Alpha Design Report

S.I.M.B.A. Team 12/10/2023 Professor Murray

Ian Beck

Curtis Bucher

Luis David Garcia

Braedan Kennedy

Brian Nguyen

Sepp Williams

Tyler Bovenzi – Client

Table of Contents

1 Introduction

1.1 Project Overview

Our computer engineering capstone project, originally named Rover on A Chip (ROACH) and now rebranded as Solar Integrated MoBile Automaton (SIMBA), represents a groundbreaking endeavor in the realm of autonomous rover technology, guided by the expertise of our client, Tyler Bovenzi. This venture signifies a substantial leap forward in rover capabilities compared to preceding projects. At the heart of our mission lies the seamless integration of the powerful Kria K26 System-on-Module (SoM). Achieving this milestone involves a multi-faceted approach, including the pivotal task of migrating the Verilog firmware from the previously employed PYNQ-Z1 Field-Programmable-Gate-Array (FPGA) to the new platform. This endeavor also encompasses the comprehensive reconfiguration and expansion of the accompanying C code, essential for interfacing with and governing the FPGA. The transition to the Kria K26 SoM is the development of a customized printed circuit board (PCB), meticulously engineered to house both the SoM and supporting electronics within a single, cohesive unit. Fueled by innovation and a commitment to cost-effectiveness, the finalized SIMBA platform is set to establish precedence in a new era of potential in autonomous rover technology.

1.2 Justification and Motivation

The space exploration industry, particularly focused on Mars, has become increasingly important in present-day society. The imperative for cost-effective vehicles capable of uncovering new scientific insights propels the SIMBA team towards improving the existing space rover to enhance efficiency and functionality. With the upcoming launch window to Mars in 2026, the team is focused on transitioning the rover system architecture from a developmental stage to a fully deployable vehicle for exploration. As a cohesive team comprising software, computer, and hardware engineers, we are excited to apply our skills toward innovative rover technology.

1.3 Clients and Customers

Client

The SIMBA capstone project benefits significantly from the involvement of Tyler Bovenzi as its client. Tyler's unique background as a foundational member of the original ROACH project brings an exceptional advantage to our team. His comprehensive experience, having successfully completed his capstone and senior project on the rover, encompasses a broad spectrum of skills including software development, digital design, and PCB design. This breadth of expertise provides the SIMBA team with direct access to a wealth of knowledge, particularly in navigating the challenges inherent in advancing the SIMBA rover. Furthermore, the SIMBA rover project is a collaborative endeavor that extends beyond a single team. It involves the concerted efforts of two additional mechanical engineering teams, bringing the total team count to fifteen. These include an Arm team, tasked with the development of the rover's arm, and a Vision team, dedicated to the computer vision aspects of the rover. Together, these teams play a crucial role in the collection and identification of Mars space capsules. Guiding this multifaceted collaboration is Professor Murray, also referred to as client M. Professor Murray's role is pivotal in facilitating a seamless exchange of knowledge and expertise among the teams. This collaborative framework significantly augments the capabilities of the SIMBA team, enriching their collective skill set and enhancing their potential for achieving the project's objectives successfully. This synergy is instrumental in driving the project towards its goal – the proficient and innovative development of the SIMBA rover.

Client Archetypes

The SIMBA team has identified three distinct customer profiles. The *first c*ategory encompasses space agencies such as NASA and SpaceX, spanning both public and private sectors. For these space agencies, the SIMBA rover serves as an economical vehicle for space exploration and a versatile development platform. Despite often operating with billion-dollar budgets, these organizations find that integrating and testing new hardware and software components on a cost-effective rover is a notably more budget-friendly alternative to working directly with expensive production hardware. The *second* customer profile consists of universities seeking to harness the potential of the SIMBA rover for robotics courses and research. Within the academic realm, students gain the opportunity to delve into various disciplines, including software, digital design, PCB design, power, and mechanical engineering. On the research front, both students and professors can utilize the rover for conducting driving tests and developing components, whether related to motors, architecture, or any other aspect. This eliminates the need to construct a rover from the ground up, providing a pre-built development platform for rapid and cost-effective progress. The *third* customer archetype pertains to the military, which could potentially reap benefits from the SIMBA rover, particularly in executing hazardous material missions and reconnaissance. The incorporation of an arm into the rover enhances its capabilities for these tasks. Once again, cost-effectiveness plays a pivotal role, as the replaceability of the rover helps mitigate potential financial losses.

1.4 Stakeholders

One of the primary stakeholders in the SIMBA project, in addition to our client, is the Cal Poly Computer Engineering (CPE) Department. Our progress and achievements are directly reflected in the department, as they have provided us with invaluable opportunities to engage and participate in this collaborative endeavor. Their support is instrumental in the success of this project, and we are committed to delivering results that not only benefit our client but also positively reflect upon the CPE Department and the CPE capstone program.

1.5 Framed Insights and Opportunities

Over the course of the SIMBA project, our team has engaged in extensive discussions and communications with our client, Tyler Bovenzi, as well as other stakeholders, to gain a comprehensive understanding of their respective needs and expectations. This section aims to synthesize the insights gathered from these interactions, distinguishing between the specific requirements of the client and those of other stakeholders, including community partners.

Client-Centric Insights

Tyler Bovenzi has provided invaluable insights into the project's core objectives and technical intricacies. Some key takeaways from our discussions with Tyler include:

1. **Minimum Viable Products**: Tyler emphasized the critical components required for the SIMBA project's success, covering aspects like hardware, firmware, and software development. These include the integration of the new Kria K26 SoM, migration of Verilog firmware, and implementation of IMU and GPS functionalities.
2. **Access to Codebase**: Tyler provided access to essential repositories, including the GoScout GitHub, which contains Python code used GPS/IMU integration. Additionally, he highlighted the significance of creating a dedicated GitHub repository for Verilog firmware, providing a way to keep track of changes and improvements as development continues.

