with the camera-lidar application running. . . . 42
4.5 Remote shell feed from the Raspberry Pi connect portal. 43
4.6 Visual guide on launching the GUI-based camera-lidar software. 44
4.7 Streaming feed of the Raspberry Pi with the camera-lidar application running. . . . 45
4.8 Setup to launch the GUI-less version of the camera-lidar software. 46
4.9 Home page of the GUI-less camera-lidar software. 47
4.10 SSH extensions in Visual Studio for data retrieval. 48

4.11 Hostname command in the terminal. 49
4.12 (Left) Control Center from Desktop GUI (Right) Interface Settings. 49
4.13 VS Code SSH connection command inputs. 50
4.14 SSH connect to Pi option. 50
4.15 VS Code SSH connection password prompt. 51
4.16 VS Code SSH open folder taskbar. 51
4.17 VS Code SSH open folder UI. 52
4.18 VS Code SSH data location and preview. 52
4.19 Downloading a file from the camera-lidar subsystem in VS Code. 53

5.1 Sample images taken with the Camera module in 1920x1080p setting. 54
5.2 Proof of functionality of the camera module. 55
5.3 (Left) Top view of a stationary camera-lidar system. (Right) Front-view of a stationary camera-lidar system. 56
5.4 Video feed of the camera module in the stationary setup. 57
5.5 Photo of the video feed taken from the SJSU Aerospace office. 57
5.6 Photo of the 8th floor science building with the camera live feed. 58
5.7 Visual demonstration of accessing the camera live feed from a remote setting. . . . 59
5.8 Lidar CSV data file. 60

List of Tables

1.1 Camera-lidar system mission requirements. 9
1.2 Gantt chart for AE295A/295B 10

5.1 Initial battery test of camera-lidar system at full power with the Anker Zulu power bank 61

A.1 Specifications for the Arducam 64MP OwlSight [24] 70
A.2 Specifications for the Garmin Rangefinder V3HP lite [29] 71
A.3 Specifications for the 8GB Raspberry Pi 5 board [33] 72

B.1 Comparison of camera components [21, 24] 73
B.2 Comparison of microcontroller components [31, 32] 74

C.1 Pechlivani et al. bill of materials [39] 75
C.2 Barjaktarovic et al. bill of materials [37] 76
C.3 Current bill of materials 77

Symbols

Symbol	Definition	Units (SI)
Acronyms		
Lidar	Light Detection and Ranging	—
UAV	Unmanned Aerial Vehicle	—
DSLR	Digital Single-Lens Reflex	—

1. Introduction

1.1 Motivation

There has been significant interest among archaeological researchers at the Popocatépetl volcano in Mexico in the uncovering of hidden Aztec structures. Previous research has uncovered various ancient structures and magma chambers associated with the volcano [1, 2]. However, the data used to collect these studies have involved several ground stations and the utilization of small aerial vehicles to survey the surrounding terrain [3]. Due to Popocatépetl's history as an active volcano [4], collecting terrain data in the surrounding area poses several challenges.

The first is that Popocatépetl hosts an active volcano that can emit ash for several miles, potentially endangering nearby researchers. Second, there are limited human amenities, roads, and resources near the volcano, as it is largely covered by dense vegetation. Two immediate solutions to alleviate these issues are to embed different sensors, such as a topological sensor, a camera, and/or a lidar system, on a small aircraft and fly over the volcanic region to perform a terrain survey. Another option could be to do something similar, but embed the systems on a UAV drone and also fly it over the volcanic region. Both ideas could, in theory, conduct a terrain-surveying mission to uncover hidden Aztec structures, contributing to the region's history. However, both ideas also share a third roadblock: they can be costly.

While it may be intuitive to pursue the UAV drone solution, this option introduces a fourth consideration: payload weight limit [5]. This can be an issue, as UAVs generally have a much lower weight limit than aircraft. Even then, high-tech systems such as DSLR cameras and 3D lidar systems can not only balloon budgets to tens of thousands of dollars but also carry substantial payloads. Such systems will require larger thrust on UAV drones, larger battery banks, which also contribute to weight, and sufficient power management systems to ensure nominal flight [6]. Mitigation of such issues can be achieved, but it may require a flexible budget — as one may notice when sourcing prices from online marketplaces [7, 8].

If budget is not an issue, theoretically, a perfect solution exists that solves these problems; however, such a solution is not currently available. However, that is one of the driving factors for this research project. The research presented in this paper addresses the following question: Is it possible to develop a low-cost hybrid camera-LiDAR system capable of performing a terrain-surveying mission? Can a system exist that not only has significantly reduced weight and power consumption compared to conventional models but also produces analyzable imagery? These research questions form the basis for the project explored in this paper, with the ultimate goal of creating a more equitable approach to performing aerial terrain-surveying missions through more innovative, more cost-efficient methods.

1.2 Literature Review

1.2.1 Research Applications of UAVs

Recent journal papers have highlighted the versatility of UAV drones across several fields. It is particularly useful due to its remote sensing capabilities, which enable the capture of terrain data with greater flexibility compared to a satellite. The consensus is that Zhang and Zhu's research on the current state of UAVs in the field can be primarily categorized into four areas: agriculture, forestry and wildfire management, power systems, and earth sciences [9].

In agriculture, water management plays a vital role in crop production. As Jaafar and Sujud's research suggests, a primary challenge arises not only in water distribution but also in classification, which helps logistically determine the optimal irrigation periods for crops. UAVs have been used in the Agricultural industry since they reduce the manual labor required to assess crop health. In the study's example, a university in Beirut successfully deployed 11 UAVs to measure evapotranspiration levels in a potato field throughout its growing season. The UAVs are equipped with sensors to measure crop water content. The data are then benchmarked against a flux tower that also measures crop evapotranspiration. The results from the UAV and the flux tower trend closely, though there is some variation between drones. There is room for improvement in the model used to parse UAV imaging data, which can more closely approximate actual evapotranspiration levels [10].

NASA has been developing an airspace management system for UAVs, specifically for wildfire scenarios, known as the Advanced Capabilities for Emergency Response Operations (ACERO). ACERO's use of UAVs is intriguing, as the UAVs are not only used for reconnaissance but also for offensive operations with fire-suppressant payloads. Previous norms relied on manned flights for overhead reconnaissance, surveillance, and data collection, but these were constrained by the need for human personnel. ACERO is working to enhance data collection scalability by developing an airspace management system that enables UAVs to operate 24/7. The use of UAVs enables firefighting missions in conditions that affect visibility, which may hinder them [11, 12].

Just as UAVs can be used to inspect crop health, they can be used to inspect power lines and systems for potential damage. Ma et al. describe a UAV-integrated system that performs real-time detection of power-line components to identify potential maintenance needs. The unique aspect of this study stems from the image classification model: when a UAV generates an image dataset, numerous background objects must be filtered out. This could pose a potential issue when identifying power system components. A combination of edge detection and image segmentation is used to match similar images in the dataset, thereby developing a depth map. This filters out unrelated background objects, enabling the UAV-generated image dataset to be used to identify and process power systems [13].

Román et al. provide an intriguing example of an Earth science study that compares UAV and satellite imagery using a machine-learning classification algorithm. A geographical survey of Antarctica, including observations of penguin colonies, was conducted as a benchmark for the algorithm. Two image-classification datasets were generated from the survey: one from a satellite and the other from a UAV. Using machine-learning classification to identify penguin colonies, the study concludes that the UAV dataset outperforms the satellite dataset in imagery classification. However, it does not necessarily rule out the use of satellite data altogether; in fact, an argument is made that a hybrid UAV-satellite dataset can improve image classification performance [14].

1.2.2 Research Applications of Lidars

Several recent journal papers highlight the use of lidar systems, with mission goals similar to those of this paper. Hedelt et al. conducted a terrain analysis mission utilizing several ground stations equipped with a spectrometric fluorescence lidar system. The system can collect atmospheric and height data around the Canary Islands' Tajogaite volcano, specifically in the vicinity of the volcanic plume. Data collected from the system can detect aerosol particles emitted from volcanic terrain, enabling researchers to estimate the environment's microphysical properties. Several lidar systems were used to obtain height measurements of the volcano, with the height data averaging within 1-3 km of the conventional satellite height measurement. One of the key results of the study concerns their lidar systems and the team's ability to obtain reliable height measurements for accurate mapping of the volcanic plume [15].

Pappalardo et al. conducted a similar study, also utilizing lidar systems to measure the properties of volcanic plumes. Their study provides a more detailed in-depth description of their "state-of-the-art" lidar system. The claim that they make is that their lidar system is innovative; not only can it take measurements in the UV and IR spectral regions, but also in more specific regions, such as:

- 1064 nanometers.
- 532 nanometers.
- 607 nanometers (N_2 Raman channel).
- 407 nanometers (H_2O Raman channel).

These wavelengths enable more precise measurements of volcanic aerosols. What makes this lidar system unique is its ability to emit three wavelengths simultaneously, a feature not typically found in lidar systems. This results in a flexible, highly accurate lidar system capable of measuring the microphysical properties of volcanic aerosols with configurable wavelength settings [16].

Canuto et al.'s work draws inspiration from one of the project's goals outlined in this paper. Their research details how they used a lidar system mounted on an aerial vehicle to perform

topological mapping of a Guatemalan rainforest in search of ancient Mayan settlements. The lidar system used is the Teledyne Optech Titan MultiWave, a 3D lidar system capable of capturing terrain data from an altitude of 650 meters above ground level. The lidar system generated multiple datasets across several regions in Guatemala, providing high-resolution, high-accuracy data. Using an aerial vehicle equipped with a lidar system, several datasets encompassing 2,144 km^2 of land. The parsing of the datasets involved a mix of computerized image modifications and manual labor to identify patterns resembling Mayan settlements — no artificial intelligence applications were used. The key result of the study identified 61,480 structures within 362 km^2 of modified land, revealing thousands of archaeological sites discovered using lidar systems [17].

Lidar systems are not limited to above-ground operations; they can also perform terrain analysis of underwater geometries. Harris et al. use a high-resolution lidar system to estimate coral reef surface roughness and correlate declining surface roughness with ecologically unhealthy ecosystems. The lidar solution used was a Rigel VQ-820-G topo-bathymetric lidar, combined with a Q860i-S topographic scanner and a Canon EOS 5D Mark IV DSLR camera, all of which were embedded onto a small aircraft to collect data. The parsing of the results utilized computer imagery and human classification to categorize several types of coral reef geometries, based on their roughness and landscape [18]. This research has inspired the hybrid design solution proposed in this paper, which uses a camera-lidar system embedded on a UAV drone rather than on a small aircraft.

1.2.3 Research Applications of UAVs with Camera-Lidar Sensors

One of the primary objectives of this project is to integrate a low-cost camera-lidar system onto a UAV drone capable of performing terrain-surveying missions. Several examples highlighted in journals feature a similar ecosystem, with varying configurations and purposes, designed to perform specific tasks. These journals highlight not only the capabilities of such a hybrid system but also the versatility of the research objectives, their respective outcomes, and conclusions. Xu, Li, and Bernardes developed a UAV drone equipped with RGB, multispectral, and thermal cameras, complemented by a lidar system. As seen in Figure 1.1, the research team developed a sophisticated

mounting system for the different camera sensors, all of which are controlled by a single Raspberry Pi 3.

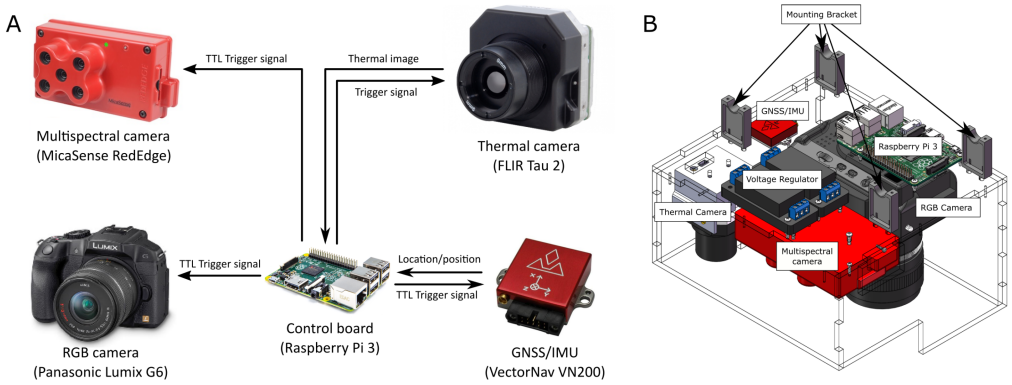


Figure 1.1 – Multi-camera and microcontroller UAV mounting system [19].

The mounting system shown in the previous figure is complemented by a 3D lidar camera and other components, forming the payload, as illustrated in Figure 1.2. The components mounted on the UAV conducted several terrain-surveying missions, capturing diverse data and producing more descriptive terrain information. While this setup is ideal for the mission objectives described in this paper, Xu, Li, and Bernardes' team did not perform an analysis with the lidar data generated from this study. Our current work involves not only developing a more cost-effective version of the drone described in the study but also performing a sensitivity analysis using lidar data [19].

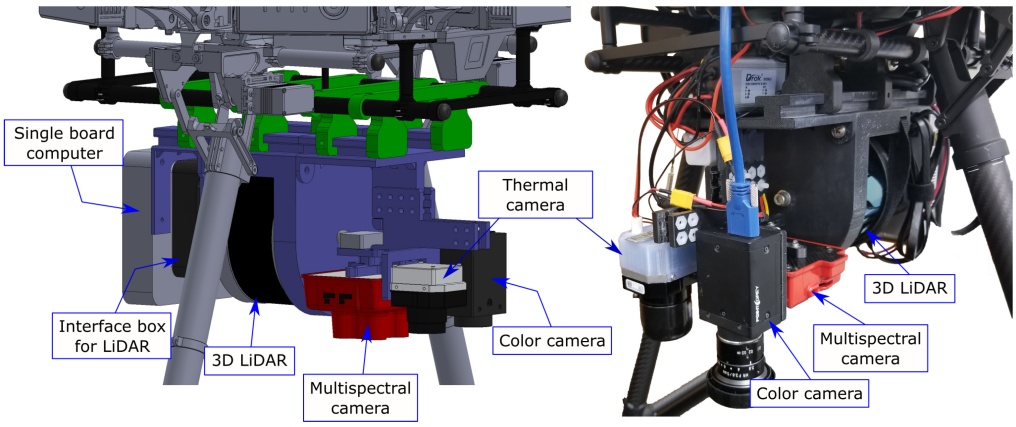


Figure 1.2 – Multi-camera, microprocessor, and lidar mounting configuration on a UAV drone [19].

It is further evident that terrain-surveying missions typically employ a multi-camera setup complemented by lidar systems, as demonstrated by Qin et al. Their research used a multispectral,

RGB, and lidar camera system to identify individual tree species. The 3D lidar system produced a series of volumetric point clouds that recreated the surveyed terrain geometry. The spectral and RGB cameras produced high-resolution image sets with finer texture detail in the trees, facilitating easier identification. Data collected from several missions were used to train a machine-learning algorithm to identify 18 tree species across a region in China. The key result of the machine-learning algorithm is that, using data collected from hyperspectral, RGB, and lidar systems, it correctly identifies tree species with 92% accuracy [20].

1.3 Project Proposal

1.3.1 Project Goals

The goal of this project is to develop a low-cost hybrid camera-lidar implementation for drone applications, with an emphasis on terrain-surveying missions. Normally, a capable 3D lidar system would be sufficient; however, the weight, cost, and/or power consumption of these lidars are infeasible for UAV applications. As evident from the literature review, these systems are typically attached to aerial vehicles, such as drones, with very large payload capacities that can exceed tens of thousands of dollars. This paper proposes a low-cost hybrid solution: a camera-lidar payload subsystem capable of capturing photos, videos, and distance measurements, providing a foundational proof of concept and functionality for remote data collection.

Using this system, a practical mission will be carried out with the Project Ares drone at the active volcano Popocatépetl in Mexico. The camera-lidar embedded system will be integrated into the drone's subsystems to reconstruct the geometry of the volcano's surroundings. This will contribute to ongoing research to identify archaeological ruins and potential gas tunnels near the active volcano. The mission will result in the creation of a 3D model of the surveyed environment. Pattern recognition will highlight terrain features and anomalies, potentially revealing hidden archaeological sites. This project provides the first milestone towards developing that system by providing a low-cost foundational unit capable of capturing video and measuring distance with lidar, all while operating in a standalone, portable unit that is easy to maintain and use. The completion

of the project will mark the first of many technical readiness stages that will eventually lead to a complete system.

1.3.2 Requirements

The purpose of the requirements is to ensure that if the camera-lidar systems operate within the minimum target specifications, they will collect sufficient data to complete a mission. While the Project Ares drone project has its own mission requirements and objectives, the requirements listed in Table 1.1 can, in theory, be fulfilled independently using any UAV compatible with the embedded systems proposed in this paper. The Project Ares drone is chosen due to its compatibility, as it shares a similar philosophy in designing a low-cost solution.

Table 1.1 – Camera-lidar system mission requirements.

Req. #	**Description**
1	The camera-lidar shall be self-contained and secured on a standalone payload housing unit.
2	The camera-lidar shall operate for a minimum of 15 minutes.
3	The camera-lidar shall save data obtained into a storage system (i.e., MicroSD).

1.4 Methodology

Over the two semesters, several meetings with the Project Ares engineering team have been conducted, as the camera-lidar system must integrate successfully with their microcontrollers and meet their specific requirements. To successfully integrate the camera-lidar subsystem into the UAV drone, a structured methodological process is developed. The basic methodology process follows a chronological list as follows:

1. Literature Review.
2. Obtaining necessary parts for the camera-lidar system.
3. Integration of the camera-lidar component onto a payload for the Project ARES UAV drone.
4. Designing the camera-lidar housing to house the microcontroller, wiring, battery, and sensors.

5. Selecting a battery system capable of powering the whole subsystem.
6. Assembling the housing and battery.
7. Performing tests of the camera-lidar subsystem.
8. Refine and optimize systems and processes.

To ensure that the project stays on track, a Gantt chart has been developed, as seen in Table 1.2. The planned objective for AE 295A was to conduct a literature review, recommend and procure parts for the camera-lidar system, integrate the camera-lidar system into a payload housing unit for the Project ARES UAV, select a battery system, and perform initial testing of the camera-lidar system. The plan for AE 295B involved assembling the housing, components, and software together in a secure housing.

Table 1.2 – Gantt chart for AE295A/295B

Objective	**Aug**	**Sept**	**Oct**	**Nov**	**Dec**	**Jan**	**Feb**	**Mar**	**Apr**	**May**
1: Literature Review										
2: Obtain camera-lidar parts										
3: Implement camera-lidar										
4: Design camera-lidar housing										
5: Select battery system										
6: Assemble housing and battery										
7: Perform initial tests										
8: Refine & optimize systems										

2. Hardware Consideration and Implementation

2.1 Introduction

In this chapter, the overall hardware considerations for implementing the camera-lidar system are discussed, including component selection, justification of component choices, and the architecture of the camera-lidar subsystem. The layout of this chapter is as follows: Section 2.2 discusses the chosen camera component, its technical specifications, and initial results to justify the selection. Section 2.3 discusses the Lidar component selected, including its specifications, capabilities, and limitations. Section 2.4 discusses the justification for the Microcontroller chosen. Section 2.5 presents additional considerations for integrating additional sensors and components within a flexible budget. Lastly, Section 3 discusses the integration of the selected components through various schematics and diagrams.

2.2 Camera

There are several camera options available. However, to narrow the selection of cameras, three considerations are heavily factored: weight, integrability, and cost, while providing decent image quality.

Weight is considered since the camera-lidar system must be easily carried by the Project ARES UAV drone. While there are no concrete payload specifications for the drone at the time of writing, a safe estimate is that a camera weighing less than 1 lb should suffice. Xu, Li, and Bernardes utilize multiple camera systems, including a multispectral, a thermal, and a DSLR camera, in their configuration [19]. For initial prototyping of this project, the primary focus is on the camera specifications typically used in DSLR cameras. While their research demonstrates that a microcontroller-operated DSLR camera can be integrated into a UAV, acquiring a capable DSLR camera may be costly. Therefore, a lightweight, low-cost camera is desired.

Integrability refers to the ease of use in interfacing between the camera and the microcontroller. Regardless of the chosen camera option, it must be capable of receiving and outputting data to

a microcontroller. While in flight, the UAV must be capable of capturing video and photos with sufficient fidelity to easily distinguish objects from at least 40 meters above the ground. This implies that it must be able to control the camera's focus via a microcontroller, preferably one with both Autofocus (AF) and Manual Focus (MF). It is also desirable to adjust the camera's light intake via the microcontroller by setting the ISO. Therefore, the camera option needs adjustable settings that can be easily interfaced with a microcontroller with minimal hassle.

Cost, one of the main drivers of this project, ultimately sets an upper limit on the available options. The intended upper limit for the camera selection is set to $100. Given the emphasis on operability with microprocessors, it's reasonable to start by selecting microprocessor-oriented cameras. Several vendors offer various camera options, but for simplicity, it is assumed that the microcontroller will be an Arduino or a Raspberry Pi.

The first camera consideration comes from the official Raspberry Pi website. It lists a configurable 12.3 MP camera sensor that is programmable via a Raspberry Pi board [21]. As shown in Fig. 2.1, the camera measures 7.99 millimeters, with a back-illuminated sensor architecture and adjustable back focus. It also supports various lens mounts, enabling customizable configurations that can potentially enhance image quality. Online retailers sell the component for about $50 [22]. The downside is that there is no mention of configurable ISO or AF/MF parameters. This could potentially pose an issue in calibrating the camera for 40-meter tests. While software tuning and additional accessories can provide remedies, there are further options that offer more configurable parameters out of the box.

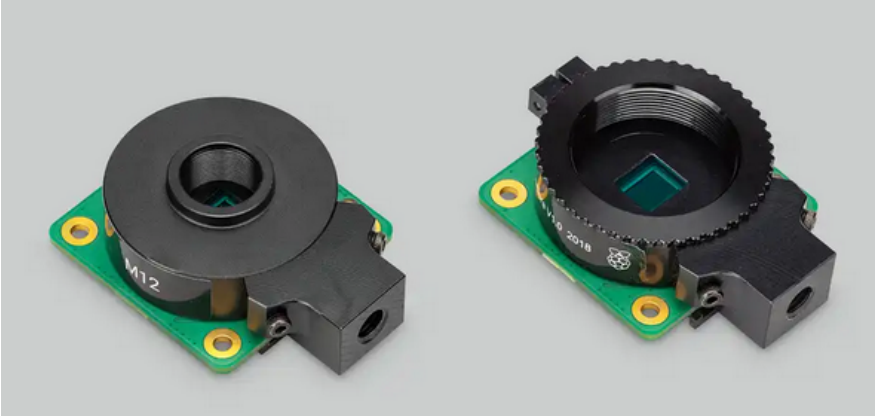


Figure 2.1 – Raspberry pi high quality camera [21].

The next selection of cameras comes from a range of options available from Arducam. They offer a wide range of cameras for Arduino and Raspberry Pi-based projects, with various configurations, setups, and hardware options. What was ultimately chosen was the Arducam 64MP OwlSight camera, which provides a prebuilt camera casing, resolutions up to 9248x6944, both AF and MF modes, and a software library with good documentation [23]. The cost for the camera also fits within budget specifications, costing about $60, and it comes with the necessary connections and a prebuilt case, as seen in Fig. 2.2 [24]. The camera specifications are laid out in Appendix A, Table A.1. Based on the specifications, the camera appears to offer good image quality, low weight, and strong interoperability with microcontrollers at a reasonable cost. A full comparison of the Arducam OwlSight and the Raspberry Pi camera is provided in Appendix B, Table B.1.



Figure 2.2 – Arducam owlsight 64mp camera with a prebuilt case [22].

2.3 Lidar

The utility from selecting a Lidar component is proportional to the amount an individual is willing to spend on it. This poses an issue, since the previous literature review of UAV systems using Lidar included relatively expensive (>> $1000) Lidar units. One of the popular 3D lidar devices in the surveying industry, the DJI Zenmuse L2 Lidar sensor, as seen in Fig. 2.3, has quoted costs that range $13,000-$15,000 at base price [25, 26]. While it is a capable device for UAVs, the main bottleneck is its cost. Thus, the selection of the Lidar primarily focuses on cost, preferably in the $100-$200 range.



Figure 2.3 – DJI Zenmuse L2 lidar integrated onto a UAV drone [25].

To elaborate further on the utility that can be gained from expensive Lidar components, specifically 3D Lidars. Such Lidar systems are beneficial because they can generate large volumes of point cloud data per measurement instance. The alternative, 2D Lidars, are also capable of generating point-cloud data but generally limit each measurement to a single point rather than multiple points. In terms of mission planning, it would require several measurement points for a 2D Lidar to match a single measurement that a 3D Lidar is capable of [27, 28]. However, in theory, it is possible to replicate the results of 3D Lidars. It requires more thoughtful mission planning

to compensate for 2D Lidar's limitations, which may be worth considering to save money while delivering comparable results.

After careful consideration, the Lidar component that was chosen is the Garmin Rangefinder V3HP lite, as seen in Fig. 2.4, which offers up to 40 meters of accurate range for $150 [29]. Although it is a 2D Lidar with limitations relative to a typical 3D Lidar, it offers advantages, primarily in weight and power consumption. The specifications for the Rangefinder V3HP are located within Appendix A, Table A.2, with notable key features [29] such as:

- Weight: 38 grams
- Range: 5 centimeters to 40 meters
- Nominal power: 0.425 W (5V, 85 mA)
- Peak power: 1.3 W
- Size: 54.99 x 40.18 x 35 mm (W x D x H)
- Cost: $150

In comparison to typical 3D Lidar systems, such as the DJI Zenmuse L2 Lidar, its notable key features [30] are:

- Weight: 905 ± 5 grams
- Range: 450 meters @50% reflectivity; 250 meters @10% reflectivity
- Nominal power: 28 W
- Peak power: 58 W
- Size: 155×128×176 mm (L×W×H)
- Cost: $13,000 - $15,000



Figure 2.4 – Garmin rangefinder v3hp lite [29].

The key takeaway from comparing the Rangefinder V3HP to a capable 3D Lidar system, such as the DJI Zenmuse L2 Lidar, is the weight and power consumption. The weight of the Rangefinder V3HP is 38 grams, while the DJI is about 905 grams. This is important to consider, since a lighter form factor and weight can help address both payload limitations and weight-balancing considerations for the drone to which it will be attached. The peak power draw of the Rangefinder V3HP Lidar is more than 50 times lower than that of the DJI Zenmuse L2, thereby placing fewer constraints on the battery system used with it. Additionally, unlike Xu, Li, and Bernardes' approach, which housed their 3D Lidar unit separately from the Raspberry Pi, the Rangefinder's power specs are compatible with a microcontroller such as an Arduino or Raspberry Pi. This can significantly reduce hardware complexity. Lastly, the cost of the Rangefinder V3HP is about 100 times lower than that of the DJI Zenmuse. While the Rangefinder's maximum range is significantly lower than the DJI Zenmuse L2's, this paper aims to determine whether a longer range is even necessary and whether it justifies the purchase of Lidar equipment that is 100 times more expensive. Therefore, the Garmin Rangefinder V3HP is a reasonable initial choice for the lidar component.

2.4 Microcontroller

The microcontroller serves as the core component of a camera-lidar system. Without it, there would be no direct or intuitive way for the camera, lidar, or other electrical components

to communicate or output data to other systems. Thus, it is highly convenient and justifiable to incorporate a microcontroller into the design ecosystem. However, several microcontrollers are available on the market. The design consideration developed by Xu, Li, and Bernardes utilizes a Raspberry Pi 3 as a central control board to send signals to the camera systems. Since then, newer versions of the Raspberry Pi have been developed, offering better performance with more features [31].

While the Raspberry Pi, as demonstrated in previous literature, has proven sufficient for similar objectives to this project, it is still worthwhile to investigate other components to clarify why it was chosen. Some of the main objectives in choosing a microcontroller are to provide sufficient power and interoperability among hardware components, while maintaining a software layer that can directly interact with them. This narrows the choices to two popular microcontroller brands: Arduinos and Raspberry Pis.

2.4.1 Arduino Uno Q

Arduino boards typically have fewer components and are less powerful than a Raspberry Pi. However, the simplicity of Arduinos can be advantageous, as they do not run an operating system; instead, they run the code uploaded to the device. The latest Arduino board, the Arduino UNO Q, connects directly via USB-C to any computer, where code is developed and maintained. It includes prebuilt apps written in Python or C/C++, serving as a baseline for programming projects, and real-time AI capabilities powered by pre-trained machine learning models. It also features several header pins and GPIO ports to support expansion modules, thereby adding additional hardware features to the board. Some downsides, as observed in Fig. 2.5, stem from the lack of a board and the absence of ribbon slots for camera sensors. It also lacks a MicroSD card slot, instead relying on onboard 16GB of eMMC storage [32]. These limitations may be mitigated with additional expansion kits for the Arduino Uno Q. Since the Arducam OwlSight camera module requires a ribbon connector for interfacing, it is highly desirable to choose a microcontroller with a dedicated hardware camera interface, at least for initial prototyping.

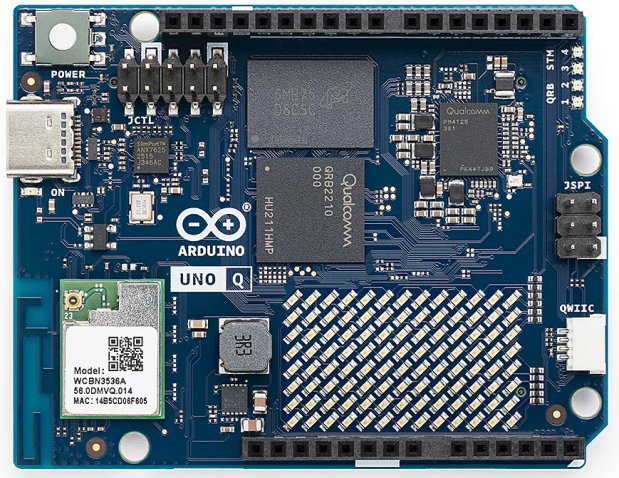


Figure 2.5 – Arduino uno q [32].

2.4.2 Raspberry Pi 5

The second Microcontroller consideration, the Raspberry Pi 5, is given greater weight, as previous literature has demonstrated that Raspberry Pis can interface with multiple cameras and sensors. The Raspberry Pi 5 is the latest hardware revision within the Raspberry Pi family, which offers more processing power and more ports (i.e., 2xUSB 3.0, 2xUSB 2.0, 1xEthernet Port, 2x camera transceivers, etc.) than the Arduino Uno Q [31]. As seen in Fig. 2.6, the Raspberry Pi 5 has a form factor similar to the Arduino Uno Q. Its main difference is that it is more of a miniature computer, as it does not require a direct connection to a computer to operate. Instead, it requires a flashed MicroSD card with an operating system (e.g., Ubuntu, Raspberry Pi OS, Debian, etc.) that runs natively on the Raspberry Pi. This allows greater independence, since it does not require any external computers to transfer code; however, it also supports wireless code transfer via the Secure Shell Protocol (SSH). Therefore, it is reasonable to choose the Raspberry Pi 5 as an initial Microcontroller because it has the necessary ports to support the Arducam OwlSight and Rangefinder V3HP Lidar system, it has additional open ports to add more sensors in the future, and it costs about $90 for a board with 8GB of RAM [33]. It is a relatively inexpensive Microcontroller capable of handling the components needed for a camera-lidar system. A complete list of hardware

specifications for the Raspberry Pi 5 is in Appendix A, Table A.3. A full comparison between the Raspberry Pi 5 and the Arduino Uno Q is presented in Appendix B, Table B.2.

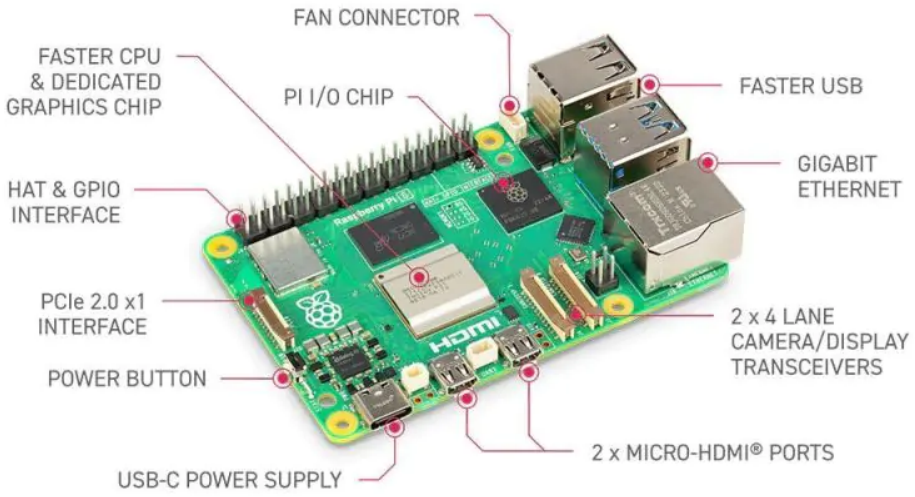


Figure 2.6 – 8gb raspberry pi 5 board [34].

2.5 Additional Considerations

Several additional hardware components can be integrated into the camera-lidar system to provide additional terrain data. The components listed in subsections 2.5.1 through 2.5.2 are additional sensors that can provide additional information for a terrain-surveying mission. However, the following sensors are used to gather highly specific information; while beneficial, they are not mandatory to meet the minimal mission requirements stated in this paper. Two additional camera systems are considered: a multispectral camera and a thermal imaging camera, all of which can be integrated in parallel with the camera-lidar ecosystem with the Raspberry Pi 5.

2.5.1 Multispectral Camera

The first sensor to add is a multispectral camera. The benefit of this camera lies in its ability to capture images across specific electromagnetic bands, revealing information that may be hidden from visible light. According to the literature review, multispectral cameras on UAVs are primarily used to monitor vegetation and crop health. In light of this paper's mission objectives, it may be

beneficial to use a Multispectral camera to observe the agricultural surroundings of archaeological sites near the volcano.

The main roadblock to incorporating a multispectral camera into the camera-lidar system is the high cost of the cameras. Several varieties of multispectral cameras are available for purchase on online marketplaces. The main commonality, however, is that the prices of multispectral cameras typically range from $5,000 to $10,000 [35, 36]. One example of a purchasable multispectral camera is the MicaSense RedEdge-P Multispectral Sensor Kit, as seen in Fig. 2.7. According to its specification, it weighs approximately 300 grams, requires 7-25.2 V, and requires a proprietary mounting unit for drone installation.



Figure 2.7 – Micasense rededge-p multispectral sensor kit [36].

The main concerns regarding the MicaSense RedEdge-P Multispectral Sensor Kit, aside from the $9,000 price tag, are its weight and operating voltage. Since the camera-lidar system is a relatively lightweight solution, an analysis is needed to ensure that the multispectral camera, Raspberry Pi camera, lidar, and Raspberry Pi can not only interface with each other but also maintain a balanced weight distribution. The second, more concerning factor, concerns the input voltage. The Raspberry Pi can supply 5V, which is enough for the camera and lidar; however, the voltage

may need to be stepped up to power the multispectral camera. This may significantly affect power consumption, potentially reducing drone flight time.

Even if a multispectral camera is available, factors such as shipping time and compatibility with the Raspberry Pi 5 must also be considered. Also, assuming, for redundancy, that the UAV carrying the camera-lidar system fails in flight, maintaining the high-end multispectral cameras can be expensive to repair or replace. Therefore, if a multispectral camera were to be integrated into the camera-lidar system, a low-cost, replaceable-parts approach should be considered.

Barjaktarovic et al. recognize the high cost barrier to multispectral cameras and have developed a Raspberry Pi-integrated custom solution that uses off-the-shelf components to construct a multispectral camera [37]. It also includes a thermal camera, which is discussed further in subsection 2.5.2. Their team opted for a modular camera design, using a 3D-printed housing to integrate the cameras and optical filters into a single system. Their camera implementation uses a $300 Arducam 4x1 MP Quadrascopic Camera [38], which requires only one USB port and one camera interface port on the Raspberry Pi. To achieve multispectral capabilities, they opted to purchase four filters: one single-band, one dual-band, and two triple-band filters, resulting in a combination that provides wavelength readings from 400 nm to 850 nm.

A power measurement study was conducted, and the operating voltage was 5V and the average current was 2A, yielding a total operating power of about 10W. They tested this against a 10000 mAh battery and found that the system can last about 5 hours under full usage. The final design, as shown in Fig. 2.8, uses a 3D-printed plastic enclosure; they recommend aluminum for hot sensing environments. In surveying an active volcano, this may be necessary. If this project were to include a multispectral camera system, the solutions proposed by Pechlivani et al. and Barjaktarovic et al. offer a relatively low-cost approach to its implementation.



Figure 2.8 – 3D printed housing for four multispectral cameras and one thermal camera [37].

Pechlivani et al. provide a second, lower-cost approach than the implementation described by Barjaktarovic et al. Their proposed system spans the spectral range from 379 nm to 937 nm, using a Raspberry Pi NoIR camera, a macro lens, a front-camera telephoto lens, and a diffraction grating. The entire subsystem, including a Raspberry Pi 4, costs just under $300, which is reasonable for the scope of this project. They assembled the parts into a 3D-printed portable case that holds the entire system, as seen in Fig. 2.9. The unique approach here is to use a single camera rather than four to capture information across different wavelengths. Some notable recommendations for future improvements include selecting a higher-fidelity camera component and experimenting with different optical lenses. Otherwise, the results from their study highlight its ability to capture up to 290 wavebands spanning 379 nm to 937 nm. The authors encourage readers to replicate the setup and to expand upon it, and a full bill of materials is located in Appendix C, Table C.1. Based on the work of Barjaktarovic et al., which offers a higher wavelength range, and Pechlivani et al., which offers an optimal low-cost solution, it appears that a hybrid solution can be developed that leverages strategies from both teams to deliver a more efficient, low-cost multispectral camera.

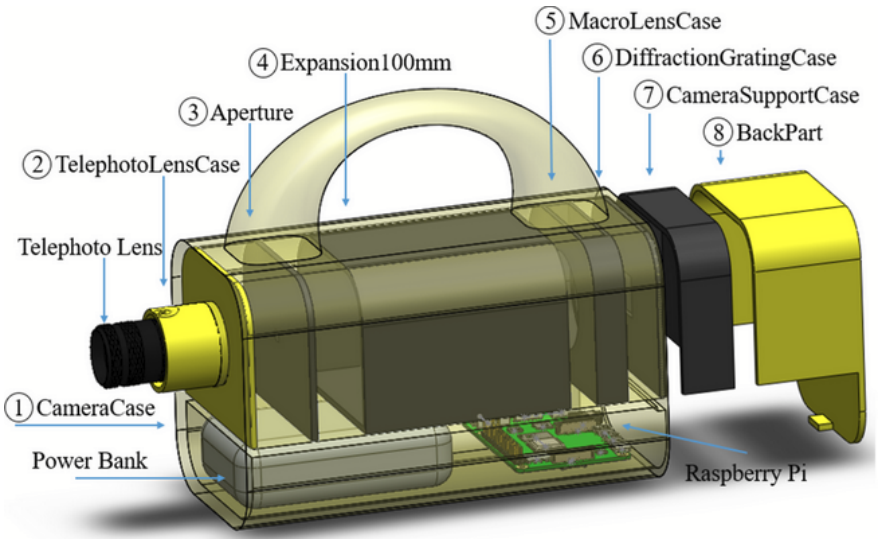


Figure 2.9 – Multispectral system through single camera with diffraction grating [39].

2.5.2 Thermal Imaging Camera

The next component to consider is a thermal imaging camera, which allows for temperature readings across a terrain. As with multispectral cameras, the main roadblock to integrating the thermal camera is cost. When observing online marketplaces to get a quote for thermal cameras, the price range ranged from about $4,000 to $17,000, depending on the model [40]. The higher-priced items, such as the Sentera 6X Thermal Pro Sensor Kit, as shown in Fig.2.10, also include a combination of multispectral, RGB, and thermal cameras. If cost were not a primary factor, then a device like the Sentera 6X could be the ”all-in-one” camera solution.



Figure 2.10 – Sentera 6X thermal pro sensor kit [40].

The benefit of a thermal camera, in combination with a multispectral camera, is that it can provide enough data to conduct studies regarding soil and vegetation moisture levels, as well as general thermal imaging across a terrain. It can support the mission goals outlined in this paper by enabling the UAV to uncover archaeological waterways near the volcano.

As previously mentioned, Barjaktarovic et al. provided a low-cost implementation that includes not only multispectral cameras but also a thermal camera. This is also visible in the previous subsection (Fig. 2.8), where the 3D housing case features a 2x2 multispectral camera grid, with an additional hole beneath it housing the thermal camera. Their team implemented a MosaicCORE 320x240 thermal camera, as seen in Fig. 2.11. According to their paper, a thermal camera costs approximately $500. Although it does not appear low-cost, it is about one-tenth the price of the lower-cost end of the industry's thermal cameras. Their paper also demonstrates that it can be integrated into a Raspberry Pi ecosystem. The total cost, including the multispectral and thermal cameras from their approach, amounts to about $2200. A full bill of materials from the paper by Barjaktarovic et al. is located in Appendix C, Table C.2. Therefore, if a low-cost thermal camera is needed for the camera-lidar system, the approach taken by Barjaktarovic et al. seems optimal



Figure 2.11 – MosaicCORE 320x240 and 200x150 resolution [41].

2.6 Battery System

A limitation of the setup is that power is provided by a power cable plugged into an outlet. To perform future experiments, the power source must also be mobile. To ensure power-source

portability, a 10,000 mAh power bank was purchased on Amazon, equipped with an end-to-end USB-C cable capable of delivering 5V/5A. The product is called the Anker Zulu 10,000 mAh Power Bank, as shown in Fig. 2.12. This was purchased to have a robust, portable power source for prototyping. This initial selection is not the final implementation of the battery system, but rather an accessible first choice.

Recent testing indicates the power bank can power the Raspberry Pi ecosystem, including the OwlSight camera. The main drawback is its size and weight. According to the datasheet on Amazon, it weighs 212 grams, with the item dimensions in inches of: L x 2.58” W x 0.98” H [42]. Although the included cable is supposedly capable of 30W power delivery, initial testing showed various power instability issues. To fix this, the USB-C out port next to the included cable was used, and a 240W-capable cable was used instead to deliver power from the power bank to the Pi, as seen in Fig. 2.13 Overall, for prototyping, this solution works great and is ready out of the box; however, there is ample room to improve battery sizing to reduce weight.



Figure 2.12 – Anker Zulu 10,000 mAh power bank [42].



Figure 2.13 – INIU 240W 1ft cable [**?**].

To provide additional detail on the future battery system, several options are being explored at the time of writing. The main plausible approach to reducing the weight of the battery payload is to design a lithium-ion rechargeable battery system that integrates battery cells in series and/or parallel to meet the desired watt-hour target. More specific details are needed to determine the exact power target, but based on the previous Amazon purchase, the Anker Zulu Power Bank is 10,000 mAh. For the final iteration, given the mission target of 15-minute missions, the watt-hour target can be comfortably reduced to about 5,000 mAh. This lower target is reasonable because it would exceed the 15-minute target even if the subsystem were operating at full power. Comprehensive power calculations will be included in a future iteration of this report.

The current concept involves a 4-series-2-parallel battery system, with an initial target of 5,000 mAh. The battery cells to be used are the Samsung INR18650-25R 2500 mAh batteries, as seen in Fig. 2.14. According to the Samsung battery datasheet, four batteries in series would yield

16.8 V at full capacity. An additional four batteries in series, parallel to the other four, would double the energy storage capacity from 2500 mAh to 5000 mAh, hence the name 4S2P.



Figure 2.14 – Samsung INR18650-25R 2500 mAh battery cell [43].

The challenge is to design a system that can safely monitor, charge, and discharge batteries in a redundant configuration. Typical 4S2P battery packs on the market include a "Battery Management System" (BMS) that sits in between the battery cells and the load/charger [44]. At the time of writing, the selection for the BMS has not been finalized. However, it is worth detailing the features typical BMS systems provide, such as:

- Preventing overcharging and short circuiting
- Load balancing, to ensure all cells in series have the same voltage
- Current limiting and power shutoff in the case of unsafe power levels
- Temperature monitoring to terminate charging/discharging states in the case of overheating
- Routes the charging current safely to cells using controlled constant current/constant voltage charging

Although the BMS can manage power effectively, it does not include a charger. This is a separate component that plugs into a wall socket and has a port that interfaces with the BMS. Ideally, the lithium battery charger should include internal charging protection as a first layer of redundancy before reaching the BMS. The initial selection is a lithium-ion battery charger from Amazon, as

seen in Fig. 2.15. It provides an output voltage of 16.8 V to charge 4S lithium battery packs [45]. It also includes an LED indicator on the charger to indicate whether power is flowing properly.