3. **Software Considerations:** Tyler provided in-depth knowledge of which GPS and IMU support libraries should be used and avoided. Additionally, Tyler suggested a comprehensive understanding of both module's register maps, which enabled us to configure the IMU module effectively for read/write operations. This also helped resolve the GPS read/write operations by informing us to adjust the message send rate and measurement frequency, ensuring that an appropriate number of bytes was accurately written per GPS message received. Lastly, Tyler recommended conducting a meticulous inspection for potential memory leaks.
4. **Firmware Considerations:** Tyler demonstrated and described his Verilog modules for controlling the motors on the GoScout project. Using this information, as well as his source code, we were able to adapt the legacy motor driver source code for the ROACH project.
5. **Hardware Considerations**: Tyler's guidance on hardware design, such as providing PCB schematics and noting cases where additional voltage regulation may be required, underpins the project's success. Additionally, Tyler recommended different software tools and manufacturers for designing and producing the PCB.
6. **Operating System Selection**: Tyler shared insights on the potential operating systems for the SIMBA, weighing the advantages of PetaLinux and FreeRTOS against the resource-intensive nature of Ubuntu.

Community Partners and Stakeholder Perspectives

While our primary focus has been on meeting Tyler's explicit requirements, we recognize the broader implications of the project on the community and other stakeholders. Some critical observations include:

1. **Educational Impact**: The project's potential as an educational tool for universities and robotics courses is substantial. Through the SIMBA rover, students can engage in and learn about a diverse array of engineering disciplines, including software, digital design, PCB design, and mechanical engineering.
2. **Space Agencies and Exploration**: The SIMBA rover holds promise for space agencies, both in the public and private sectors. Its cost-effectiveness presents a valuable opportunity for testing new hardware and software components, potentially revolutionizing space exploration.
3. **Military Applications**: The addition of an arm to the rover expands its applicability to military scenarios, particularly in hazardous material extraction. The cost-effectiveness of the rover aligns with the military's need for adaptable and replaceable equipment.
4. **Potential for Innovation**: By providing an accessible platform for development and testing, the project could spur innovation within the field of robotics, leading to advancements that benefit a wide range of industries.

1.6 Goals and Objectives

The SIMBA project aims to achieve a seamless integration of the state-of-the-art Kria K26 SoM, revolutionizing the rover's processing capabilities. Additionally, objectives include firmware migration and optimization, enhanced software functionalities, establishment of GitHub repositories for version control, and cost-effective development that does not compromise performance or functionality.

Project Goals

1. **Integration of SoM:** The primary goal of the SIMBA project is the seamless integration of a cutting-edge System on Module (SoM). This upgrade is anticipated to greatly enhance processing capabilities, enabling the rover to execute complex tasks with unprecedented efficiency.

2. **Firmware Migration and Optimization:** A key objective is the migration of Verilog firmware to ensure compatibility with the new SoM. This task encompasses optimizing the codebase for performance and efficiency, paving the way for future enhancements.
3. **Enhanced Software Capabilities:** SIMBA's software suite will be enriched with functionalities borrowed from the successful GoScout project. This includes advanced navigation algorithms, data processing techniques, and adaptive decision-making capabilities.
4. **Streamlined Hardware Design:** A custom PCB will be designed to hold all the control electronics and provide all necessary I/O with some options for future expansion.
5. **GitHub Repositories for Version Control:** Establishing dedicated GitHub repositories is crucial for streamlined collaboration and version control. This initiative ensures transparency in development and serves as a knowledge hub for future teams.
6. **Cost-Effective Development:** A paramount objective is to develop cost-effectively without compromising on functionality or performance. This is specific to hardware, ensuring that the team actively searches for the best-priced components and manufacturing services when creating the PCB. This approach not only benefits our clients but also opens avenues for broader applications in education, research, and industry.

Specific Objectives

1. **Complete SoM Integration:** Achieve seamless integration of the SoM, ensuring all hardware components communicate effectively.
2. **Firmware Compatibility Testing:** Rigorously test the migrated Verilog firmware to verify its compatibility with the new SoM.
3. **Optimize Verilog Firmware:** Identify and implement optimizations in the Verilog codebase to enhance performance and resource utilization.
4. **Implement Advanced Navigation Algorithms:** Integrate advanced navigation algorithms from the GoScout project to enhance SIMBA's autonomous capabilities.
5. **Develop a Custom System Control PCB:** Design and integrate a PCB containing the Kria K26 SoM, I/O, power management, and motor control modules.
6. **Establish GitHub Repositories:** Set up dedicated repositories on GitHub for organized version control, ensuring seamless collaboration among team members.
7. **Cost Analysis and Optimization:** Conduct a thorough cost analysis of components and materials, seeking opportunities for optimization without compromising quality.
8. **Documentation and Knowledge Transfer:** Maintain comprehensive documentation of all project developments, ensuring seamless knowledge transfer to future teams.

1.7 Outcomes and Deliverables

The SIMBA project will culminate in a series of tangible deliverables and outcomes meticulously designed to meet the needs of our client, community partners, and other stakeholders. These deliverables are structured to serve as clear indicators of progress and successful project completion.

1. **Integrated SoM**: The pivotal achievement will be the seamless integration of a cutting-edge System on Module (SoM). This enhancement will grant the rover improved capabilities, revolutionizing its capacity to perform intricate tasks efficiently. The Minimum Viable Product (MVP) will comprise a PCB housing the SoM, complete with remapped I/O for optimized functionality.

2. **Firmware Migration and Optimization**: The Verilog firmware will undergo a migration process to ensure seamless compatibility with the new SoM module. This task encompasses thorough codebase optimization to enhance performance and efficiency. The outcome will be a refined firmware foundation, setting the stage for future enhancements.
3. **Enhanced Software Capabilities**: Enhanced Software Capabilities: Building on the foundations laid by the successful GoScout project, the SIMBA software suite is set to undergo a significant enhancement, specifically having the software be updated from Python to C++ to facilitate greater performance. Both GPS and IMU libraries have been modularized using object-oriented programming to make servicing GPS and IMU function calls more accessible to future software components.

Upon reaching these critical milestones, the team will have successfully navigated through the MVP phase. The subsequent steps involve furthering hardware and software capabilities, considering scalability for additional motors, and investigating power optimization strategies. Additionally, the integration of advanced control algorithms and continuous refinement of the PCB layout will be prioritized.

2 Background Research

2.1 Introduction

This background section outlines the SIMBA system in its present state, detailing a comprehensive understanding of the software, hardware, and digital design decisions made, as well as research regarding the overall functionality of the previous design. Additionally, the software for the SIMBA project plans to expand its functionality by incorporating GPS and IMU functionality inspired by another project called GoScout. For more information about background research, please view the initial research document [2].

2.2 PCB Design and Integration

The move to the Kria K26 SoM means a new PCB design that houses only the necessary electronics. This is because we want to consolidate functionality to a minimal set of hardware to improve power efficiency. To accomplish this, the hardware team will place the Kria K26 SoM onto a custom PCB that incorporates only essential connections to peripheral devices. When finished, the PCB should replace the original Kria KR260 development board. However, the hardware team's first revision PCB works in conjunction with the development board instead of replacing it.

Some improvements were made to the original PCB design to make it a more power-efficient system. This includes switching to a more power efficient switching voltage regulator from a linear voltage regulator. A 12V switching voltage regulator from Pololu was chosen to provide power to the Kria development board. This part was chosen based on previous experience with the manufacturer and a recommendation from Professor Murray. Additionally, the grounding of the board was improved by moving to a four-layer board design from two layers. This change will be beneficial for the final design housing the SoM to accommodate the increased component density.

Finally, an additional change to improve power efficiency will be the removal of the cooling fan currently present on the Kria KR260 development board. According to the Kria documentation from AMD [15] the standalone Kria K26 SoM can handle approximately 6.8W of thermal load dissipation without any cooling solutions. The current power draw of the Kria will need to be evaluated and any additional thermal dissipation required will be accomplished through a passively cooled metal plate attached to the thermal pad of the K26.

2.3 Kria KR260 Development and Integration

The original design of the ROACH project made use of the PYNQ-Z1 development board which consisted of an ARM Cortex A9 processor and an FPGA. The new platform that the digital design team is switching to is the Kria KR260 robotics starter kit – a development board for the Kria K26 SoM which features a quad-core ARM Cortex-A53 processor paired with an FPGA. The move to the new development board was warranted by the need for a more powerful chip which doubled the number of cores compared to the original PYNQ board. It is also built specifically for robotics applications, so the board is well-suited for motor control and interfacing with the GPS and IMU that we have selected.

The CPU and FPGA modules on the Kria SoM communicate via a high-speed Advanced eXtensible Interface (AXI) interconnect that can be configured in a Vivado block diagram. In addition to this, the general-

purpose input/output (GPIO) pins can be configured to allow for the FPGA to communicate with external hardware like the motor controllers, or the GPS and IMU. Since the GPS and IMU use I2C to communicate, the Kria board will have to accommodate this by configuring certain GPIO pins to act as I2C-compatible pins.

One problem that needs to be solved when implementing an autonomous rover design on an FPGA is power considerations. As this rover is meant for exploration on Mars, it will need to run for prolonged periods of time without external assistance from any humans. To do this, power conservation will need to be a main priority, and this can be optimized in the FPGA by minimizing the number of lookup tables (LUT) wherever possible. One of the most practical ways we might do this is to migrate from a finite state machine to a block ram. Another way we could minimize LUT's is to avoid active low signals. These require an inversion before they can directly derive the control port of a flip flop and take up one more LUT than active high signals.

2.4 GPS and IMU Development and Integration

One of the primary software goals for the SIMBA rover is onboard integration with the SAM-M8Q (GPS) [6] and ICM-20948 (IMU) [7] modules, which requires the creation of C++ libraries for these modules for portability. Currently, this functionality is implemented in Python using standard packages created for these modules with the GPS data sheet found respectively at [3] and the IMU datasheet at [10], but with the SIMBA rover focusing on performance optimization, the software team is focused on migrating the functionality implemented in Python to the more efficient C++ programming language.

For GPS integration, the team has delved into understanding key libraries and communication protocols since the GPS is u-blox protocol based. They have extensively researched the Melopero_SAM-M8Q library and the u-blox protocol, which is commonly used with u-blox and GNSS GPS receivers. The UBX protocol employs a binary format to transmit messages, each of which contains a message identifier, payload data, and checksum information [7]. These messages will be transmitted via I2C and interpreted by the Melopero_SAM-M8Q library, which serves as an API between the development board and the GPS module [2]. To support I2C communication, the team plans to use the libi2c-dev library instead of the previously used "smBus2" library in the Python implementation [11]. Overall, this requires the incorporation of a library for u-blox protocol to format the buffers and message. Ultimately, GPS integration is crucial for providing the rover with accurate global position information, and in the larger scheme for creation of mapping to generate maps to determine the terrain and establish a coordinate system like Earth.

In the process of integrating the ICM-20948 (IMU) board, the SIMBA team is transitioning from the smBus2 library and board module used for I2C interfacing in Python to the libi2c-dev library in C++. This change aligns with the approach taken for GPS integration, ensuring consistency across the development process. The IMU provides critical support by supplying accelerometer and gyroscope data, which is particularly valuable in instances where GPS signals are unavailable or unreliable. This built-in redundancy is crucial for preserving the operational functionality and navigational accuracy of the rover under such conditions.