Figure 2.15 – 16.8V 1A power adapter li-ion battery charger with LED indicator from Amazon [45].

The final component under consideration is a power regulator that takes input from the BMS and delivers a reliable 5V/5A output to the Raspberry Pi. A purchasable PCB board is available online, designed to do precisely that, as shown in Fig. 2.16. This device, called the Pichronida, takes the BMS as input and outputs a reliable 5V/5A to the Raspberry Pi via USB-C cable [46]. The 5V/5A power supply powers the Raspberry Pi, the camera, and the lidar. However, it may not be using 5V/5A continuously; the Raspberry Pi documentation recommends this power target to ensure all systems receive sufficient operating power. This completes the interfacing circuit between the battery and the Raspberry Pi, and a block diagram of the battery system is shown in Fig. 2.17.

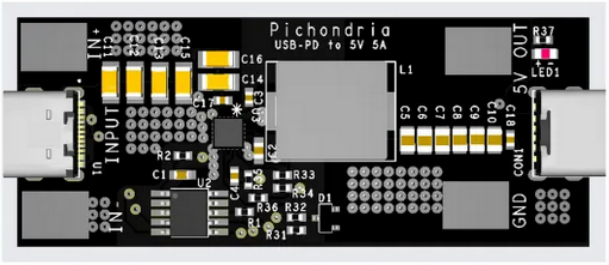


Figure 2.16 – USB-PD 2.0 / 3.0 to 5V 5A converter for Raspberry Pi 5 [46].

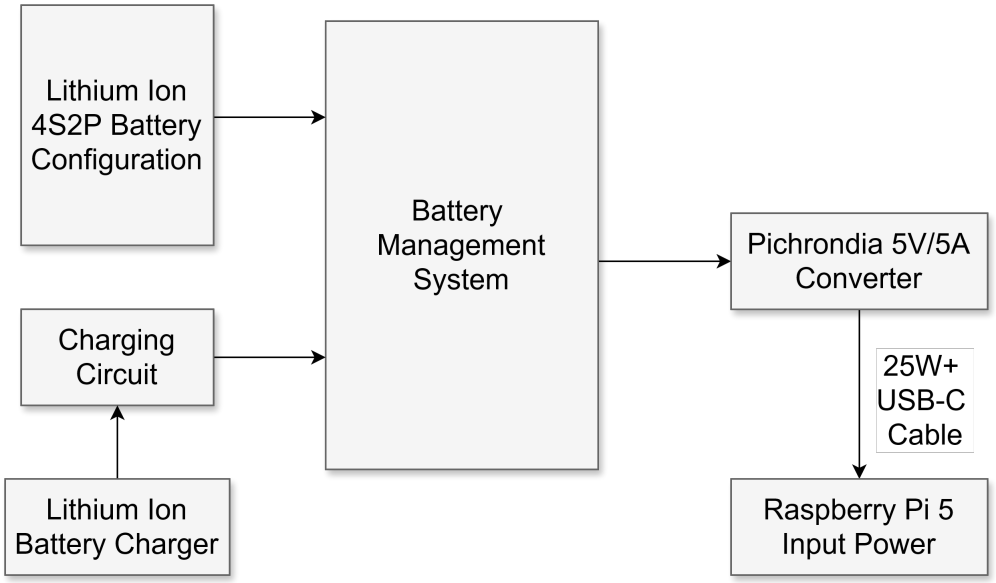


Figure 2.17 – Block diagram of the battery system for the camera-lidar system.

This detailed concept could significantly reduce the weight of the camera-lidar battery system, though implementing it would require substantial development time. Future iterations of this project should begin with this improvement to reduce weight while maintaining sufficient energy for the subsystem.

3. Subsystem Housing and Integration

3.1 Subsystem Housing

The housing system for the camera-lidar subsystem comprises three fully 3D components. To ensure the parts work together seamlessly, a housing system is required to hold all components and their wiring. The first part, as shown in Fig. 3.1, houses the sensors and microcontrollers, including the Raspberry Pi 5, the Garmin Rangefinder V3HP, and the OwlSight 64 MP camera. Fig. 3.2 shows the CAD of a simple plate with several holes. The holes are sized based on the screwing holes on each of the previous 3 hardware components. All components are to be screwed down into the 3D-printed plate, which is sufficient to secure them in a stable position. Each corner of the sensor plate contains a large cylindrical pole that protrudes upward. Additionally, there are two holes in the one-sided view of the housing that are intended to hold a small breadboard, allowing some circuitry components, such as resistors or capacitors, to help protect the components from inrush currents. The holes utilize M2 screws. These are added to the plate, as shown in the next paragraph, to house the part that holds the battery component on top of the sensor plate. The vertical height budget is 86.5 mm from the bottom of the UAV drone to the landing-gear bar, indicating that careful consideration is required to ensure that components do not contact the landing-gear bar.

The second part, as seen in Fig. 3.3, along with the dimensions in Fig. 3.4. The design contains a rectangular cage with holes on the bottom. The intention for this design is to hold the 10,000 mAh battery, which lacks mounting capabilities, in a cage with a 45W cable exiting from one end, preferably closer to the USB-C power port on the Raspberry Pi 5. Before the battery gets inserted into the cage, there are holes inside the cage that are meant to be screwed into the four pillars from the first part. The hole dimensions are designed to fit M2.5 screws, so the hole tolerance is about 2.2 mm. There is a larger circle around the hole meant to make the screw sit flush with the surface where the battery is supposed to go. This way, the screws do not come into contact with the battery. There is a large honeycomb design on the roof of the cage, with four honeycombs large enough to allow a screwdriver to pass through and twist the screws comfortably. Once the screws

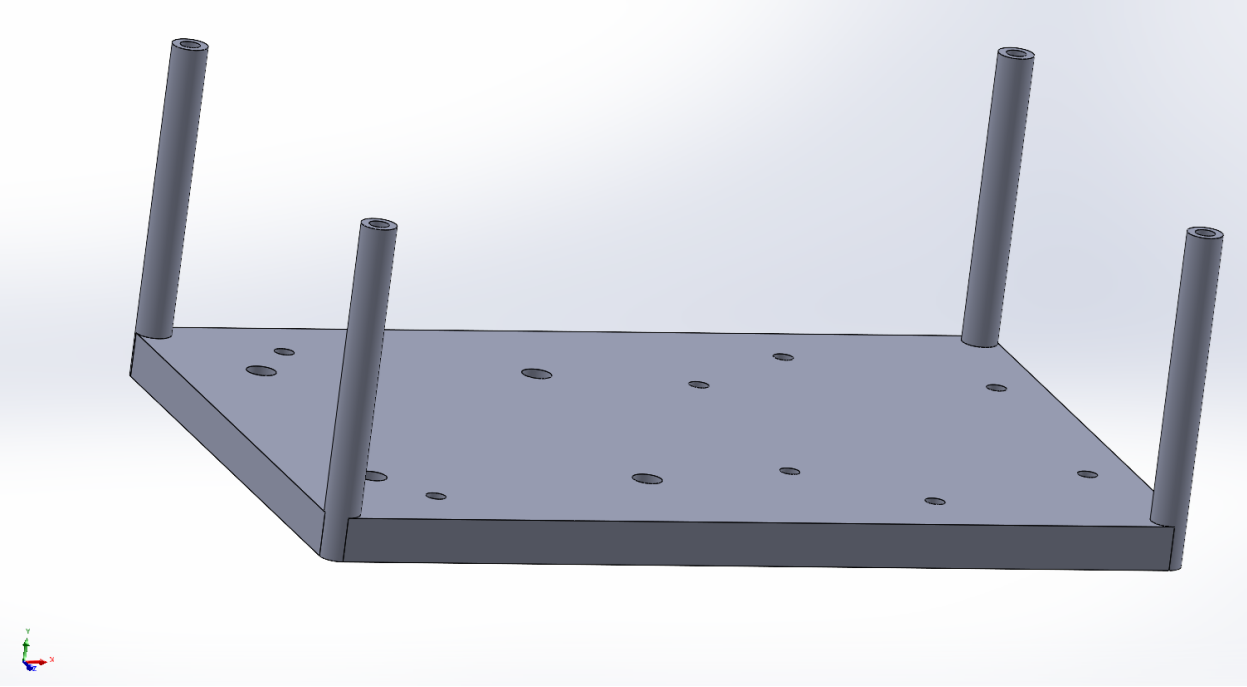


Figure 3.1 – SolidWorks model of the sensor plate.

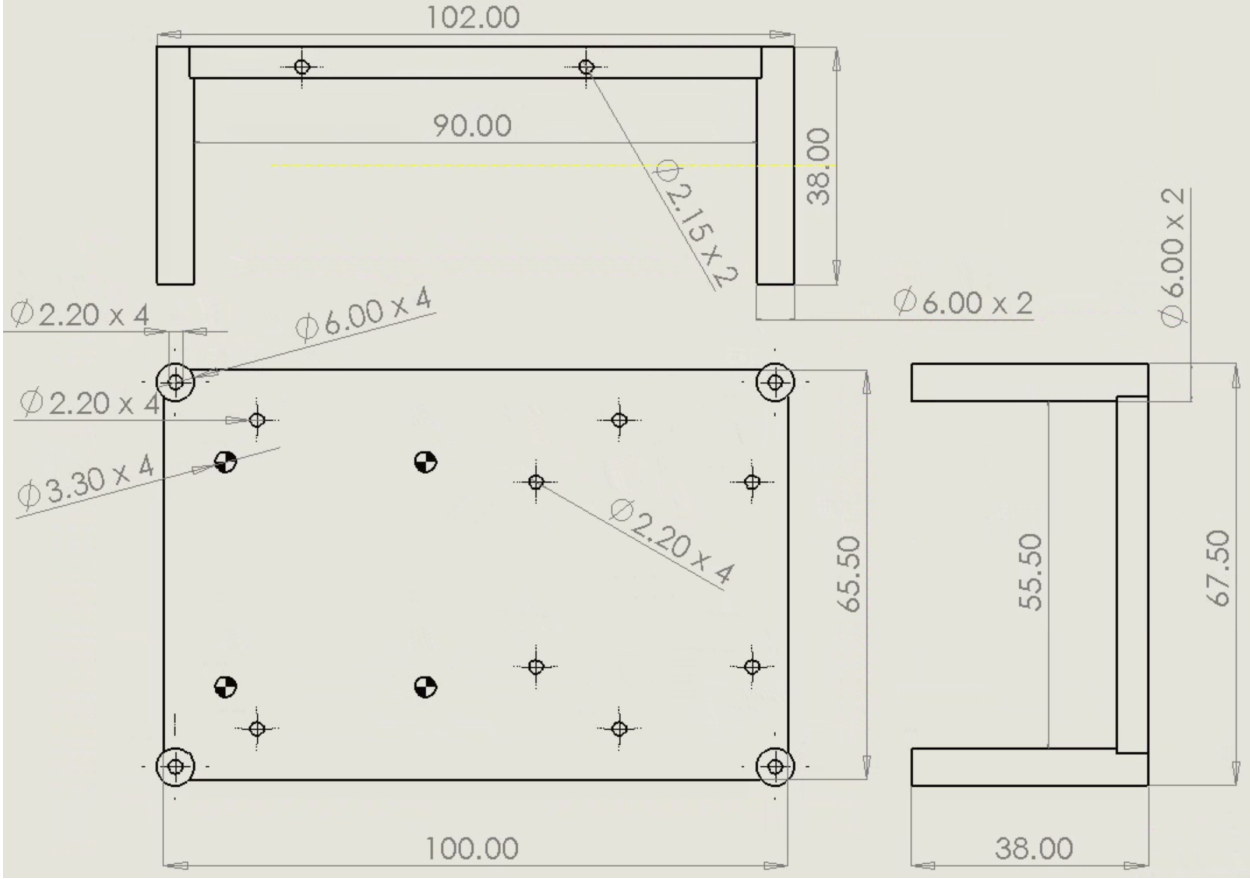


Figure 3.2 – Dimensions of the sensor plate.

are in, the battery can be inserted into the cage comfortably. There is an opening on the left side of the cage that allows the user to turn on the battery from the side. Additionally, looking at the top, there are four large holes at the corners. These are the mounting screw holes for the housing to be screwed onto the bottom of the UAV. As requested by the UAV design lead, the holes are intended to accommodate M4 screws. This provides enough security to lock the housing onto the UAV.

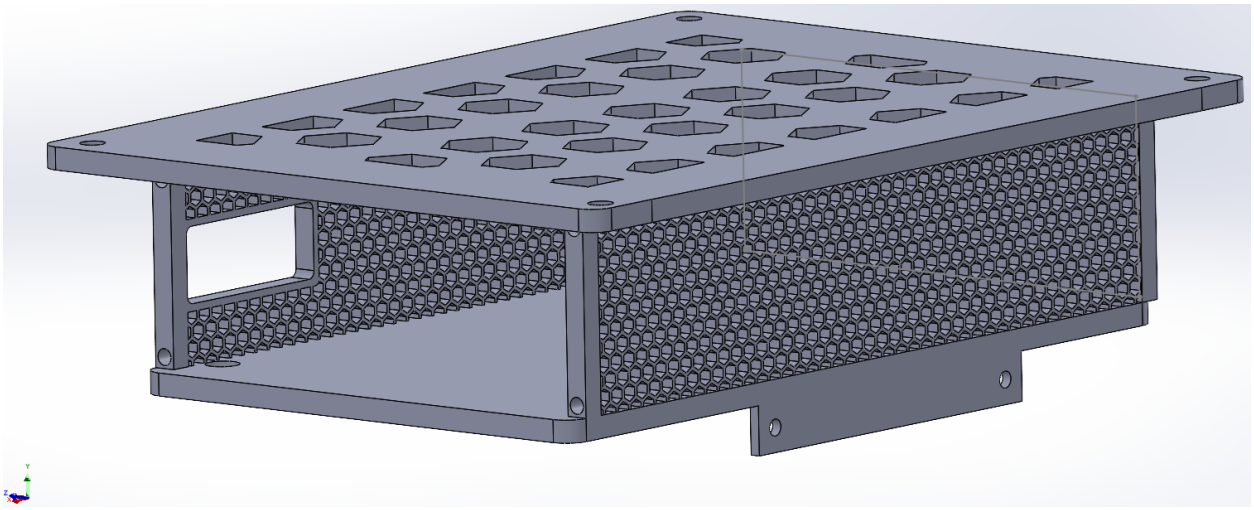


Figure 3.3 – SolidWorks model of the battery cage.

The final part is the door that secures the battery in the cage so it doesn't fall out, as shown in Fig. 3.5, with dimensions shown with the dimensions in Fig. 3.6. As shown on the front of the cage, there are four holes: two on the left and two on the right. The door to secure the battery is to be screwed into the holes, so that the battery remains secure. The holes also use M2.5 screws, so the PLA hole dimensions are 2.2 mm. The front door features a hole, where the 45W cable exits the design and feeds into the Raspberry Pi.

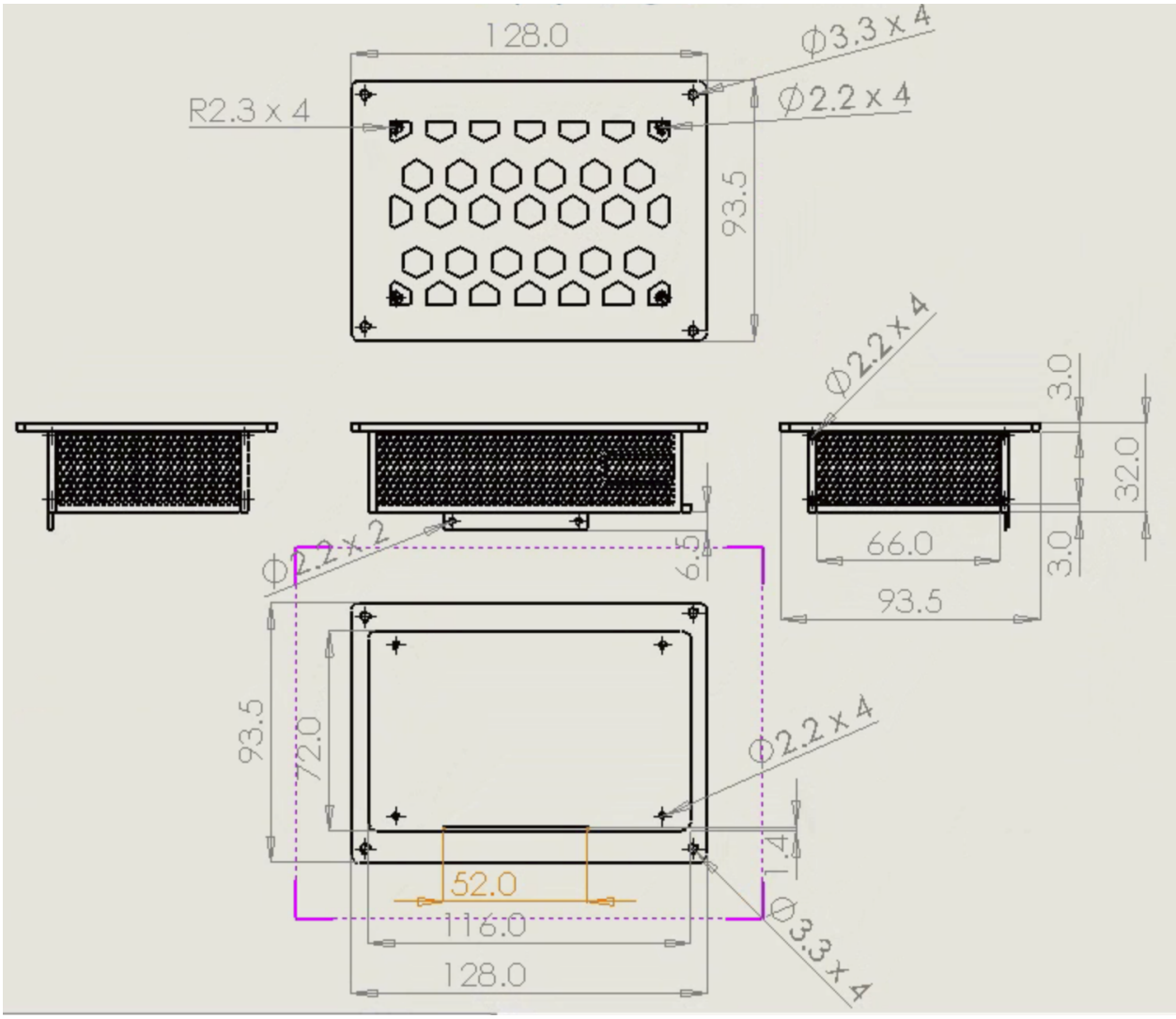


Figure 3.4 – Dimensions of the battery cage.

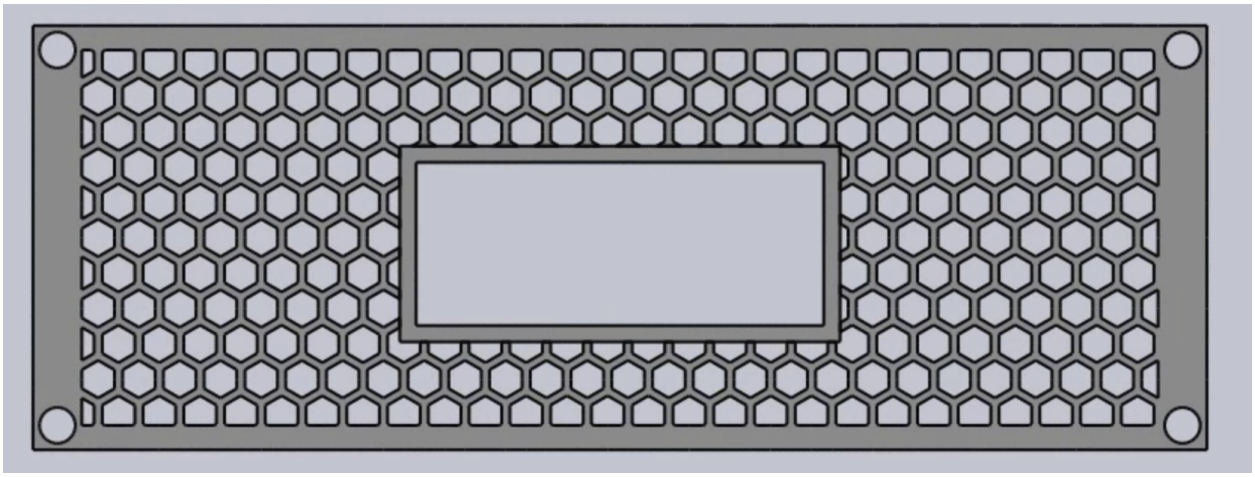


Figure 3.5 – SolidWorks model of the cage door.

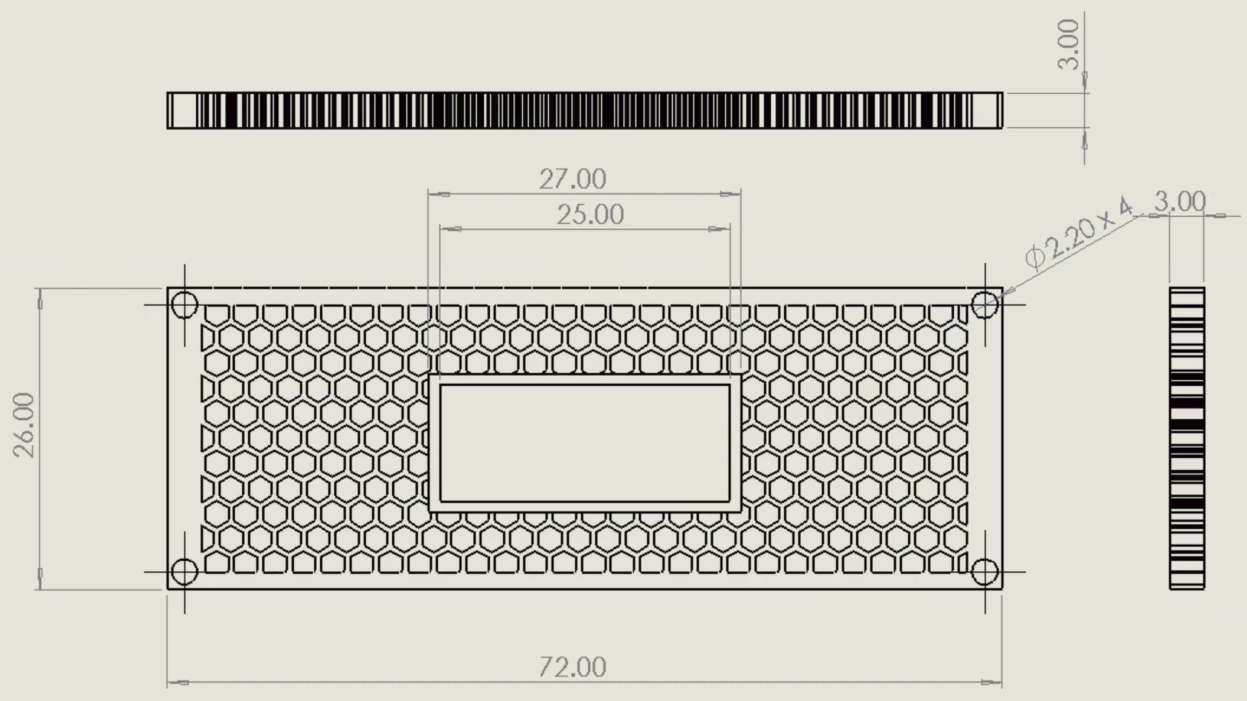


Figure 3.6 – Dimensions of the cage door.