In addition, the team has been evaluating various C++ libraries to facilitate real-time plotting of acceleration and gyroscope data during testing and analysis. After careful consideration, the decision was

made to adopt the matplotlibc++ library, which replicates the functionality of Python's matplotlib library [13]. The team recognized that matplotlibc++ requires CMake, which required learning to utilize it to compile the libraries. The data plots generated are instrumental in the development of filtering algorithms, which will process the raw data to ensure that the readings fall within acceptable ranges and adhere to predetermined standards.

3 Marketing and Customer Requirements

3.1 Project Goals

The objective of this project is to create a fully operational prototype of the SIMBA rover, incorporating materials from the previous capstone project (ROACH). The aim is to construct a space-capable rover for our client. The outlined goals and deliverables for this project are as follows:

- Integrating the KR260 development board into the existing project, replacing older hardware.
- Porting the Verilog firmware from the PYNQ-Z1 FPGA to the new platform, ensuring compatibility.
- Updating and expanding the C code used to interface with and control the FPGA.
- Designing and fabricating a PCB to house the SoM and other necessary electronics in a single package.
- Incorporating software from the GoScout project, focusing on improving navigation and enhancing data processing.

3.2 Marketing and Customer Requirements Table

Many requirements were derived through back-and-forth discussion with our client Tyler. These discussions involved a balance of fulfilling necessary client requirements, while communicating and retaining important constraints and limitations to our design. We also engaged in discussion with our partnered mechanical engineering teams, on the aspects of the design that are determined at a system level, notably power consumption, which relies heavily on the power output of the solar cells and the current draw of the motors.

Spec #	Customer / Marketing	Engineering Parameter	Engineering Requirements (with units)	Tolerance	Risk	Compliance
1	Efficient	Energy Consumption	4 W	Avg	M	A, T
2	Accurate	GPS Positional Accuracy	2.5 M	Max	M	A, T
3	Accurate	Motor Control Accuracy	3000 pulses per rotation	None	M	A, T
4	Durable	Part exp. lifetime	1 year	Min	H	A
5	Functional	Features implemented	All legacy ROACH features.	Min	M	A, S, T
6	Cheap	Production Cost (SIMBA Rover-On-Chip)	$500	Max	M	A
7	Environmental	Emission, Supply Chain	Zero Rover Emissions, Limited Battery Size (power efficiency)	Max	L	A, T

Figure 1: SIMBA Marketing and Customer Requirements Table.

3.3 Testing Plan

We plan to conduct extensive testing to ensure that we meet each of these marketing and engineering goals. Our designs will go through detailed analysis (A), to confirm that each design at least nominally fulfills the requirements listed above. Following the design phase, when we have constructed the system, certain features will undergo physical testing (T). Particularly these features will include more of the physical attributes of the rover, including energy consumption, GPS positional accuracy, motor control accuracy, and the functionality of the entire system. The initial goal is for the functionality of the SIMBA system to meet the same functionality of the ROACH system, then to add new features to the existing system.

3.4 Customer Archetypes and Use Types

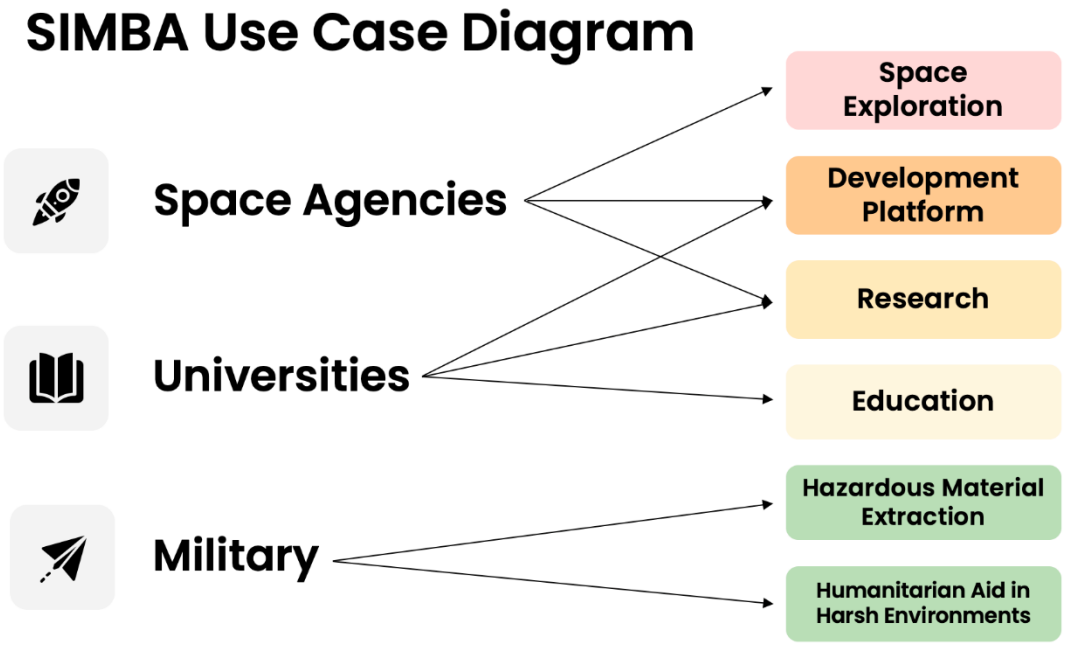

Figure 2: SIMBA use case diagram

Our customer archetypes include space agencies, universities, and the military. Possible applications include space exploration, use as a development platform for other projects, research, education, and various humanitarian applications. More information is available on these customer archetypes and their use cases in SIMBA's *Archetypes and Use Cases Report* [4].