Each part is 3D-printed in PLA. Originally, the overall printing time for the parts was about 2 hours, so to reduce both material and print time, a series of honeycombs was added to the battery housing mount. This allows the battery container to maintain structural integrity while reducing the filament required and the print time. The final assembly was done in SolidWorks, as shown in Fig. 3.7, which took longer because it required 3D-printing three separate parts with a single printer. However, it may save time in the future, since if a certain part needs modification, only that part needs to be 3D printed rather than the whole housing. The final assembly after 3D printing the parts are shown below. Fig. 3.8 shows the top and bottom views of the assembled system, and Fig. 3.9 shows all the side viewpoints of the camera-lidar subsystem.

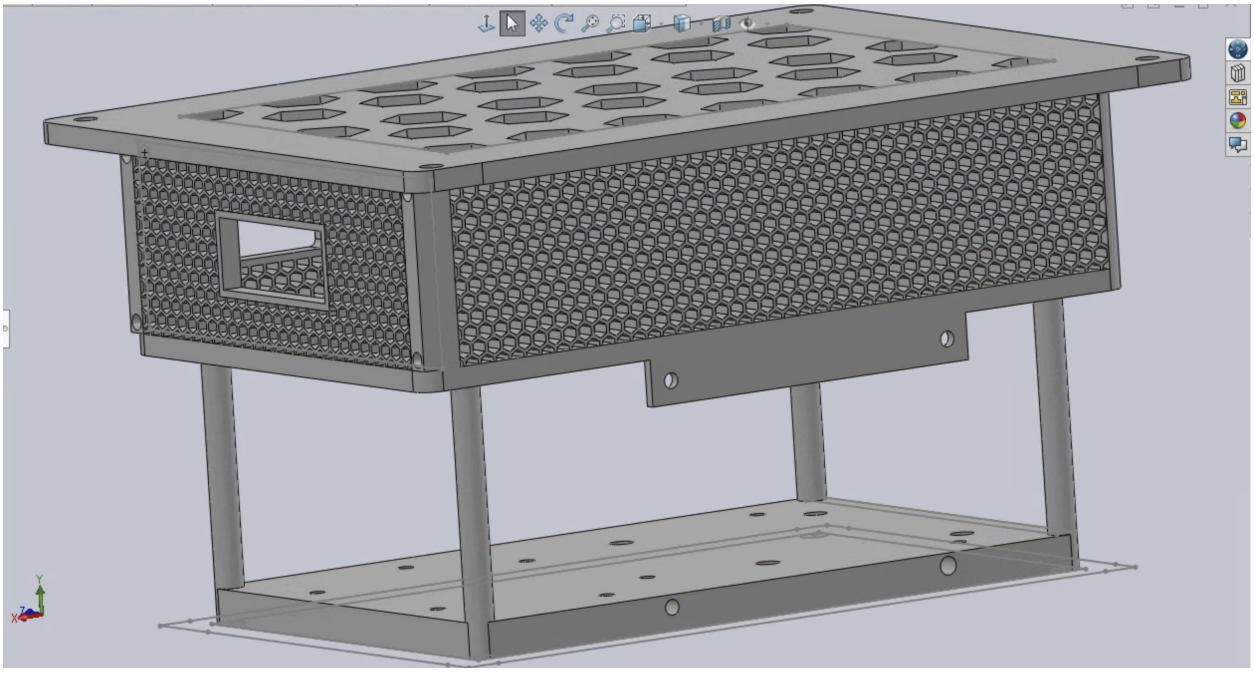


Figure 3.7 – Final assembly of the housing unit to hold the components and battery in SolidWorks.

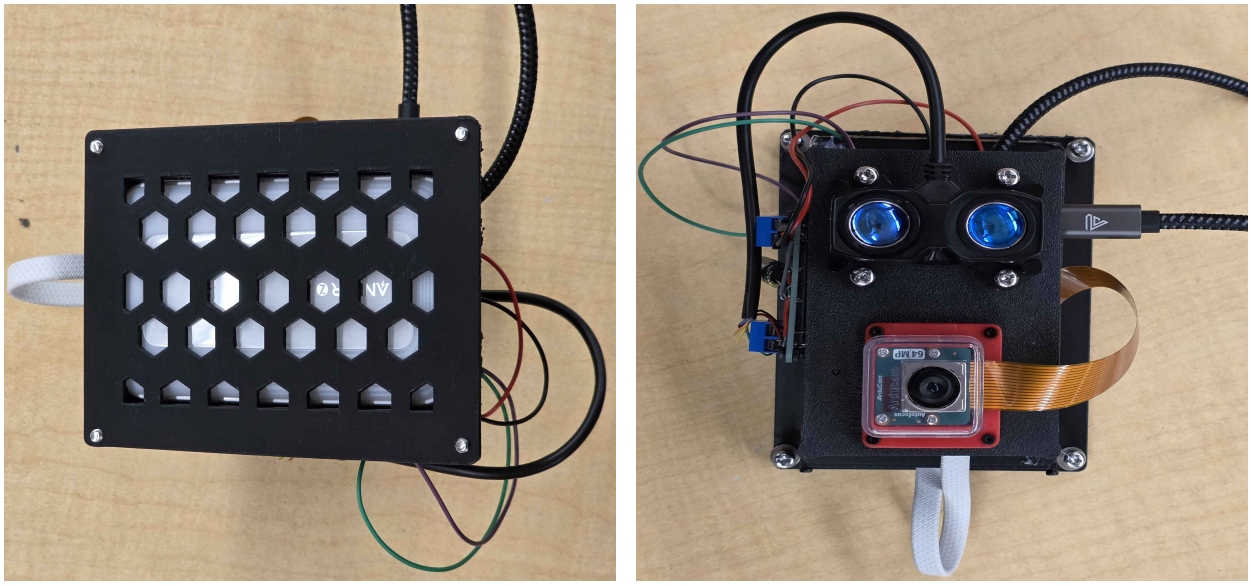


Figure 3.8 – (Left) Top view with battery bank. (Right) Bottom view with lidar and camera.

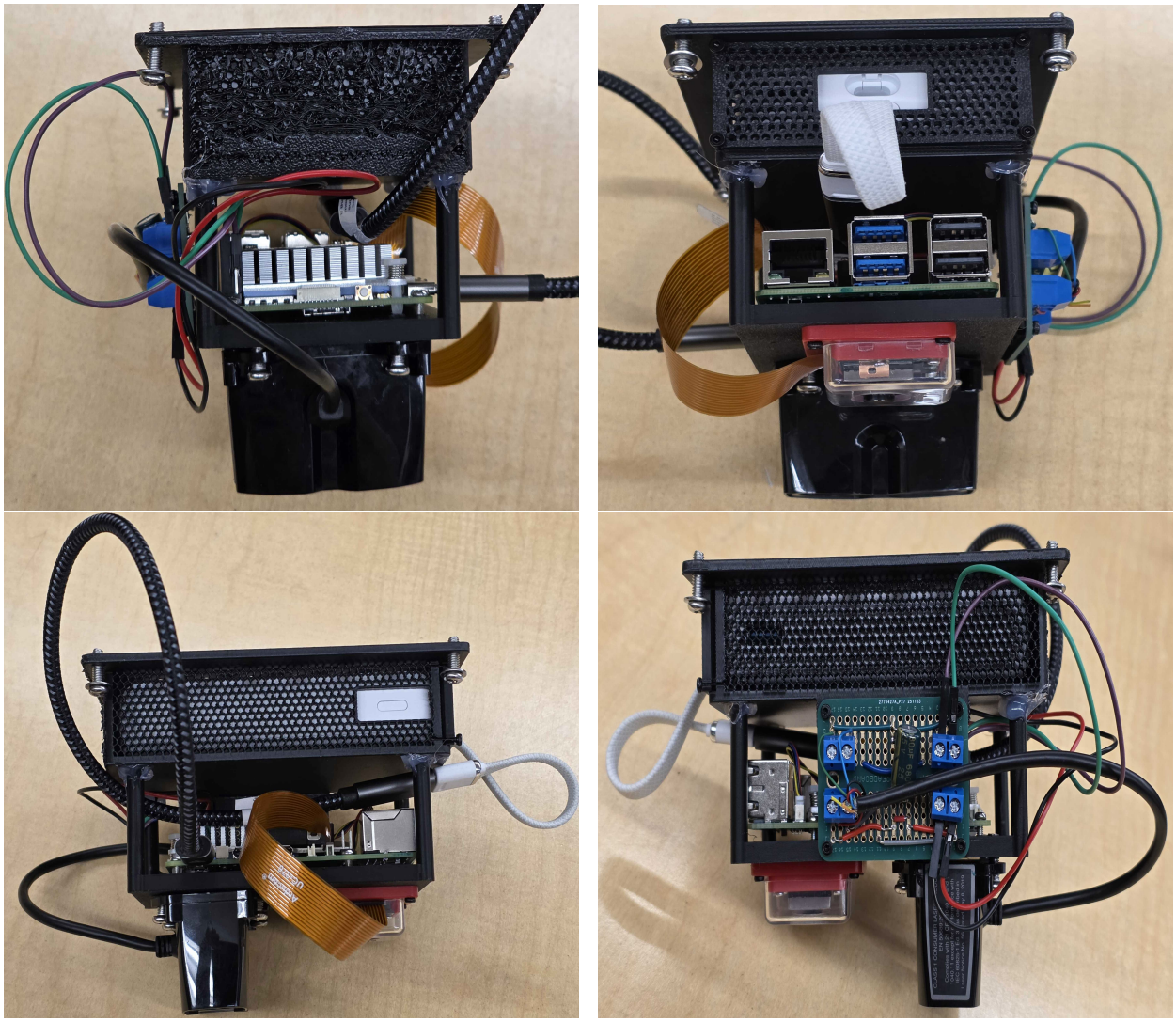


Figure 3.9 – Overall caption for the 2x2 image grid.

There can be further refinements to improve the 3D housing for the camera-lidar system. For example, a couple of the holes that hold the bottom part of the breadboard are larger than expected, and thus, do not hold the screws as securely. The upper screws hold the breadboard adequately, but more work is needed on the tolerancing of the bottom holes. Additionally, the honeycomb in the back of the top part contains a hard-to-remove support filament. The spacing between the honeycombs can also be enlarged to further reduce material while maintaining structural integrity.

3.2 Electronic Integration

According to the Garmin Rangefinder V3HP datasheet, a capacitor must be installed between the 5V and ground cables to prevent potential current rushes [29]. Additionally, there needs to be an interface that allows a direct connection between the lidar sensor's wires and the Raspberry Pi 5's GPIO pins. What has been designed is a breadboard circuit that is screwed onto the side of the housing, with wires soldered onto the back of the breadboard for secure connection, and screw-in terminals that accept the Rangefinder wires as an input, and it outputs 4 wires that lead into the GPIO pins, which are also secured by screw-in terminals. The reason for this design is to allow easy maintenance if one of the wires becomes faulty. If the lidar sensor also becomes damaged or non-functional, it would be very easy to remove it and replace it with a new one. A wiring block diagram that shows the interfacing between the lidar and GPIO pins is shown in Fig. 3.10. Based upon the lidar datasheet, the red wire needs to go to a 5V line (GPIO 2), the black wire goes to ground (GPIO 6), the green wire goes to I2C SDA (GPIO 3), and the blue/purple wire goes to I2C SCL (GPIO 5) [29, 47]. The datasheet requires a 680 μF capacitor to be interfaced between the GPIO pins and the lidar system, with 5V and ground. The cathode (short leg) is connected to ground, and the anode (long leg) is connected to 5V. To avoid mixing wires, the breadboard fully utilizes both halves: the green and blue wires on the top, and the ground and 5V on the bottom, with the capacitor. The final assembly of the breadboard is shown in Fig. 3.11, where the lidar input cables are on the left, and the output to the GPIO pins is on the right. Further improvements can be made to the circuit housing, primarily by providing a removable outer shell that protects the circuit from weather debris and external forces. Additionally, some cable management and placement optimizations can make the organization tidier.

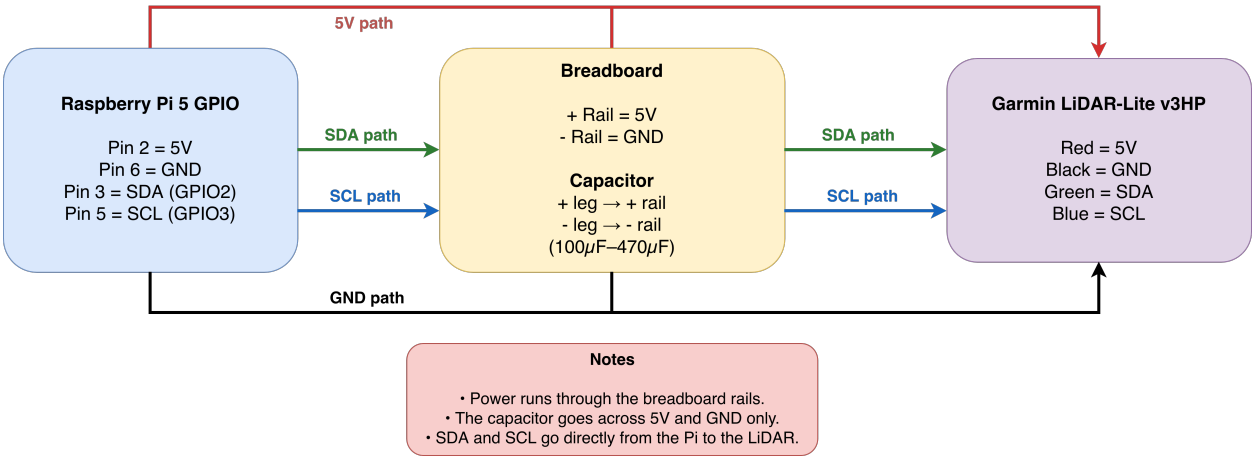


Figure 3.10 – Wiring block diagram schematic.

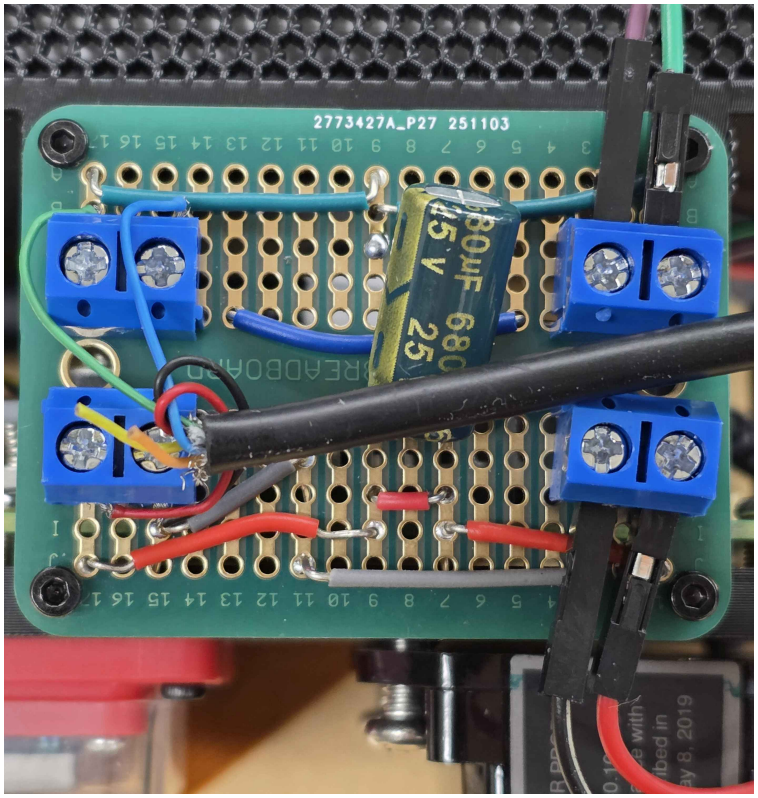


Figure 3.11 – Final assembly of the lidar to GPIO interface circuit.

4. Software Environment

4.1 Operating System and Installation

The Raspberry Pi 5's OS uses the latest version of the Raspberry Pi OS family, Trixie [48]. This operating system version features performance improvements over previous iterations, which is important since camera and lidar usage can be taxing on the CPU, so every bit of performance efficiency is desired. The operating system was flashed into a MicroSD using an SD card reader attached to a MacBook via a USB-C port. The Raspberry Pi developers provide a program called *Raspberry Pi Imager* to flash the OS onto the MicroSD card [49]. As seen in Fig. 4.1, the OS can be flashed based on a specific Raspberry Pi Model, and the OS can be chosen. Other settings, such as device username and password, can be set to the default values during installation.

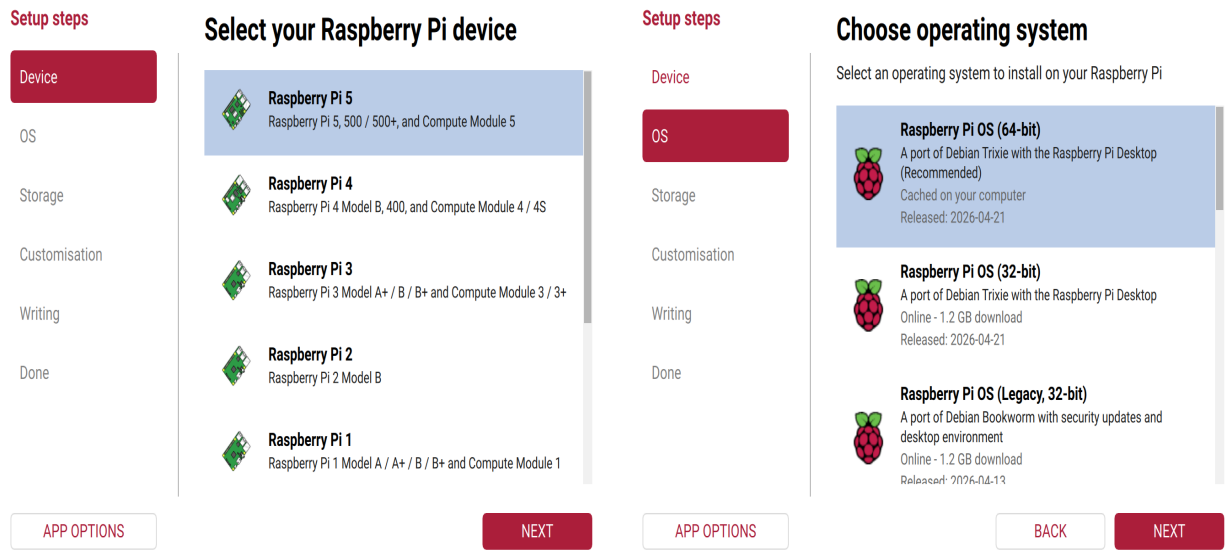


Figure 4.1 – (Left) Pi Imager: selecting Pi model. (Right) Pi Imager: selecting Trixie 64-bit OS.

4.2 Accessing the Raspberry Pi

There are three methods for accessing the Raspberry Pi, all of which are valid approaches for reading and modifying data on the Pi. Before attempting any methods, the Raspberry Pi must be powered on. This is done by simply connecting the power cable to the Raspberry Pi. If a red

light flashes on, it means the Pi is receiving power but is turned off. If this happens, press the power button once, and the LED light should go out. If the light doesn't turn on but the power cable shows a green light, then the Pi is powered on.

4.2.1 Micro-HDMI method

The most convenient method is to plug a micro-HDMI cable from one of the Pi's micro-HDMI ports into a monitor that accepts HDMI input. This allows direct visualization of the desktop GUI included with the OS. A keyboard and mouse are also needed to navigate the desktop, which is trivial to set up since the Pi has several USB ports on the back. A visual setup of this is shown in Fig. 4.2, which shows this method in action, with a finger pointing to the plugged-in gray micro-HDMI cable connected to the Pi. The monitor shown is a direct feed taken from the Pi.

The pros of this method are that it offers the least latency for accessing and editing software on the Pi. This is great for testing onboard software; however, the major con is that it severely limits the portability of the camera-lidar setup. The bottleneck in portability is determined by the length of the micro-HDMI cable, which makes it impractical to pick up the subsystem and carry it around. Additionally, powering the micro-HDMI cable may cause power instability if the Pi is powered by a portable battery.

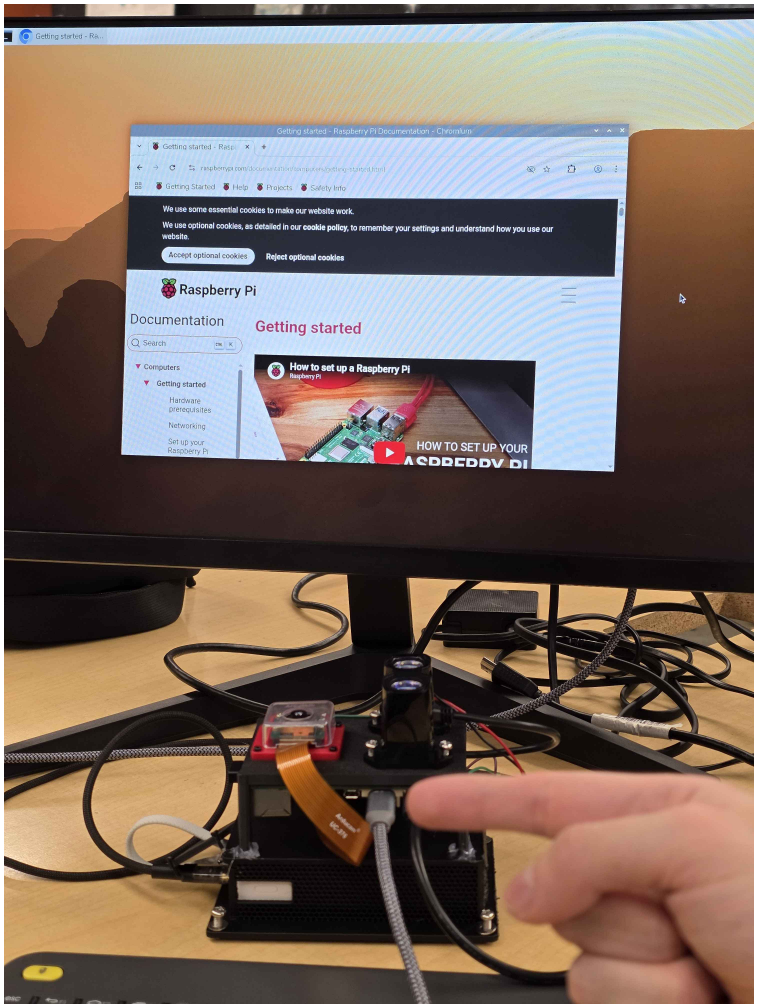


Figure 4.2 – Visual demonstration of the micro-HDMI method of accessing the Pi.

4.2.2 Raspberry Pi Connect - Desktop Streaming

The second method involves streaming the desktop to a computer, which removes the need for a physical micro-HDMI cable. This process involves linking an email account to the Raspberry Pi Connect portal at connect.raspberrypi.com. There is extensive documentation that outlines a step-by-step process for setting up the streaming feature [50]. The short explanation is that some setup is needed on the physical Pi, which initially involves using the micro-HDMI port and starting the Raspberry Pi Connect setup in the top-right corner of the desktop OS. Then, there is a process for linking the email account to the device. Once it is set up, the website portal can be accessed as shown in Fig. 4.3, where the device and its status are displayed. If the setup is done properly, the Pi

will automatically connect to the portal every time it boots and await a remote streaming connection from a computer. Once the screen-sharing option is selected, a new tab opens, and a live stream of the Pi's display is shown, as seen in Fig. 4.4. The downside is that it can be somewhat more resource-intensive, and if the feed goes down, it will interrupt the user experience. Ideally, this is used for debugging. Note, there is an existing username and password setup, but to protect the login information, contact professor Luna or an ARES team member for the login credentials.

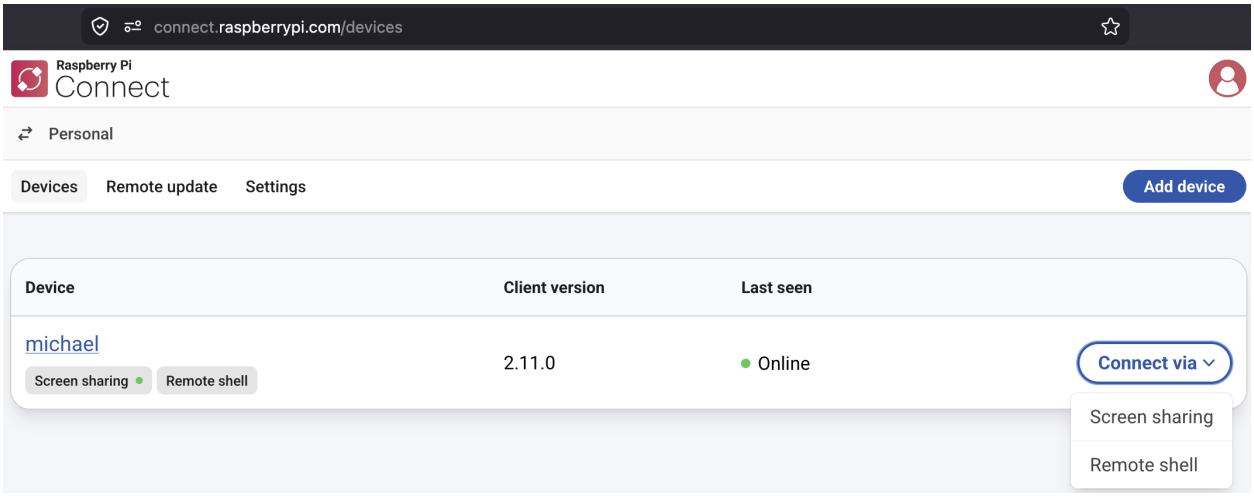


Figure 4.3 – User interface of the Raspberry Connect portal.

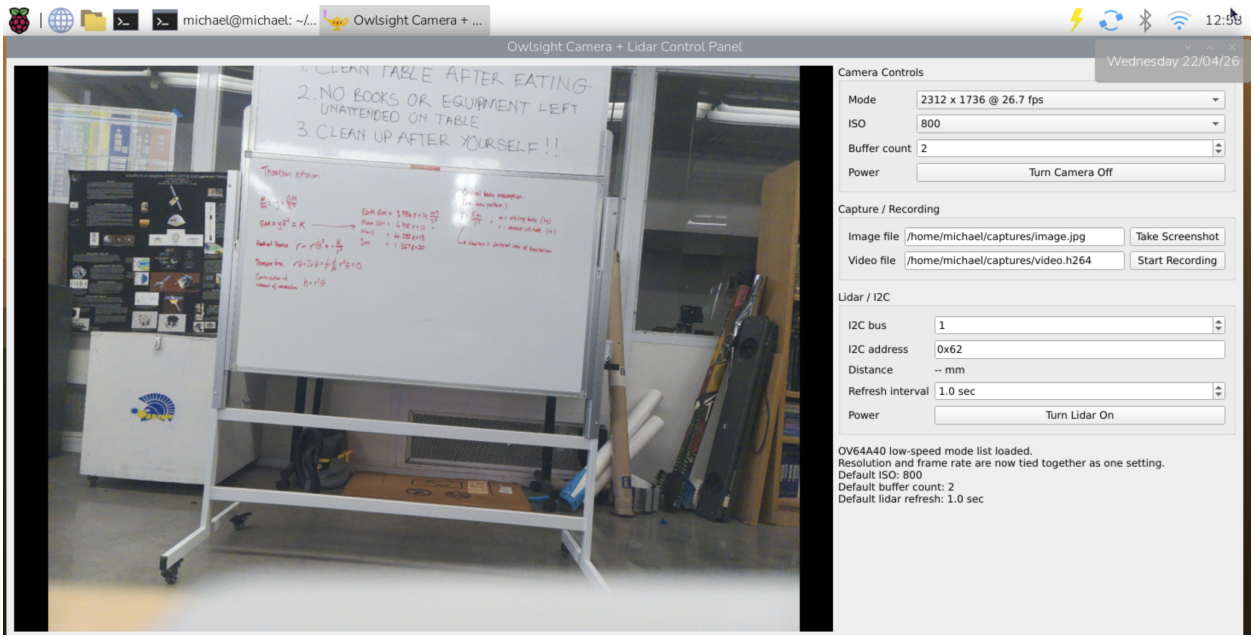


Figure 4.4 – Streaming feed of the Raspberry Pi with the camera-lidar application running.

4.2.3 Raspberry Pi Connect - Remote Shell

The last option is to use the remote shell, which follows the same setup as the previous subsection; however, the *remote shell* option is chosen instead. The major advantage over using this option is that the OS performance overhead is reduced, since it does not need to render a full desktop. Instead, the entire OS exists within a terminal. This is the optimal environment to launch drone missions using the camera-lidar subsystem. The obvious downside is that there is no live feed from the camera-lidar, and specialized code is needed that includes several print statements to figure out what is happening in the terminal. Fig. 4.5 shows what the remote shell looks like once it's working, along with the GUI-less camera-lidar code in action.

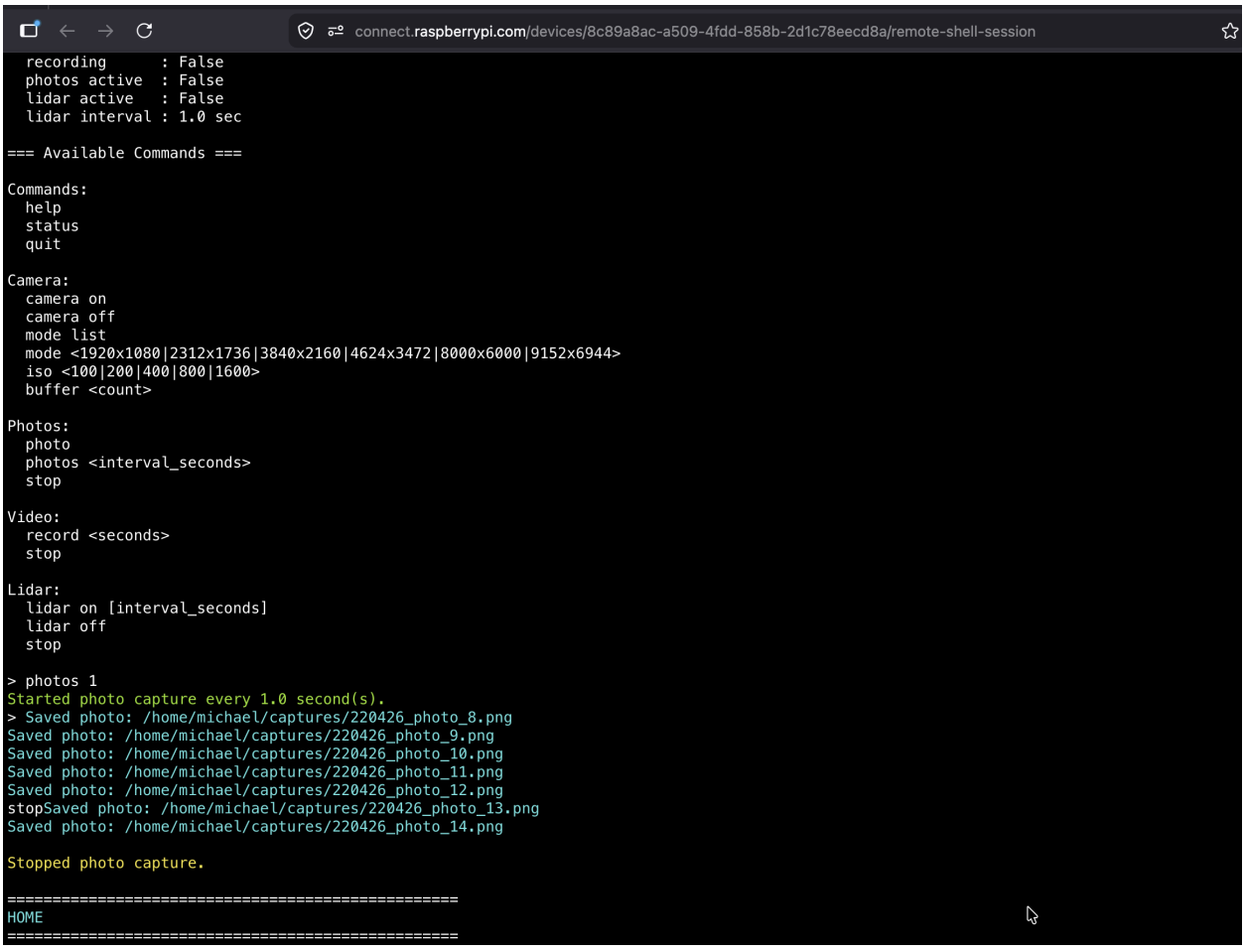


Figure 4.5 – Remote shell feed from the Raspberry Pi connect portal.

4.3 Camera-Lidar Software

4.3.1 GUI-based Software

The software graphical user interface (GUI) is developed in parallel with the physical prototyping of the camera-lidar system. One of the benefits of parallel development is the ability to experiment with several parameter settings, such as camera resolution, frame rate, autofocus, and contrast/saturation/hue settings. The GUI acts as the main interface between software and hardware. This makes obtaining information significantly easier, since, in the case of lidar, the value of the subsystem above ground is measured and displayed to the user. It should be noted that the GUI is meant for high-level visualization, experimentation, and prototyping mission concept ideas. In reality, when the subsystem is attached to a UAV in flight, accessing a GUI is presumed to be impossible because an HDMI cable must remain connected from the Pi to a computer. The GUI is used to test the functionality of the camera, lidar, and data collection to ensure that they all function properly.

To launch the GUI-based software, either the Micro-HDMI or the Desktop Streaming method can be used. The terminal/command line prompt must be opened, and the following commands must be entered as seen in Fig. 4.6. The code for the camera-lidar system, written in Python, is stored in the *code* folder. The command *cd code/* navigates the file explorer to that folder. The command *source venv/bin/activate* loads the installed packages and libraries needed to run the camera-lidar code. The last command, *python3 camera_lidar.py*, runs the GUI-based camera-lidar code.

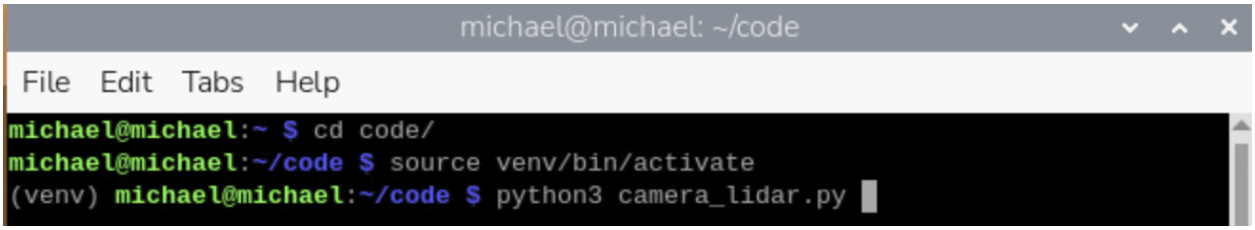


Figure 4.6 – Visual guide on launching the GUI-based camera-lidar software.

Revisiting an earlier figure as shown in Fig. 4.7, the software environment includes a video feed from the camera in the left portion of the UI, and various settings on the right side. The main

idea with the GUI is to visually test the functionality of the camera and lidar sensors, to determine functionality and optimal settings. Since OwlSight supports various resolutions and associated frame rates, a visual observation study was conducted. It was determined that the 1920x1080p 45fps setting for the camera is optimal, providing more than sufficient visual quality and clarity. Higher resolutions are available, but they appear more choppy because they run at a lower framerate and are more CPU-intensive. For the ISO setting, 800 provides a good brightness, with other values not making much difference in visual quality. Buffer count is set to 2, and it works well. Increasing the buffer count slightly improves image stability, but it also increases CPU usage, which may not be worth it. The only setting that affects lidar is the refresh interval, which defaults to 1 second. The 1-second interval works well, and it automatically records a timestamp of each data point taken and saves it to a CSV file. The option to record and take a screenshot are also present and functional, with the filepath and filename being adjustable.

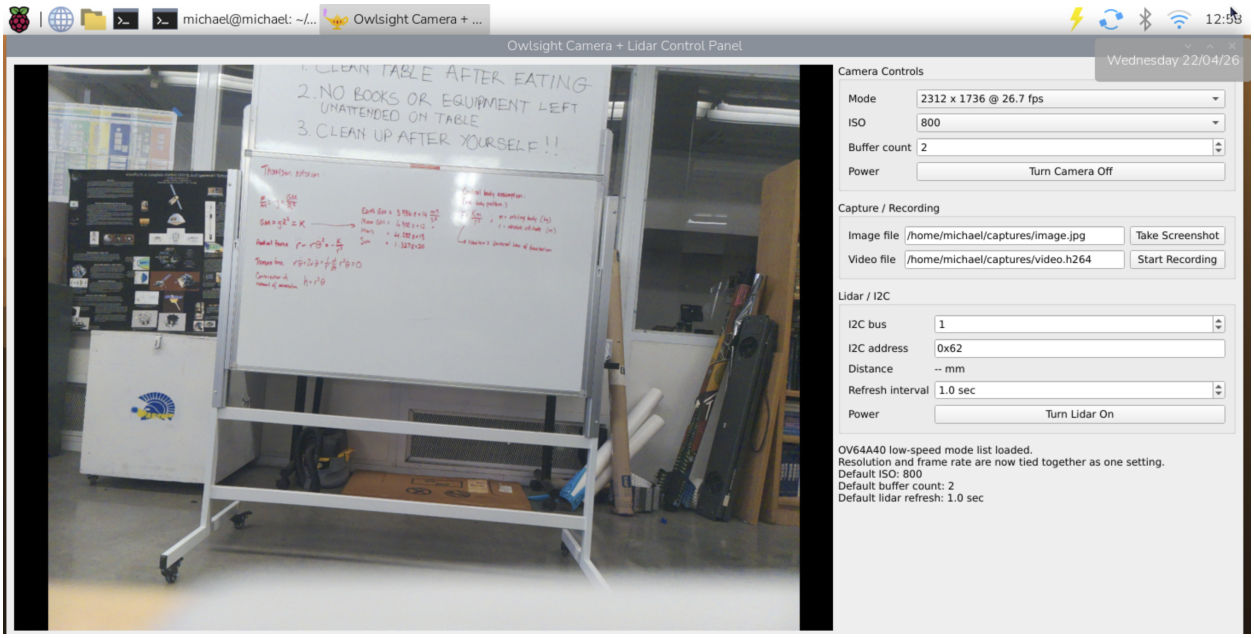


Figure 4.7 – Streaming feed of the Raspberry Pi with the camera-lidar application running.

4.3.2 Command Line Software

For practical fieldwork applications, the current approach is to connect an external computer to the Raspberry Pi using the Raspberry Pi Connect application as discussed earlier. From there,

the Raspberry Pi and the camera-lidar can be accessed remotely via the command line. It retains the exact same functionality as the Software GUI, but all the functionality has been converted into command-line arguments. Features such as recording video, obtaining lidar values, changing resolution, and others can be changed via the command line or by modifying the GUI-less script (run via the command line). Similar to how the GUI-based software is launched, Fig. 4.8 shows how to launch the GUI-less software, with the only main difference being that the software is launched via the line: *python3 nogui_camera_lidar.py*.

```
michael@michael:~ $ cd code/
michael@michael:~/code $ source venv/bin/activate
(venv) michael@michael:~/code $ python3 nogui_camera_lidar.py
```

Figure 4.8 – Setup to launch the GUI-less version of the camera-lidar software.

Once the software is launched, the GUI-less camera-lidar software home page is displayed. The user is greeted with a series of commands that can be launched, as seen in Fig. 4.9. The top of the home page shows the current status settings for the camera-lidar system, which can be manually changed using the available options and their commands shown below in *Available Commands*. Under *Camera*, there are a series of commands that can change the resolution, ISO setting, and buffer count. It can also be used to see if the camera turns on or off for debugging purposes. Once the camera settings have been adjusted, the commands under *Photos* can tell the camera to take a photo or a series of photos, with an input parameter specifying the interval, in seconds, at which the camera should take a picture. Under Video, there is a command for recording, along with an input parameter specifying the video's duration in seconds. Under *Lidar*, there is a setting called *lidar on* along with an input parameter that determines the interval for when the lidar takes a measurement in seconds. *Photos*, *Video*, and *Lidar* all include a command called *stop*, which can stop any launched command early and return to the home page. For all collected data, a file is automatically generated and stored. The process for obtaining the data is detailed in a later subsection.

```
================================================
HOME
================================================
Status:
  camera running : False
  mode           : 1920x1080 @ 45.0 fps
  iso            : 800
  buffer count   : 2
  recording      : False
  photos active  : False
  lidar active   : False
  lidar interval : 1.0 sec

=== Available Commands ===

Commands:
  help
  status
  quit

Camera:
  camera on
  camera off
  mode list
  mode <1920x1080|2312x1736|3840x2160|4624x3472|8000x6000|9152x6944>
  iso <100|200|400|800|1600>
  buffer <count>

Photos:
  photo
  photos <interval_seconds>
  stop

Video:
  record <seconds>
  stop

Lidar:
  lidar on [interval_seconds]
  lidar off
  stop
```

Figure 4.9 – Home page of the GUI-less camera-lidar software.

4.3.3 Data Retrieval

Once data has been generated, whether through GUI-based or GUI-less software, it can be gathered via SSH. While there are several options for approaching this, the easiest approach I have found is to use Visual Studio Code. To begin, install the following extensions as shown in Fig. 4.10.

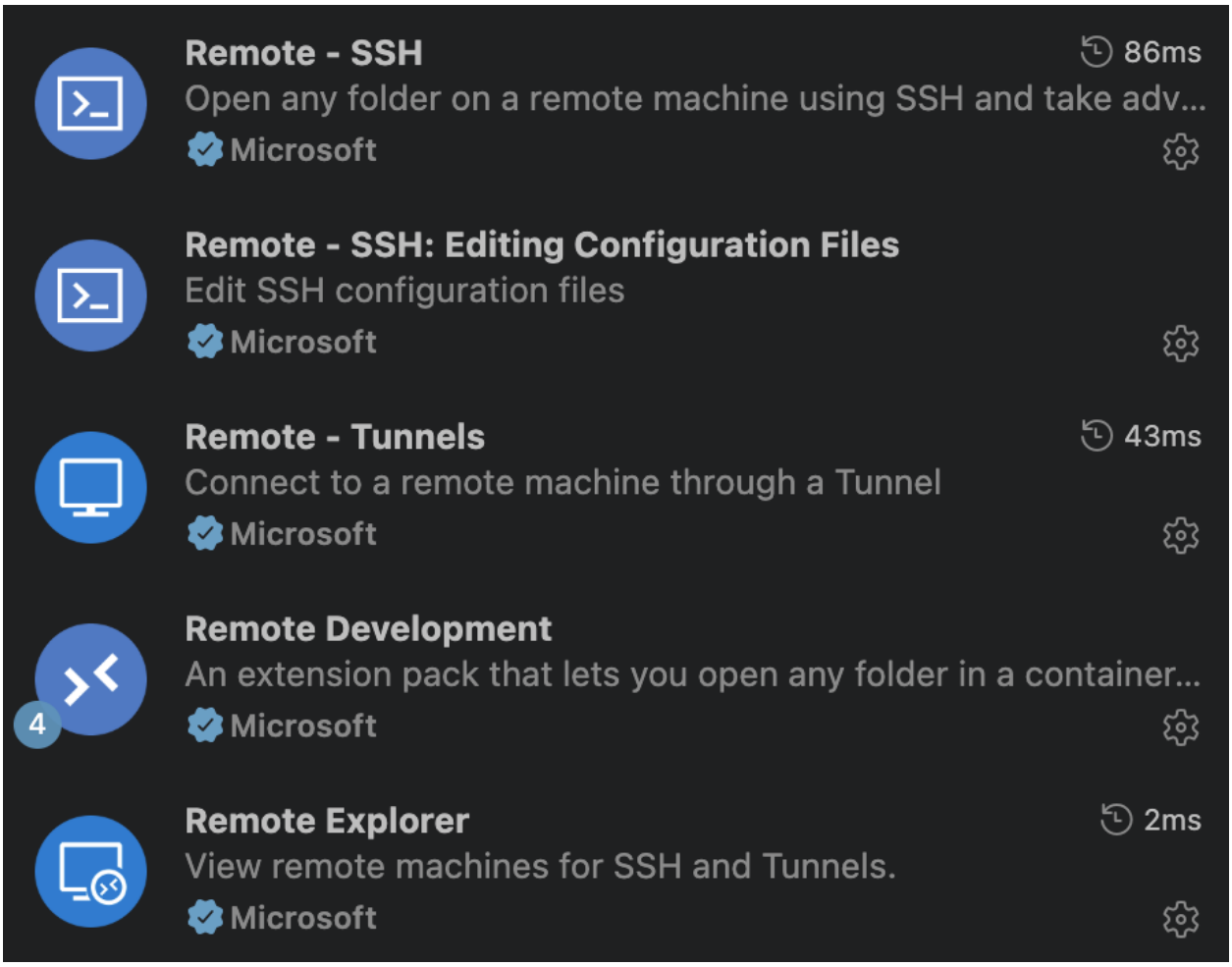


Figure 4.10 – SSH extensions in Visual Studio for data retrieval.