3.5 Marketing Data Sheet

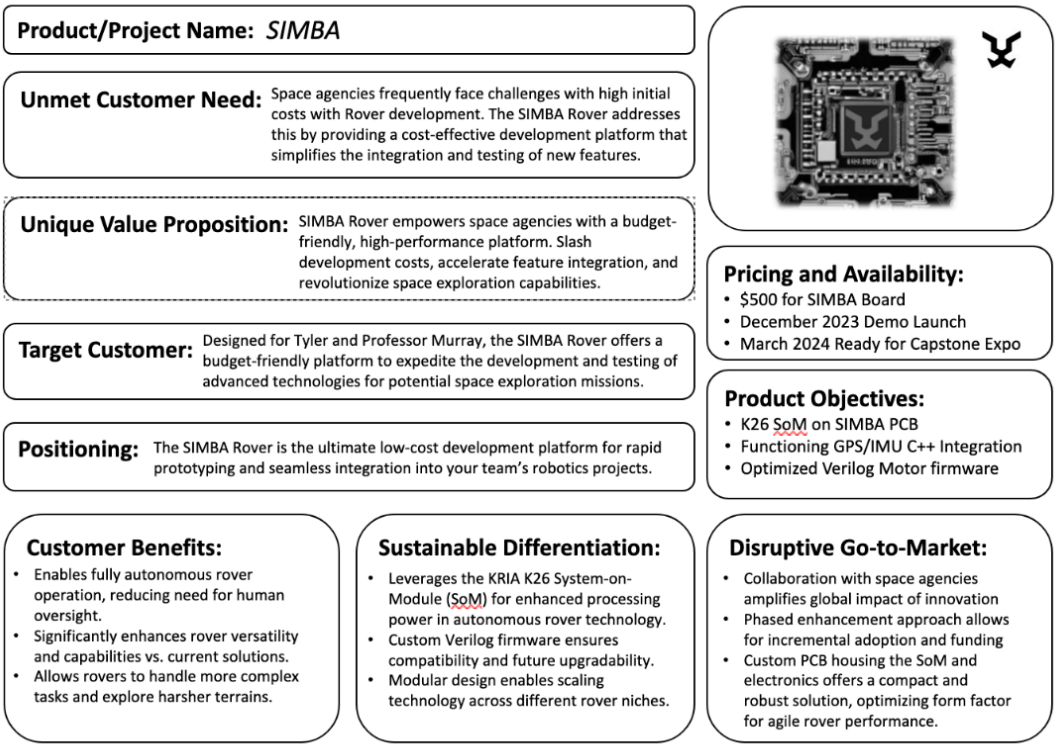

Figure 3: SIMBA Marketing Data Sheet.

The SIMBA marketing datasheet provides a concise yet comprehensive overview of our groundbreaking project. Highlighting key aspects such as unmet customer needs, unique value propositions, target customer demographics, and disruptive go-to-market strategies, this document encapsulates the essence of SIMBA's revolutionary approach in the field of autonomous rover technology. With detailed insights into product objectives, pricing, and availability, the marketing datasheet serves as an indispensable resource for stakeholders seeking a clear understanding of SIMBA's value proposition and market positioning.

4 Design Development

The ROACH and GoScout systems serve as the primary initial conceptual prototypes for the SIMBA project. Additionally, integration and testing will be facilitated using a Kria KR260 development board and a specially designed PCB intended to work with the development board. This approach ensures thorough evaluation before transitioning the new system to the physical rover.

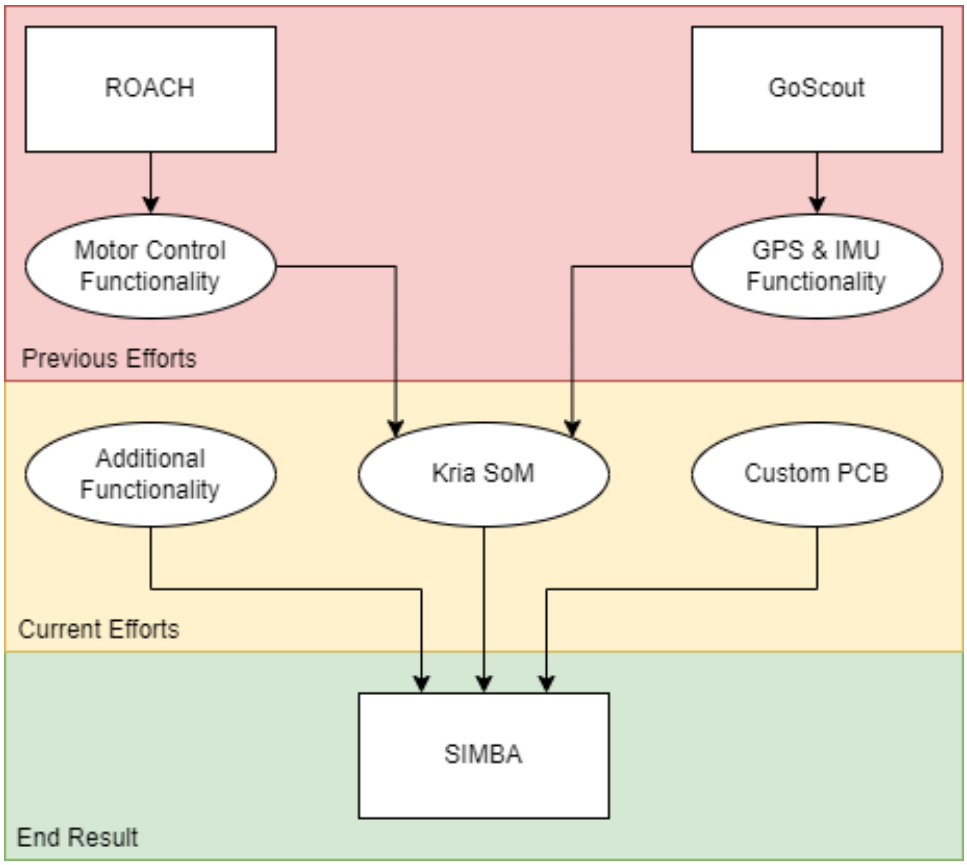

Figure 4: SIMBA Design Development Flowchart.

Current prototyping consists of breadboard I/O testing on the Kria development board, Raspberry Pi I2C communications testing with the IMU and GPS, and C software development. When the prototype motor control PCB is finished, these prototyping projects can be consolidated into the single prototype system using the Kria development board to finalize designs decisions before integrating the final board.

4.1 Hardware Development

The first iteration of the new rover control PCB is based on the motor control PCB from the ROACH capstone project. Because much of the FPGA design was based on the work done for ROACH, the updated motor control PCB maintained the same motor controller architecture with 10 for rover movement, and 4 spare for future expansion. The two primary changes to the board were the power distribution and I/O. Previously the ROACH motor control board was designed to mount on top of the PYNQ-Z1 development board. For this iteration of the SIMBA PCB, the I/O was reworked to mirror the PMOD connectors and

Raspberry PI HAT pin header on the KRIA KR260 development board to allow for easy testing and reconfiguration of connections. The power conversion system was changed from a linear regulator to a switching voltage regulator for efficiency improvements. A 12V step down converter was added to power the Kria development board and a 3.3V rail was provided from the Kria development board. The PCB design was switched to a four-layer layout from a two-layer PCB to add a ground plane and battery voltage plane for better grounding and easier routing of battery power to the motors.

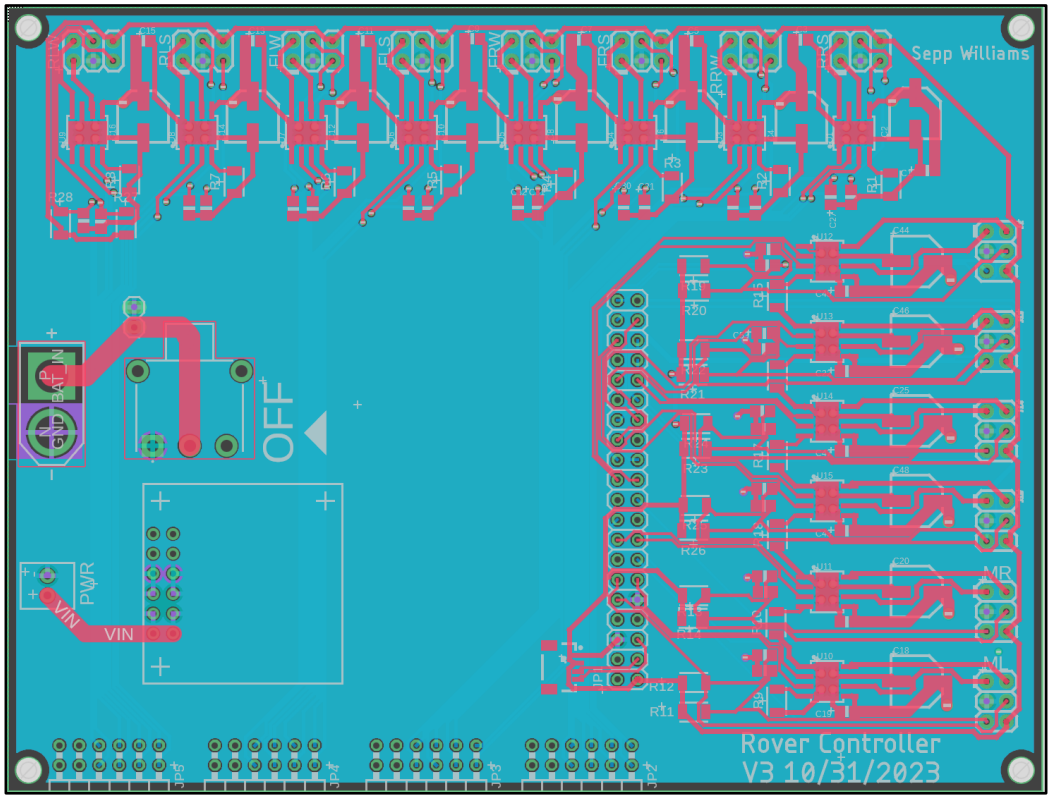

Figure 5: SIMBA PCB Schematic.

The next steps in hardware design will be incorporating the Kria K26 SoM and some of the necessary modules from the Kria KR260 development board onto the next PCB. The current PCB acted as a proof of concept for integrating the rover control onto the Kria SoM, and will provide a platform for firmware and software development while the next PCB revision is under development.

Some improvements from the current PCB iteration will include new motor to PCB connectors, and additional pull-up resistors on motor control connections. The motor connectors will be changed from 2x3 pin headers to a mounted, keyed, 6 pin connector. Having keyed connectors will help prevent incorrectly connecting the motors to the control board and should reduce the possibility of damaging components. An ongoing investigation is occurring into why some of the Kria development board GPIO are not fully functional with the suspected issue being a lack of pull-up resistors on open-drain pins.