Next, access the Raspberry Pi terminal using any of the previously mentioned 3 methods, and type *hostname -I* to get the Pi's IP address, as shown in Fig. 4.11. Then navigate to the control center ¿ interfaces and check whether SSH is enabled; if not, enable it, as shown in Fig. **??**.

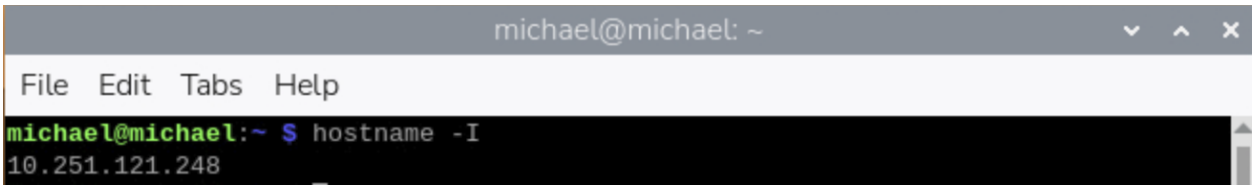


Figure 4.11 – Hostname command in the terminal.

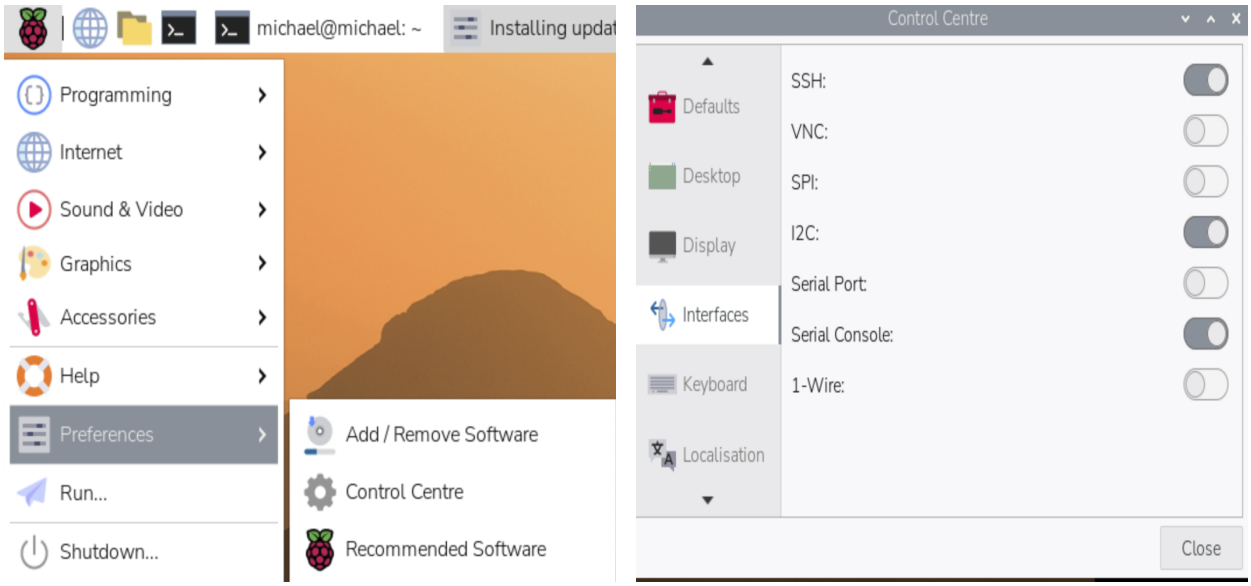


Figure 4.12 – (Left) Control Center from Desktop GUI (Right) Interface Settings.

Once that is done, open VS Code on a computer of your choice, and as shown in Fig. 4.13, there is an SSH icon on the left taskbar. Click on that, then click on the *plus sign* to the *right of a cogwheel* on the row that says *SSH*. Then, as seen in the figure, at the top of VS Code, there will be an entry to enter a *SSH command*. From there, enter ssh (name of Raspberry Pi)@(IP obtained from hostname -I). Once complete, hit enter, and select the option that says *.../.ssh/config*. Once that is entered, under SSH in remote explorer, the Raspberry Pi's IP address will appear, along with a right-facing arrow. *Click on the right arrow* to connect to the Pi, as seen in Fig. 4.14.

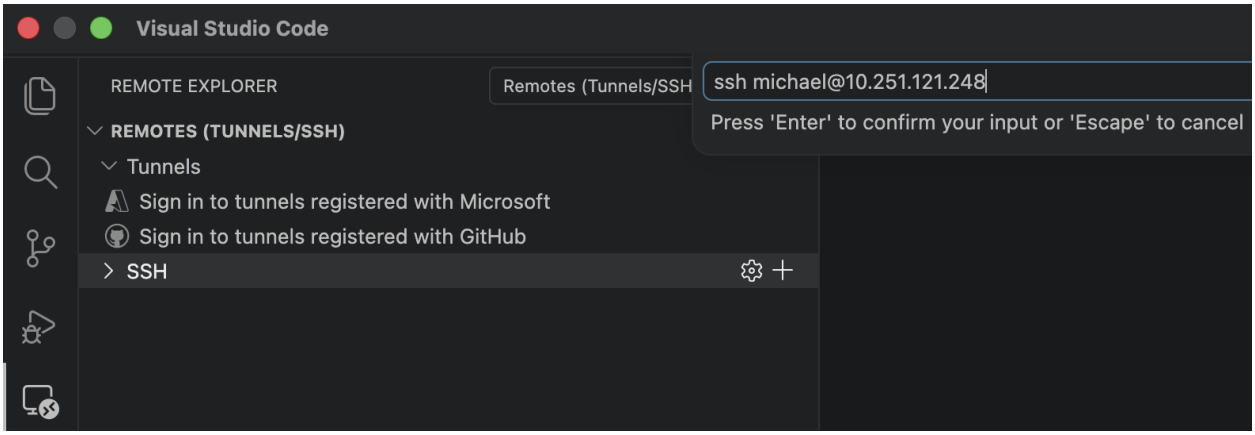


Figure 4.13 – VS Code SSH connection command inputs.

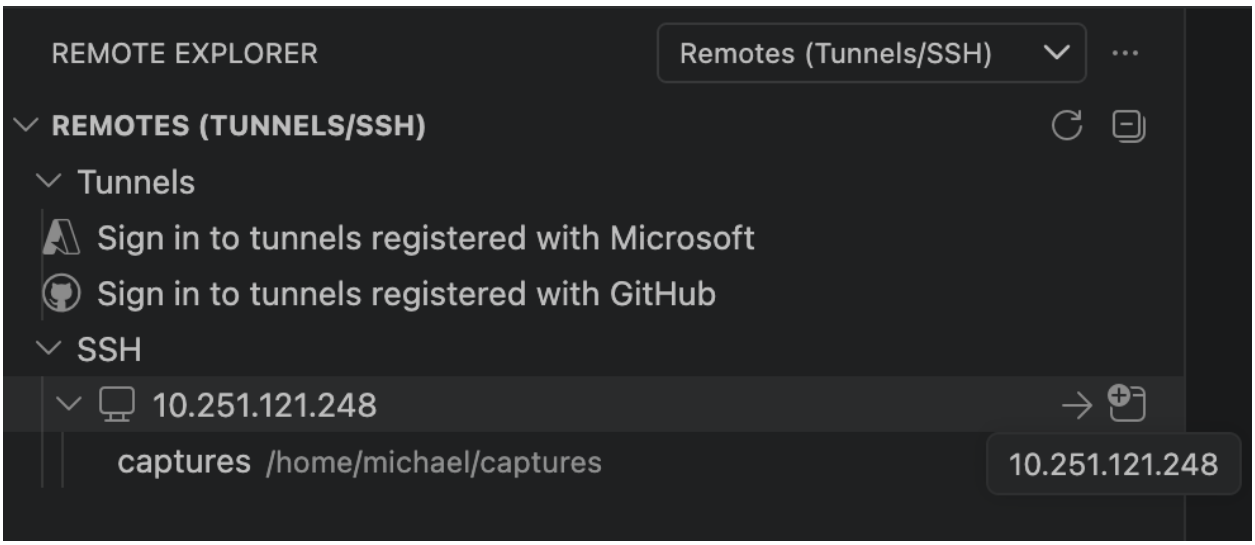


Figure 4.14 – SSH connect to Pi option.

Once the arrow has been pressed, a prompt will appear to enter your password at the top of the screen, as shown in Fig. 4.15. Enter the password set for the Pi (the default is *password*). Once set, a new page opens, and an SSH connection to the Pi is established. To access the data, click on the blue Open Folder button on the left (seen in Fig. 4.16). Then a list of items is shown; by default, the camera-lidar data is saved to a folder called *captures*. Click on that button, as seen in Fig. 4.17. Then, all the data captured by the camera-lidar is visible in VS Code, as shown in Fig. 4.18. To download data, right-click a file and select Download, as shown in Fig. 4.19. This creates an easy process for portable file transfers from the Raspberry Pi to any personal computer.

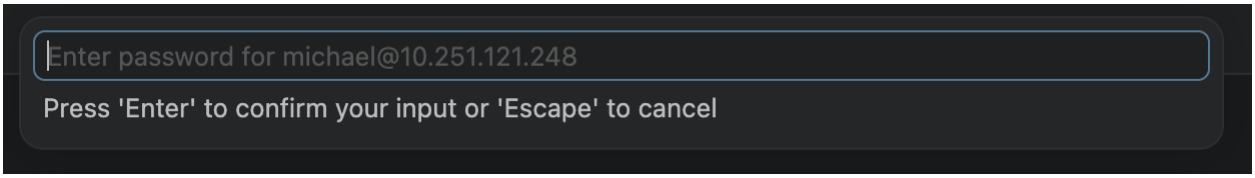


Figure 4.15 – VS Code SSH connection password prompt.

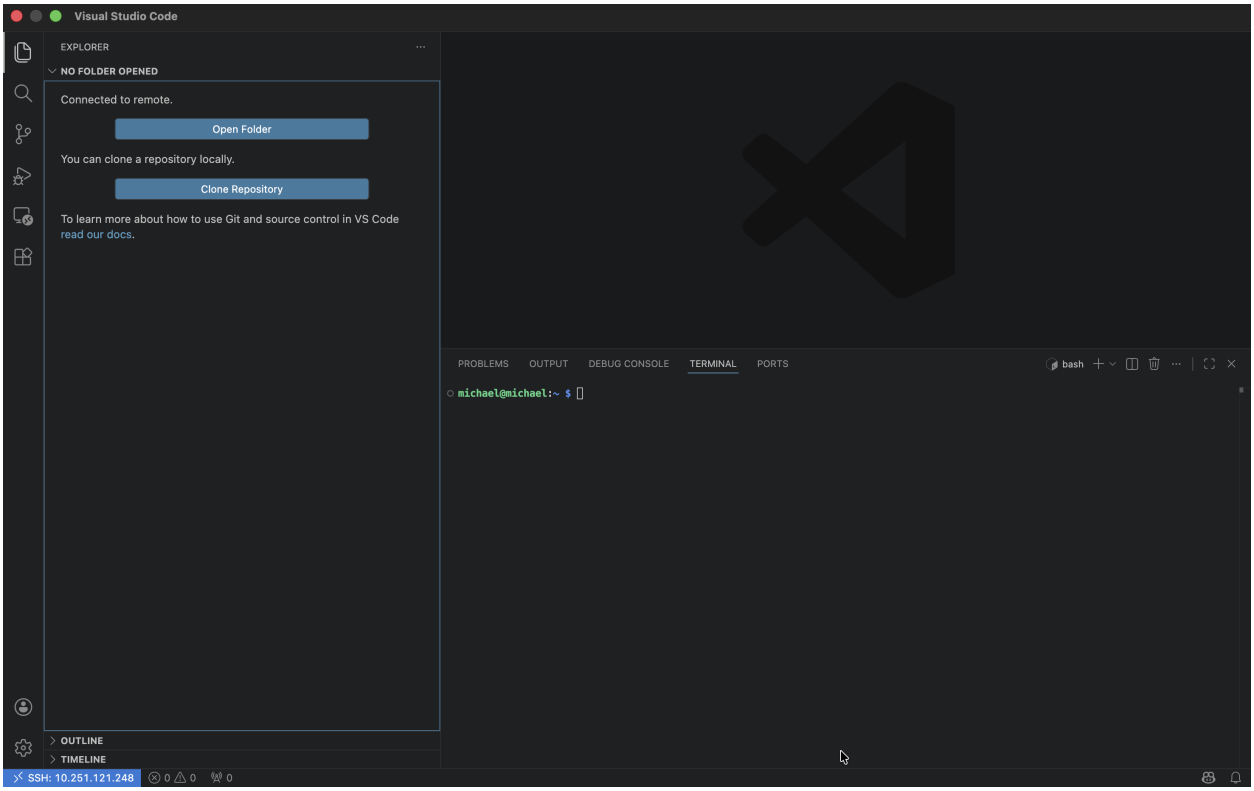


Figure 4.16 – VS Code SSH open folder taskbar.

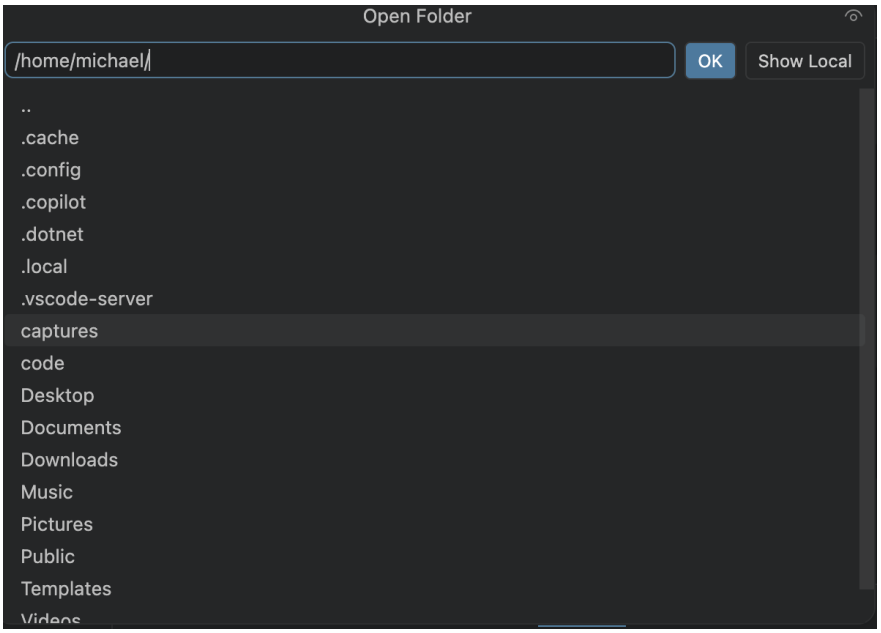


Figure 4.17 – VS Code SSH open folder UI.

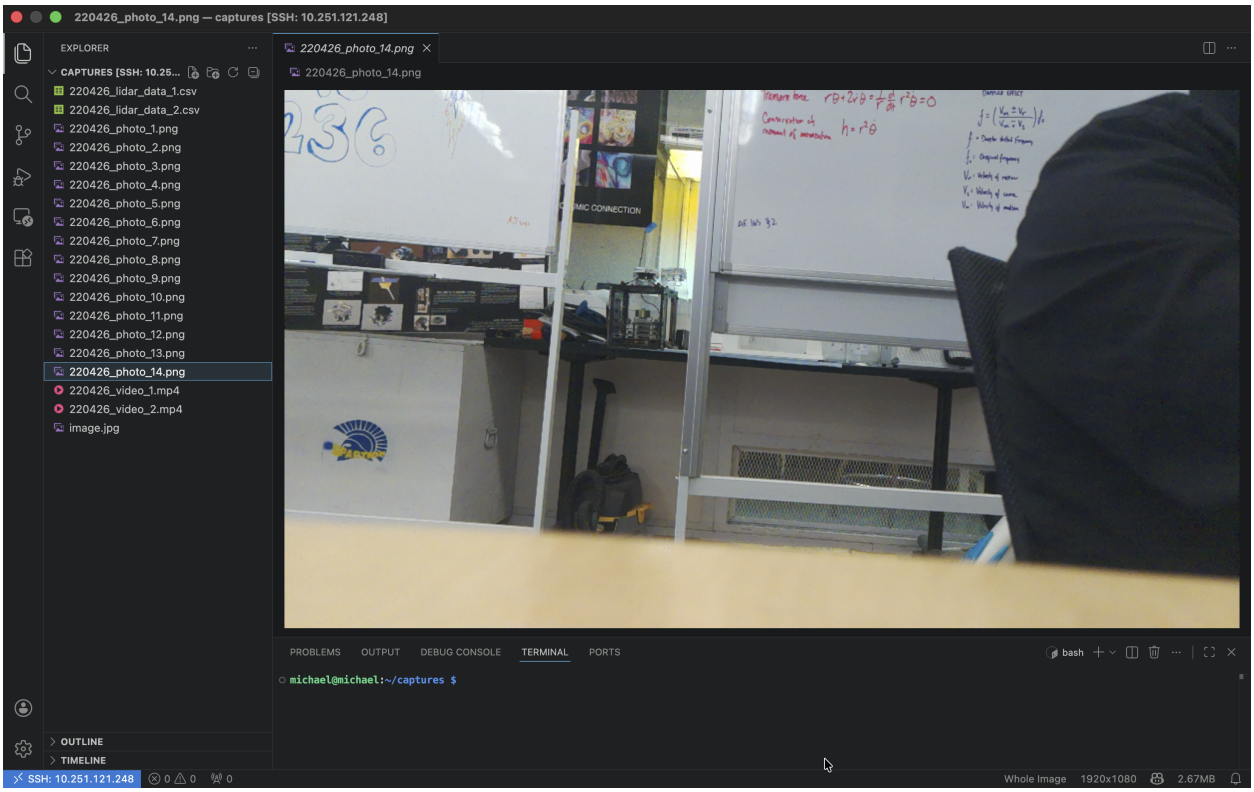


Figure 4.18 – VS Code SSH data location and preview.

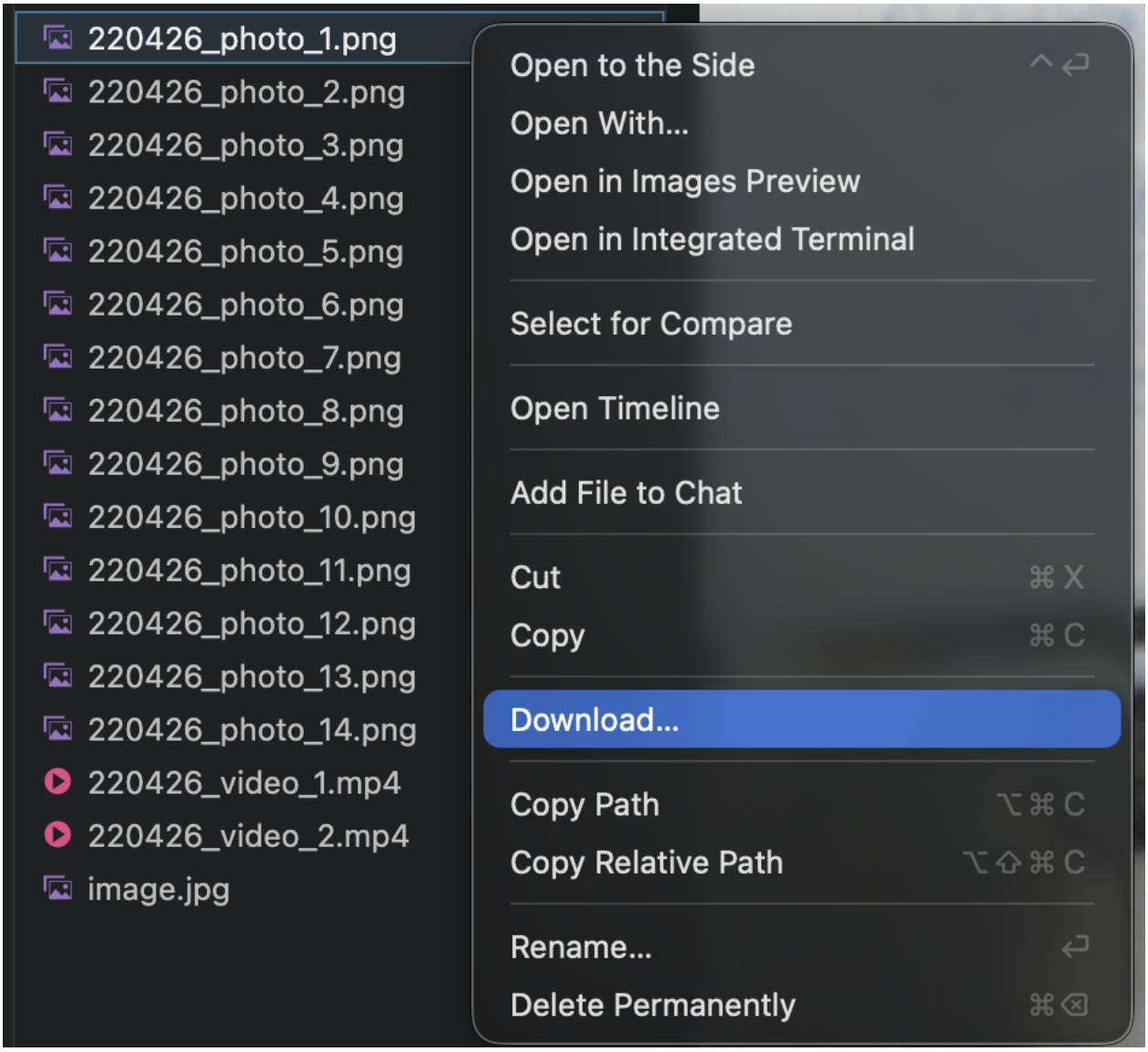


Figure 4.19 – Downloading a file from the camera-lidar subsystem in VS Code.

5. Subsystem Testing

5.0.1 Camera Testing

For the camera testing, getting images and video feed is rather straightforward. Fig. 5.1 showcases several direct image captures of various objects and scenes taken from the camera-lidar system, while fully powered by the portable battery. The photos were taken at the default 1920x1080p resolution. To demonstrate the system's capability, Fig. 5.2 shows a photo of the camera module, which also shows the desktop point of view.



Figure 5.1 – Sample images taken with the Camera module in 1920x1080p setting.

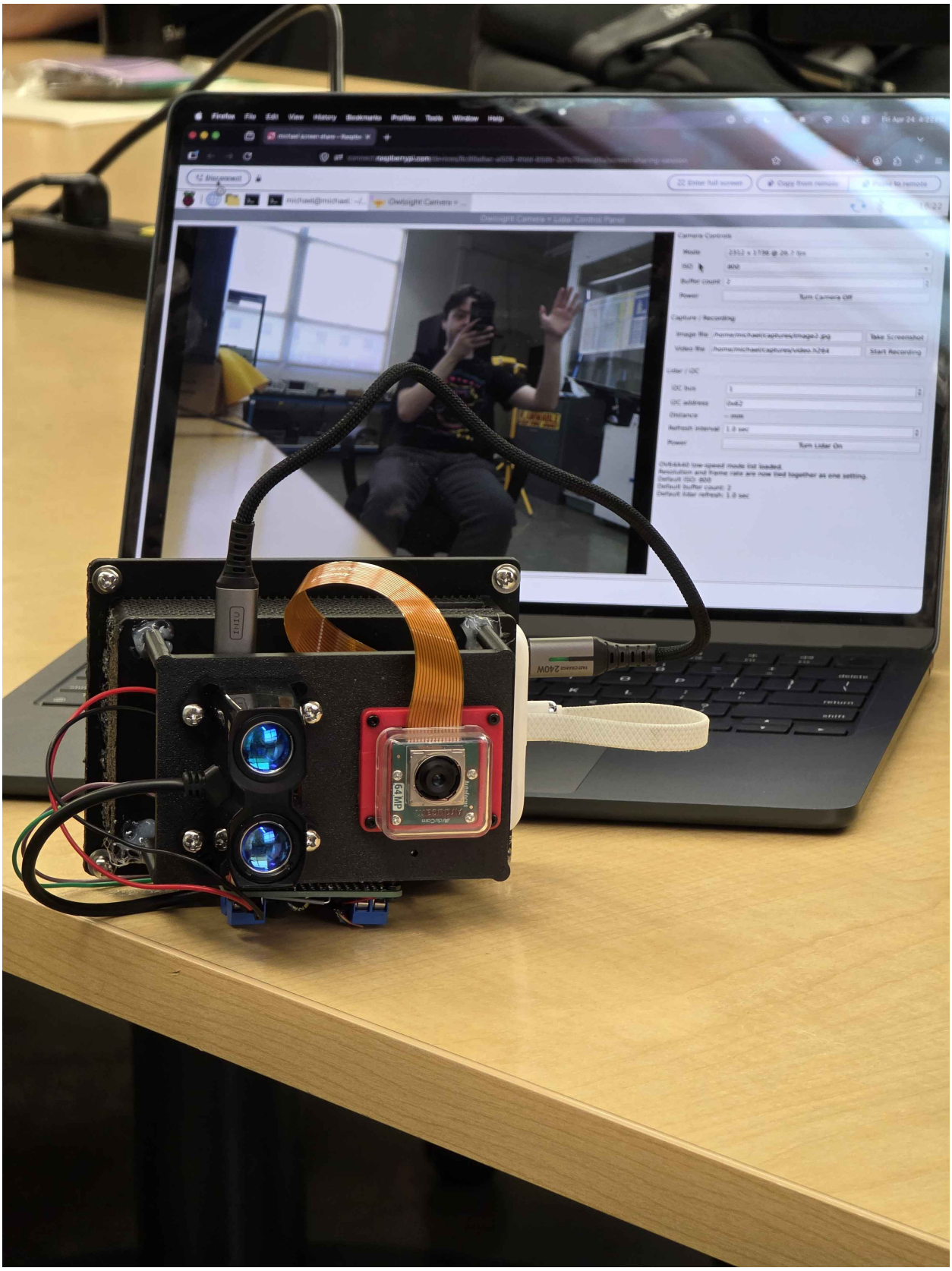


Figure 5.2 – Proof of functionality of the camera module.

5.0.2 Remote Access Testing

The remote access testing demonstrates the remote desktop capabilities for accessing the camera module from long distances. For this experiment, the camera-lidar subsystem was set in a stationary position in the AE 236 lab in the SJSU Engineering Building, as seen in Fig. 5.3. The system is fully standalone, powered by a portable battery bank, and connected to the internet.

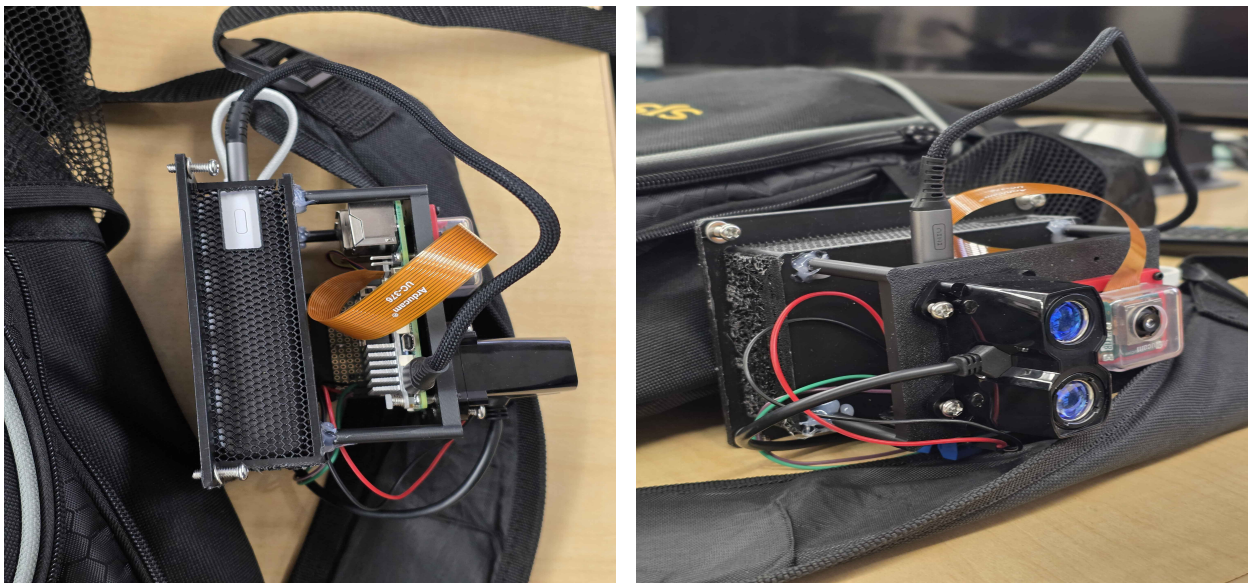


Figure 5.3 – (Left) Top view of a stationary camera-lidar system. (Right) Front-view of a stationary camera-lidar system.

The goal of this test is to demonstrate the portability of accessing the camera-lidar system. The subsystem is pointed at a whiteboard in an active room in AE236, within the SJSU engineering building, as seen in Fig. 5.4. As shown, both the camera feed and the lidar distance are displayed in real-time. The intriguing part is that the screenshot shown in that figure is taken in a completely different area from the stationary feed. As seen in Fig. 5.5, the feed was taken in the aerospace engineering office on the second floor of the SJSU engineering building, about 50ft away from the stationary setup.

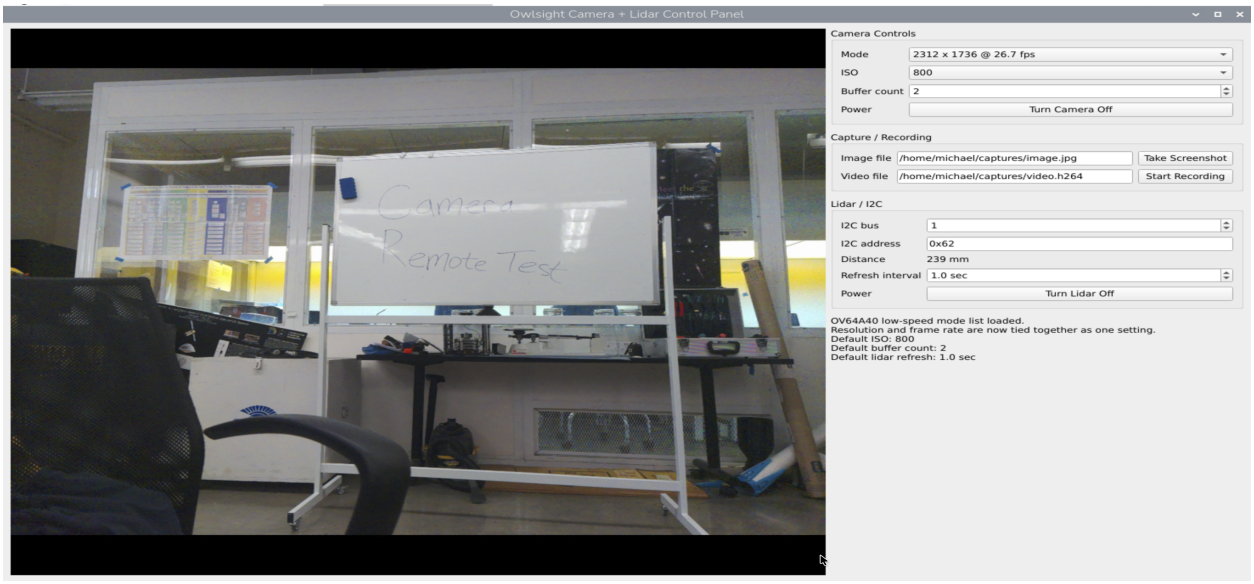


Figure 5.4 – Video feed of the camera module in the stationary setup.

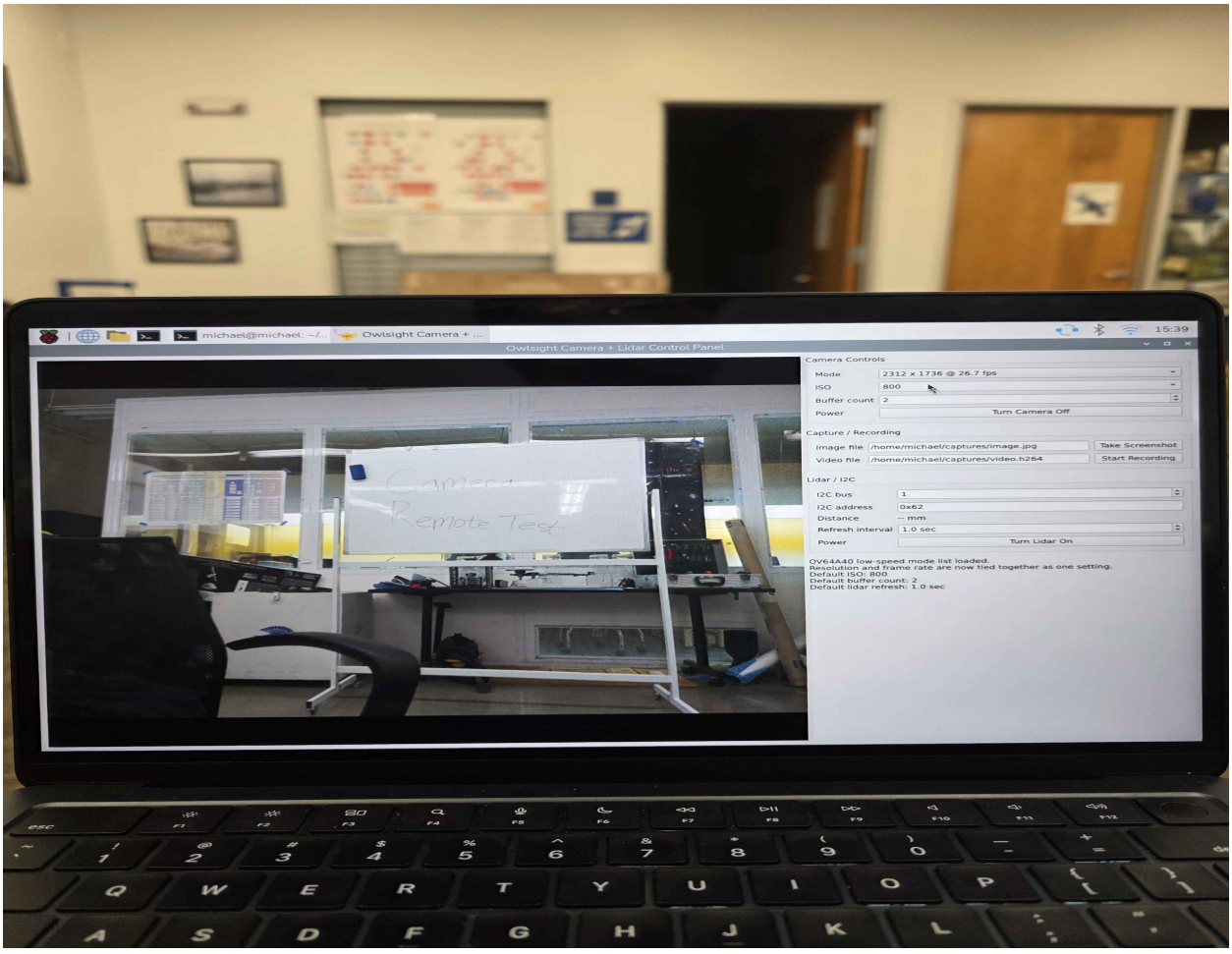


Figure 5.5 – Photo of the video feed taken from the SJSU Aerospace office.

Another test was conducted on the 8th floor of the Interdisciplinary Science building at SJSU, which uses a different Wi-Fi configuration than the Raspberry Pi. As shown in Fig. 5.6, a real-time video feed was established despite being hundreds of feet away from the stationary setup. To prove that it is a live feed, a colleague was instructed to walk in front of the camera, as shown in Fig. 5.7. This demonstrates the feasibility of accessing the standalone camera-lidar system from anywhere, provided an internet connection is available.

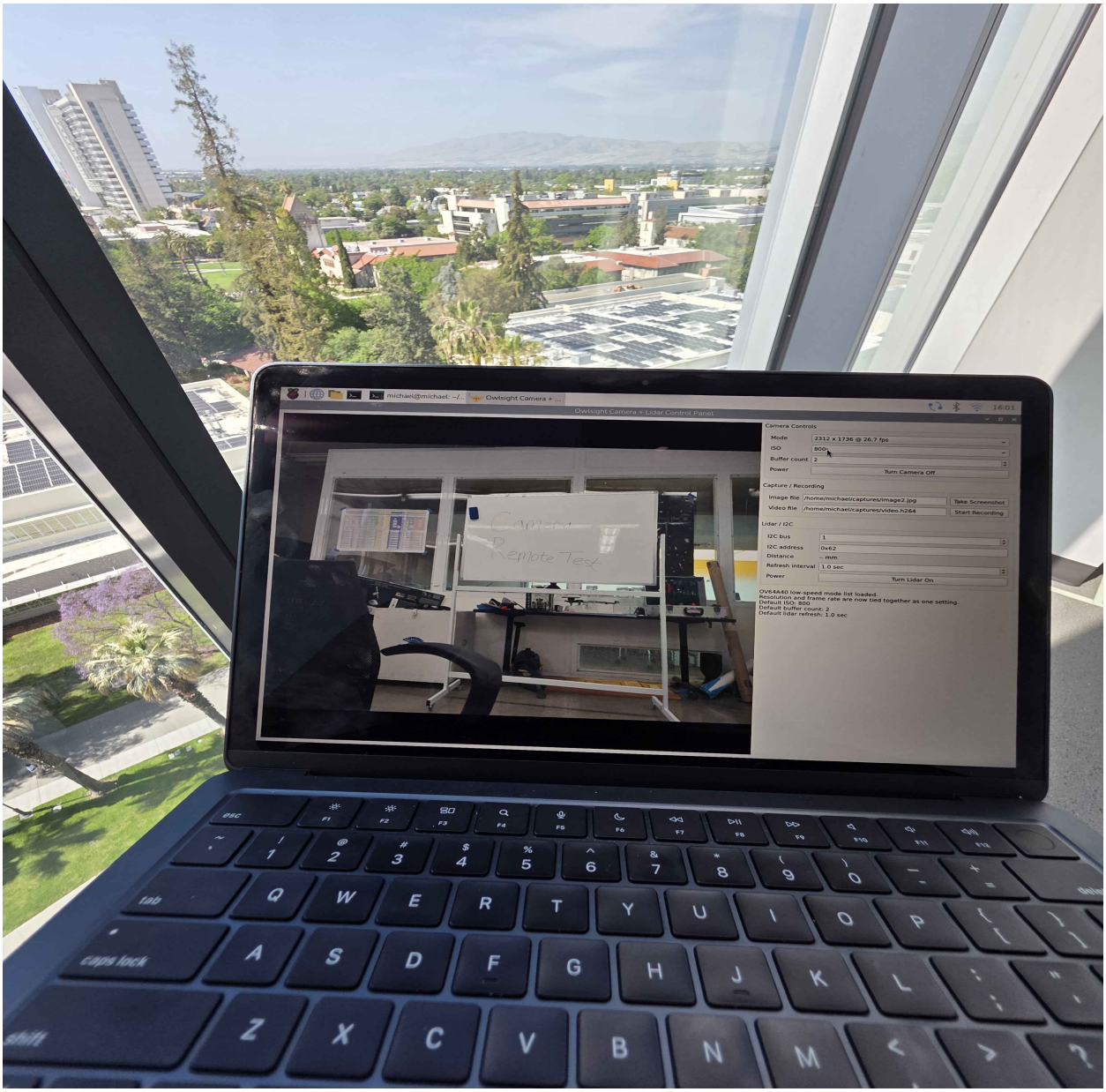


Figure 5.6 – Photo of the 8th floor science building with the camera live feed.



Figure 5.7 – Visual demonstration of accessing the camera live feed from a remote setting.

5.0.3 Lidar Testing

The lidar system underwent a manual verification with a measuring tape to ensure that the data is accurate. Once the calibration was completed, a brief experiment was conducted by placing a hand in front of the lidar sensor. As seen in Fig. 5.8, some data were obtained that capture the lidar's measurement range as a hand slowly backs away from it. The data was captured at one-second intervals, with a timestamp in the first column and the data in millimeters in the second column. The calibration and testing verify that the lidar data is accurate, functional, and easily accessible in both the GUI-based and GUI-less software programs.

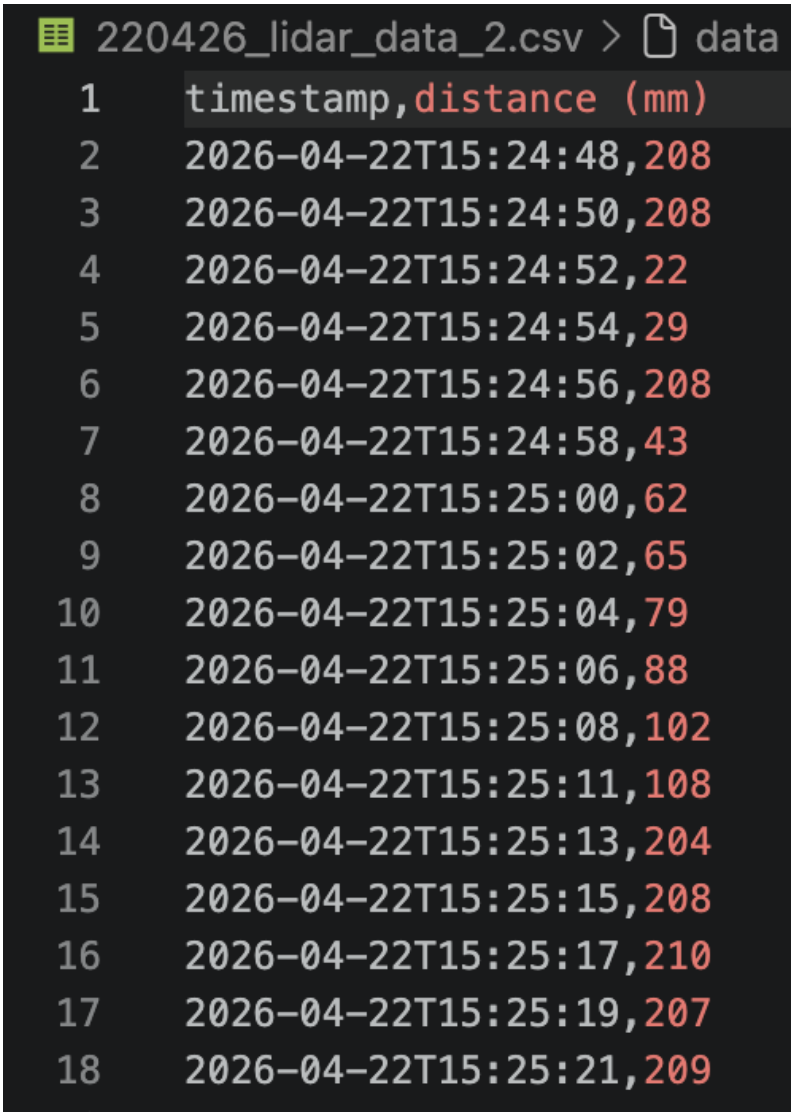


Figure 5.8 – Lidar CSV data file.

5.0.4 Battery Testing

A battery test with the current powerbank configuration installed on the subsystem has been conducted. The way that this is conducted is rather straightforward. It uses remote desktop streaming, with both the camera and lidar powered on. The camera is set to the 2312x1736p resolution at 26.7 fps, an ISO of 800, and a buffer count of 2. This configuration is running the subsystem at full power to stress test the battery. The initial test has the powerbank battery at 100%, and the time recorded to reduce the overall system's size and weight. Table 5.1 shows the overall experiment, with the left column displaying the charge levels, and the right column displaying the duration of battery life. It is set up this way because power consumption is not linear, but rather a curve that accelerates as the battery becomes lower. The general observation is that the battery provides more than enough power to last roughly 3.5 hours, far more than the expectation of 30 minutes of recording footage. This provides a justification for using a smaller battery bank to reduce weight.