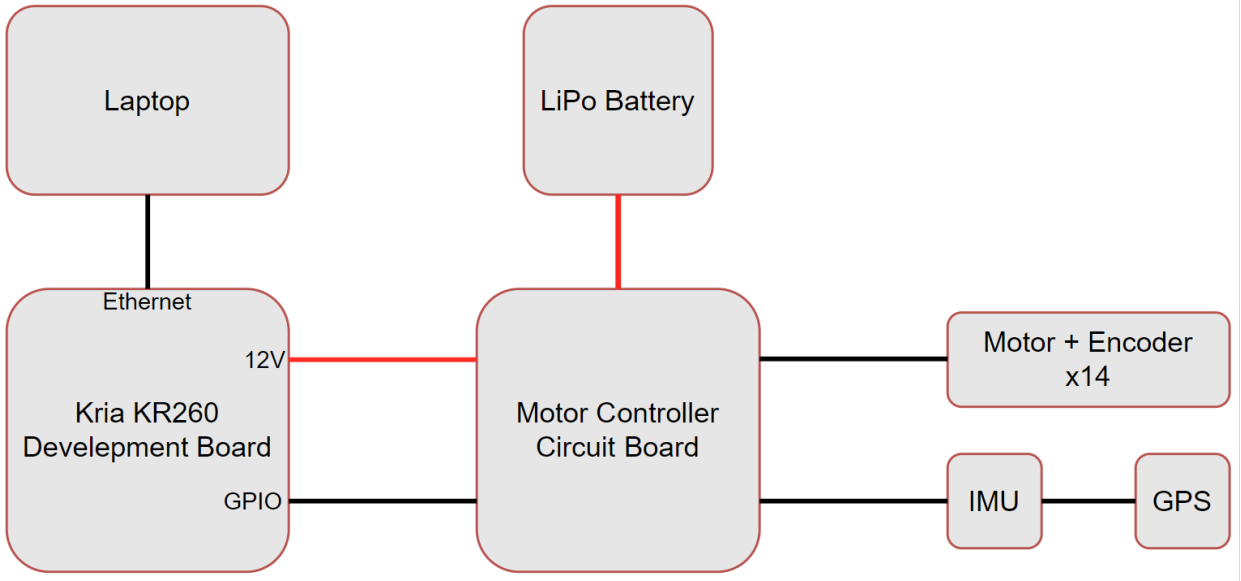

Figure 6: Current SIMBA PCB Modules Layout.

Without a cooling fan, the 12V voltage rail currently being supplied by the motor control PCB can be removed. Thus, power distribution will consist of the battery voltage, controlled by a switch, and 5V, 3.3V and 1.8V rails dependent on the Kria SoM requirements and what each module ported over from the development board requires. Additional layers could be introduced to help with power routing, particularly for the widely used 3.3V rail.

To simplify the necessary I/O, the next iteration of circuit board will drop the ethernet connectivity of the Kria development board for WIFI, thus allowing control of the rover while in motion without a tethered connection. A USB interface will be maintained for a wired computer connection and eventually possible expansion for additional components from other capstone project teams. An additional solenoid control MOSFET and connector will be added by the request of the mechanical teams.

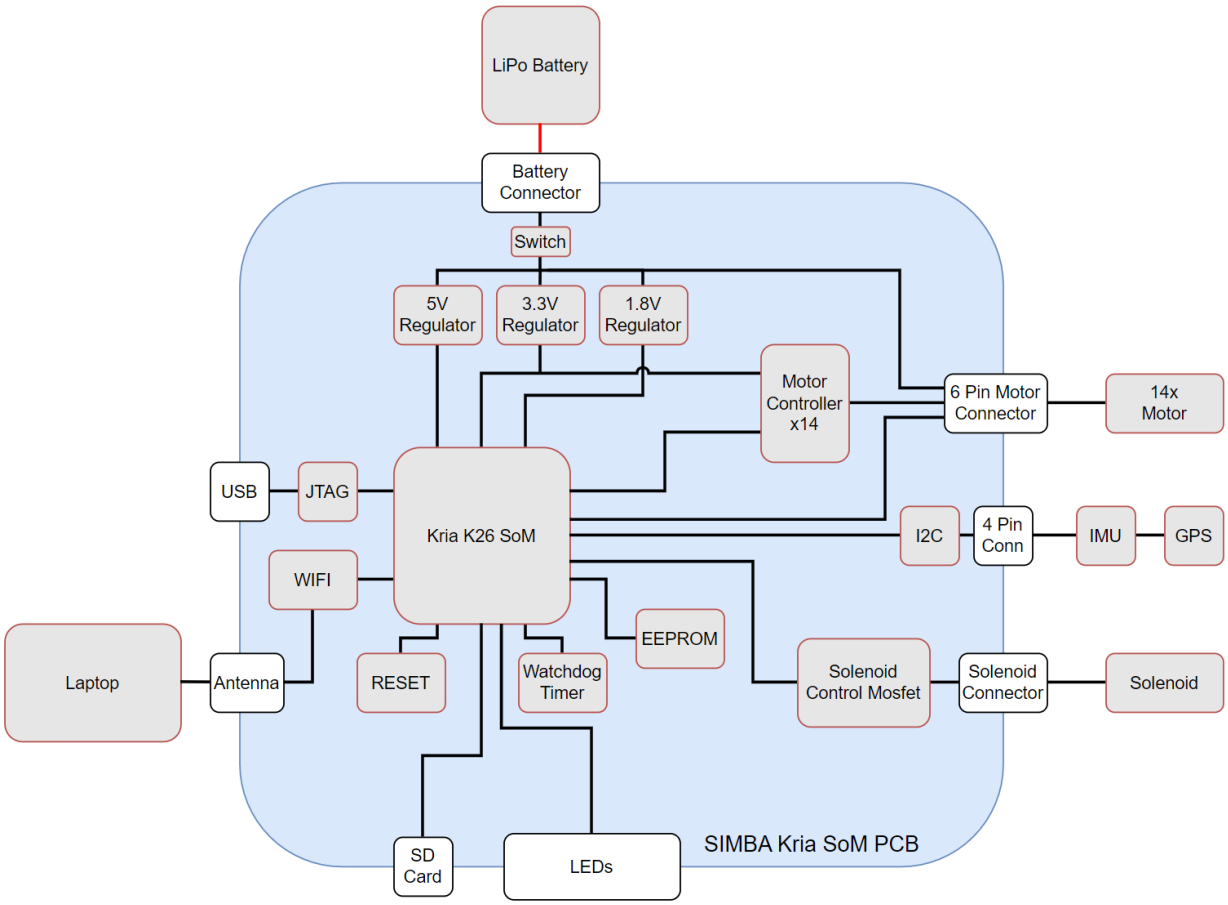

Figure 7: Future SIMBA PCB Modules Block Diagram.