Table 5.1 – Initial battery test of camera-lidar system at full power with the Anker Zulu power bank

Battery range (Start % - End %)	Time
100% - 80%	1 hour & 10 minutes
80% - 60%	55 minutes
60% - 40%	40 minutes
40% - 20%	30 minutes
20% - 0%	20 minutes
Estimated battery life time	215 minutes

6. Conclusion and Future Work

6.1 Conclusion

The foundation for the camera-lidar system is solid, as it meets the system's mission requirements. It is a self-contained unit that can remotely receive input commands to record video and lidar measurements. Although the operation time is 30 minutes, it currently runs for over 3.5 hours, suggesting that further refinements can reduce the size and weight of the overall system. The current weight comes to approximately 400 grams, which can be reduced with a custom battery. As of writing, the payload mounting unit is undergoing a redesign for the Project ARES drone, so drone testing is not currently feasible at this point; however, it is anticipated that over the summer, the work will be completed, and that it will be sized to fit the dimensions of the mounting holes provided in earlier engineering drawings in this paper.

6.2 Future Work

While the foundation for the camera-lidar is solid, there are many tasks that can still be done to improve the overall system. The first step is to perform a vibration test using the motors installed on the Project ARES drone. The drone itself does not necessarily need to take off; it can be held in a stationary position while the motors are operational. Some projects described in the literature review use gimbal systems to stabilize the camera. The current iteration has no gimbap at all; the camera is highly prone to vibration. The main thing to investigate is whether the vibrations affect the camera image quality and the lidar measurement sensor's capabilities. The second would be to verify the payload integrity, especially at the four M4 screw holes, and whether it is structurally secured or prone to falling out. The third would be to do a structural analysis on all the screws that secure the devices together. Currently, all the screws are threaded into PLA, which has worked in the short time it has been tested, but for long-term sustainability, there may be concerns about material wear or cracking. In terms of additional features for the camera-lidar system, the most immediate benefit is to develop a custom battery system or select a lighter one that can also reliably output 30W. While

the current battery installed works well, it currently powers the camera-lidar system for 5+ hours. Earlier in the paper, there is a whole section dedicated to using Samsung battery cells and a voltage converter to assemble a custom battery unit. If there are any undervolting issues, the lidar could be powered by an external power source, perhaps the same one as the Project ARES drone, since it only needs a 5V and ground connection. This leads to another point where there needs to be some sort of integration between the flight controller on the Project ARES drone and the camera-lidar system. The data obtained from the camera-lidar system is coherent, but if the lidar data were processed in real time with the ARES UAV drone flight software, a data pipeline would be required.

References

[1] Patiño, J., "UNAM achieves a 3D model of Mexico's Popocatépetl volcano with the help of AI," *EL PAÍS English*, 2025. https://english.elpais.com/science-tech/2025-07-30/unam-achieves-a-3d-model-of-mexicos-popocatepetl-volcano-with-the-help-of-ai.html.

[2] Espinasa-Perena, R., and Pozzo, A., "Morphostratigraphic evolution of Popocatépetl volcano, México," 2006, pp. 115–137. https://doi.org/10.1130/2006.2402(05).

[3] Wanucha, G., "Flight path: Into the volcanic plume," *MIT News*, 2014. http://paocweb.mit.edu/about/paoc-spotlights/flight-path-into-the-volcanic-plume.

[4] Plunket, P., and Uruñuela, G., "Mountain of sustenance, mountain of destruction: The prehispanic experience with Popocatépetl Volcano," *Journal of Volcanology and Geothermal Research*, Vol. 170, No. 1, 2008, pp. 111–120. https://doi.org/10.1016/j.jvolgeores.2007.09.012.

[5] T-DRONES, "How Much Weight Can a Drone Carry? (2025 Complete Guide)," *T-DRONES Blog*, 2024. https://www.t-drones.com/blog/how-much-weight-can-a-drone-carry.html.

[6] Liu, V., "A Framework to Improve Weight Estimation and Manufacturing Accuracy for Large Student Design Teams," *AIAA Regional Student Conferences*, 2023. https://doi.org/10.2514/6.2023-69888.

[7] E38 Survey Solutions, "Drone LiDAR Price Guide," *E38 Survey Solutions*, 2025. https://e38surveysolutions.com/pages/drone-lidar-price-guide.

[8] Pilot Institute, "LiDAR Drones: The Best Models for Surveying, Mapping, and More," *Pilot Institute*, 2025. https://pilotinstitute.com/lidar-drones/.

[9] Zhang, Z., and Zhu, L., "A Review on Unmanned Aerial Vehicle Remote Sensing: Platforms, Sensors, Data Processing Methods, and Applications," *Drones*, Vol. 7, No. 6, 2023. https://doi.org/10.3390/drones7060398.

[10] Jaafar, H. H., and Sujud, L. H., "High resolution evapotranspiration from UAV multispectral thermal imagery: Validation and comparison with EC, Landsat, and fused S2-MODIS HSEB ET," *International Journal of Applied Earth Observation and Geoinformation*, Vol. 136, 2025, p. 104359. https://doi.org/10.1016/j.jag.2025.104359.

[11] Wu, M. G., Munishkin, A. A., Jung, J., and Xue, M., "Development of a Portable Airspace Management System for Wildland Fire Operations," *AIAA AVIATION FORUM AND ASCEND*, 2025. https://doi.org/10.2514/6.2025-3305.

[12] Smith, H., "NASA Demonstrates New Wildland Fire Airspace Management System," *NASA Ames Research Center*, 2025. https://www.nasa.gov/centers-and-facilities/ames/nasa-demonstrates-new-wildland-fire-airspace-management-system/.

[13] Ma, Y., Li, Q., Chu, L., Zhou, Y., and Xu, C., "Real-Time Detection and Spatial Localization of Insulators for UAV Inspection Based on Binocular Stereo Vision," *Remote Sensing*, Vol. 13, No. 2, 2021. https://doi.org/10.3390/rs13020230.

[14] Román, A., Navarro, G., Caballero, I., and Tovar-Sánchez, A., "High-spatial resolution UAV multispectral data complementing satellite imagery to characterize a chinstrap penguin colony ecosystem on deception island (Antarctica)," *GIScience & Remote Sensing*, Vol. 59, No. 1, 2022, pp. 1159–1176. https://doi.org/10.1080/15481603.2022.2101702.

[15] Hedelt, P., Reichardt, J., Lauermann, F., Weiß, B., Theys, N., Redondas, A., Barreto, A., Garcia, O., and Loyola, D., "Analysis of the long-range transport of the volcanic plume from the 2021 Tajogaite/Cumbre Vieja eruption to Europe using TROPOMI and ground-based measurements," *Atmospheric Chemistry and Physics*, Vol. 25, No. 2, 2025, pp. 1253–1272. https://doi.org/10.5194/acp-25-1253-2025.

[16] Manzo, M., Aiesi, G., Boselli, A., Consoli, S., Damiano, R., Di Donfrancesco, G., Saraceno, B., and Scollo, S., "Upgraded Three-Wavelength Lidar for Real-Time Observations of Volcanic Aerosol Optical and Microphysical Properties at Etna (Italy): Calibration Procedures and Measurement Tests," *Sensors*, Vol. 24, No. 6, 2024, p. 1762. https://doi.org/10.3390/s24061762.

[17] Canuto, M. A., Estrada-Belli, F., Garrison, T. G., Houston, S. D., Acuña, M. J., Kováč, M., Marken, D., Nondédéo, P., Auld-Thomas, L., et al., "Ancient lowland Maya complexity as revealed by airborne laser scanning of northern Guatemala," *Science*, Vol. 361, No. 6409, 2018, p. eaau0137. https://doi.org/10.1126/science.aau0137.

[18] Harris, D. L., Webster, J. M., Vila-Concejo, A., Duce, S., Leon, J. X., and Hacker, J., "Defining multi-scale surface roughness of a coral reef using a high-resolution LiDAR digital elevation model," *Geomorphology*, Vol. 439, 2023, p. 108852. https://doi.org/10.1016/j.geomorph.2023.108852.

[19] Xu, R., Li, C., and Bernardes, S., "Development and Testing of a UAV-Based Multi-Sensor System for Plant Phenotyping and Precision Agriculture," *Remote Sensing*, Vol. 13, No. 17, 2021. https://doi.org/10.3390/rs13173517.

[20] Qin, H., Zhou, W., Yao, Y., and Wang, W., "Individual tree segmentation and tree species classification in subtropical broadleaf forests using UAV-based LiDAR, hyperspectral, and ultrahigh-resolution RGB data," *Remote Sensing of Environment*, Vol. 280, 2022, p. 113143. https://doi.org/10.1016/j.rse.2022.113143.

[21] Raspberry Pi, "Raspberry Pi High Quality Camera," n.d. Accessed on 21 Oct. 2025. https://www.raspberrypi.com/products/raspberry-pi-high-quality-camera/.

[22] Adafruit Industries, "Raspberry Pi High Quality HQ Camera [12MP]: ID 4561," n.d. Accessed on 21 Oct. 2025. https://www.adafruit.com/product/4561?src=raspberrypi.

[23] Arducam, "64MP OwlSight (OV64A40)," n.d. Accessed on 23 Oct. 2025. https://docs.arducam.com/Raspberry-Pi-Camera/Native-camera/64MP-OV64A40/.

[24] Arducam, "1/1.32" 64MP Autofocus Camera Module for Raspberry Pi," n.d. Accessed on 23 Oct. 2025. https://www.arducam.com/arducam-1-1-32-64mp-autofocus-camera-module-for-raspebrry-pi.html.

[25] Advexure, "DJI Zenmuse L2 LiDAR Sensor for Matrice 400/350/300," n.d. Accessed on 25 Oct. 2025. https://advexure.com/products/dji-zenmuse-l2-lidar.

[26] GenPac Drones, "DJI Zenmuse L2," n.d. Accessed on 25 Oct. 2025. https://www.genpacdrones.com/product/dji-zenmuse-l2/.

[27] YellowScan, "2D vs 3D LiDAR: Comparison of Laser Scanning Tech," *YellowScan Knowledge*, 2024. https://www.yellowscan.com/knowledge/2d-vs-3d-lidar-a-comparison-of-two-laser-scanning-technologies/.

[28] MorpheusTEK, "2D vs 3D LiDAR: Key Differences & Applications," *mTEK Blog*, n.d. Accessed on 25 Oct. 2025. https://morpheustek.com/differences-between-2d-and-3d-lidar/.

[29] Garmin, "LIDAR-Lite v3HP — Distance Measurement Sensor," n.d. Accessed on 25 Oct. 2025. https://www.garmin.com/en-US/p/578152/.

[30] DJI, "Zenmuse L2 — Specifications," n.d. Accessed on 25 Oct. 2025. https://enterprise.dji.com/zenmuse-l2/specs.

[31] Raspberry Pi, "Raspberry Pi 5," n.d. Accessed 19 Oct. 2025. https://www.raspberrypi.com/products/raspberry-pi-5/.

[32] Arduino, "Arduino UNO Q," n.d. Accessed on 26 Oct. 2025. https://www.arduino.cc/product-uno-q/.

[33] Adafruit Industries, "Raspberry Pi 5 – 8 GB RAM, ID 5813," n.d. Accessed on 26 Oct. 2025. https://www.adafruit.com/product/5813?src=raspberrypi.

[34] Elliott Electronic Supply, "Raspberry Pi 5 8GB Ram, RPI5-4GB-SINGLE, Single Board Computer," n.d. Accessed on 26 Oct. 2025. https://www.elliottelectronicsupply.com/raspberry-pi-5-8gb.html.

[35] Drone Nerds, "Multispectral Cameras & Sensors – Enterprise," n.d. Accessed on 27 Oct. 2025. https://www.dronenerds.com/collections/cameras-sensors-enterprise-cameras-multispectral.

[36] Advexure, "Multispectral Drones & Sensors," n.d. Accessed on 27 Oct. 2025. https://advexure.com/collections/multispectral-drones-sensors.

[37] Barjaktarovic, M., Santoni, M., and Bruzzone, L., "Design and Verification of a Low-Cost Multispectral Camera for Precision Agriculture Application," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, Vol. 17, 2024, pp. 6945–6957. https://doi.org/10.1109/JSTARS.2024.3377104.

[38] Arducam, "1 MP × 4 Quadrascopic Camera Bundle Kit for Raspberry Pi & NVIDIA Jetson Nano/Xavier NX (Four OV9782 Global Shutter Color Camera Modules and Camarray HAT)," n.d. Accessed on 27 Oct. 2025. https://www.arducam.com/arducam-1mp4-quadrascopic-camera-bundle-kit-for-raspberry-pi-nvidia-jetson-nano-xavier-nx-four-ov9782-html.

[39] Pechlivani, E. M., Papadimitriou, A., Pemas, S., Giakoumoglou, N., and Tzovaras, D., "Low-Cost Hyperspectral Imaging Device for Portable Remote Sensing," *Instruments*, Vol. 7, No. 4, 2023, p. 32. https://doi.org/10.3390/instruments7040032, URL https://www.mdpi.com/2410-390X/7/4/32.

[40] DSLRPros, "Thermal & Infrared Cameras & Sensors," n.d. Accessed on 27 Oct. 2025. https://www.dslrpros.com/collections/thermal-infrared.

[41] FLIR Systems, "Mosaic CORE – Uncooled Thermal Sensor," n.d. Accessed on 27 Oct. 2025. https://www.thermal.com/mosaic-core.html.

[42] AnkerDirect, "Anker Zulu Power Bank 10000mAh 30W," *Amazon Product Page*, n.d. Accessed on 8 May 2026. https://www.amazon.com/dp/B0CZ9LV3H2.

[43] 18650 Battery Store, "Samsung 25R 18650 Battery – 2500 mAh 20 A," *18650 Battery Store — Product Page*, n.d. Accessed on 26 Oct. 2025. https://www.18650batterystore.com/products/samsung-25r-18650.

[44] Phipps Electronics, "How to Charge Li-ion Batteries with a DC-DC Buck Converter and BMS," *Phipps Electronics Blog*, 2025. https://www.phippselectronics.com/charging-li-ion-batteries-with-a-dc-dc-converter-and-bms/?utm_source=chatgpt.com.

[45] LM YN, "16.8 V 1 A Power Adapter Li-ion Battery Charger with LED Indicator (5.5×2.5/2.1mm DC interface) – Amazon ASIN B086PLPFQ2," *Amazon Product Page*, n.d. Accessed on 27 Oct. 2025. https://www.amazon.com/LM-YN-Indicator-Interface-Suitable/dp/B086PLPFQ2.

[46] Pichondria, "USB-PD 2.0/3.0 $\rightarrow$ 5 V 5 A Converter for Raspberry Pi 5," *Pichondria — Product Page*, n.d. Accessed on 27 Oct. 2025. https://pichondria.com/usb-pd-2-0-3-0-to-5v-5a-converter-for-raspberrypi-5/.

[47] Pinout.xyz, "5v Power," Raspberry Pi GPIO Pinout, n.d. Accessed on 20 Apr. 2026. https://pinout.xyz/pinout/5v_power.

[48] Long, S., "Trixie — the new version of Raspberry Pi OS," Raspberry Pi News, Oct. 2025. https://www.raspberrypi.com/news/trixie-the-new-version-of-raspberry-pi-os/.

[49] Raspberry Pi, "Raspberry Pi OS: OS installation," , n.d.. Accessed on 21 Apr. 2026. https://www.raspberrypi.com/documentation/computers/os.html#os-installation.

[50] Raspberry Pi, "Raspberry Pi Connect," , n.d.. Accessed on 21 Apr. 2026. https://www.raspberrypi.com/documentation/services/connect.html.

Appendix A – Current Hardware Specifications

Table A.1 – Specifications for the Arducam 64MP OwlSight [24]

Image Sensor	OV64A40
Still Resolution	64 Megapixels
Color Filter Type	Color
Shutter Type	Rolling
Optical Size	1/1.32”
Pixel Size	1.008 μm x 1.008 μm
Sensor Active Area	9248(H) x 6944 (V)
Output Formats	JPEF/YUV/RGB/RAW10
Video Modes (Raspberry Pi)	1080p 30fps, 720p 60fps, 640x480p 60/90fps
Video Modes (Camera Module)	3840×2160 14.8fps, 2312×1736 26.7fps, 1920×1080 45fps
IR Sensitivity	Integral IR-cut Filter, visible light only
Focus Type	Manual/Auto
Focal Length	6.65mm
F.NO	F1.9
Field of View (FOV)	84°(D)×68°(H)×56°(V)
Default Focus Range	12cm∞
Power Requirements	3.3V/300mA
Camera Board Size	25×24 mm

Table A.2 – Specifications for the Garmin Rangefinder V3HP lite [29]

Unit dimensions	54.99 (W) x 40.18 (D) x 35 (H)
Weight	38 grams
Water rating	IPX7
Resolution	1 centimeters
Typical accuracy	+/- 2.5cm at distances greater than 2 meters
Range	5 centimeters to 40 meters
Update rate	$\geq$ 1 kHz
Interface	I2C/PWM
Operating voltage	4.75-5 VDC; 6V Max
Current consumption	65mA idle; 85mA during acquisition
Operating temperature	-20 to 60° C
Laser wave length/Peak power	905 nm/1.3 W
Beam divergence	8mRadian
Optical aperature	12.5mm

Table A.3 – Specifications for the 8GB Raspberry Pi 5 board [33]

CPU	2.4GHz quad-core 64-bit Arm Cortex-A76
RAM	8GB LPDDR4X-4267 SDRAM
Bluetooth	5.0
USB Ports	2x3.0; 2x2.0
Ethernet	1 Gigabit Ethernet Port
Transceivers	2x4-lane MIPI camera/display
PCIe	1x2.0 PCIe interface port
Operating power	5V/5A DC power via USB-C, with Power Delivery support
Header pins	Standard Raspberry Pi 40-pin header
MicroSD	1xMicroSD slot
Wi-Fi	1xDual-band 802.11ac Wi-Fi chip

Appendix B – Comparison of Components

Table B.1 – Comparison of camera components [21, 24]

Camera Name	Arducam OwlSight	Raspberry Pi HQ Camera
Image Sensor	OV64A40	Sony IMX477R
Still Resolution	64 MP	12.3 MP
Shutter Type	Rolling	Rolling
Optical Size	1/1.32”	1/2.3”
Pixel Size	1.008 μm x 1.008 μm	1.55 μm x 1.55 μm
Sensor Active Area	9248(H) x 6944 (V)	4056(H) x 3040 (V)
Output Formats	JPEF/YUV/RGB/RAW10	RAW12 / 10 / 8, COMP8
Video Modes	2312×1736 26.7fps, 1920×1080 45fps	n/a
IR Sensitivity	Integral IR-cut Filter	Integral IR-cut Filter
Focus Type	Manual/Auto	Manual/back-focus adjustable
Focal Length	6.65mm	No fixed lens (separate)
F.NO	F1.9	n/a (lens dependent)
Field of View (FOV)	84°(D)×68°(H)×56°(V)	n/a (lens dependent)
Lens Mount	Stock Lens	CS-mount
Default Focus Range	12cm−∞	n/a (lens dependent)
Power Requirements	3.3V/300mA	3.3V; Unknown current
Camera Board Size	25×24 mm	38x38 mm

Table B.2 – Comparison of microcontroller components [31, 32]

Microcontroller	Raspberry Pi 5	Arduino Uno Q
CPU	2.4GHz quad-core Arm Cortex-A76	2GHz Quad-core Arm Cortex-A53
RAM	8GB LPDDR4X-4267 SDRAM	2/4 GB
Bluetooth	5.0	5.1
USB Ports	2x3.0; 2x2.0	1xUSB-C 3.0
Ethernet	1 Gigabit Ethernet Port	n/a
Transceivers	2x4-lane MIPI camera/display	Extra header pin for expanion
PCIe	1x2.0 PCIe interface port	n/a
Operating power	5V/5A DC power via USB-C	5V/3A
Header pins	Standard Raspberry Pi 40-pin header	2x17-pin header
MicroSD	1xMicroSD slot	n/a, onboard 16 GB eMMC storage
Wi-Fi	1xDual-band 802.11ac Wi-Fi chip	1xDual-band 802.11ac Wi-Fi chip

Appendix C – Bill of Materials

Table C.1 – Pechlivani et al. bill of materials [39]

Item	Estimated Cost (EUR)
Front Camera Telephoto Lens	41.27
Diffraction Grating	3.69
Macro Lens	23.39
Raspberry Pi 4	48.31
Raspberry Pi NoIR Camera Module V2.1	25.73
Verger S20 Power Bank	32.20
Raspberry Pi Cooling Fan	2.34
Total cost (w/ EUR to USD conversion)	$207 USD

Table C.2 – Barjaktarovic et al. bill of materials [37]

Item	Estimated Cost (EUR)
Raspberry Pi 4	70
Arducam 4x Camera Kit	300
Thermal Camera	500
10,000 mAh Power Bank	20
Optical Filter 432/517/615 nm	300
Optical Filter 577/690 nm	270
Optical Filter 750 nm	140
Optical Filter 550/660/850 nm	300
3D printing materials	70
Total cost (w/ EUR to USD conversion)	$2,295 USD

Table C.3 – Current bill of materials

Item	Quantity	Cost ($)	Purchase Link
Arducam OwlSight 64MP	1	56	www.amazon.com/dp/B0D3XHZ9S8
Garmin Rangefinder V3HP	1	150	www.garmin.com/en-US/p/578152
Raspberry Pi 5 8GB Kit	1	100	www.amazon.com/dp/B0DNF5JWB4
Logitech K400 Wireless Keyboard	1	30	www.amazon.com/dp/B014EUQOGK
4k Micro HDMI to HDMI Cable	1	6.40	www.amazon.com/dp/B08C2PVP7J
SanDisk 128GB MicroSD	1	13.60	www.amazon.com/dp/B0B7NTY2S6
SD Card Reader	1	7	www.amazon.com/dp/B07D1J88CF
25V 680 μF Capacitor	1	5	www.amazon.com/dp/B0D78PHBD1
Anker Zulu 10000mAh 30W battery	1	30	www.amazon.com/dp/B0CZ9LV3H2
INIU 240W USB C cable	1	7	www.amazon.com/dp/B0DHJQLGPB
	Total cost	408	