4.2 Digital Design Development

The new SIMBA firmware builds off the ROACH project's original Verilog code. However, several improvements were made. It was refactored to support a more robust design and accommodate the new Kria KR260 development board for constraints and the Kria K26 SoM for internal design [16].

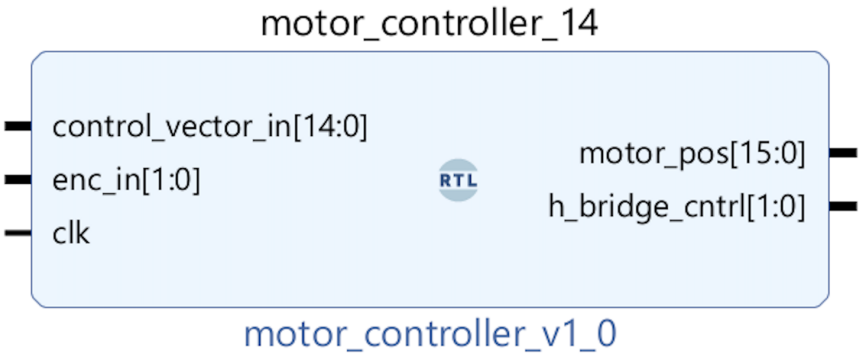

Figure 8: Single Motor Controller Verilog Module.

$$(\neg PWM \ \vee \ DIR) \ \wedge EN = OUT1$$

$$\neg(PWM \ \wedge \ DIR \ ) \ \wedge \ EN = OUT2$$

Figure 9: H-Bridge Decoder.

Because ROACH employs a different SoM than GoScout, the architecture of the digital design needed to be heavily modified to support communication between the ARM CPU and FPGA within the Kria KR260. It was also particularly challenging to adapt the project to work with the new constraints defined by the Kria system.

To support a more robust system, the block design was modified to use a hierarchical, abstracted structure where each motor is assigned its own motor controller, and AXI GPIO address. This is opposed to ROACH's flat design, where all 14 motors are controlled by a single, monolithic, multi-motor control module. While this increased modularity increases the complexity of the architecture and block design, it makes the project more maintainable, flexible, and scalable. Adding or removing a motor to the rover design only requires adding or removing two blocks from the block design. In addition to these modifications, a single AXI I2C block was added to the block design to communicate with the GPS and IMU.

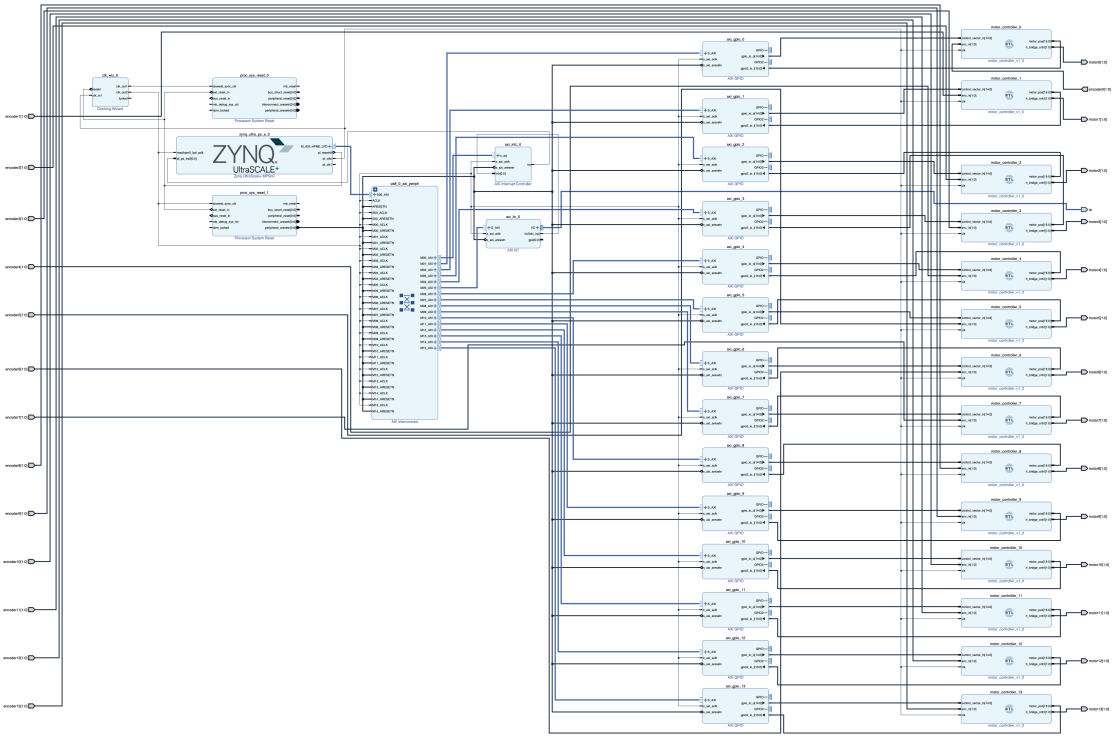

Figure 10: Rover Block Design in Vivado.

In the image above, you can see a diagram of this hierarchical architecture. Each of the 14 blocks on the far right are our individual motor control modules. Immediately to the left of those blocks are the associated AXI GPIO modules, that are each assigned to a specific motor controller module.

While this hierarchical system increases modularity compared to previous rover generations, we still faced significant challenges in bringing all 14 motors to life. Because our first-generation PCB connects to the

Kria KR260 development board, rather than connecting to the Kria SoM itself, we were required to use the limited GPIO available from the development board to control each motor, leading to several issues and setbacks. Several I/O pins on the Kria development board are pulled high or low without respect to our digital design needs, causing several motors to drive uncontrollably or failing to report their position accurately. To circumvent this issue, we were able to utilize auxiliary motor control ports to control most motors on the rover, but a long-term solution will require further debugging efforts, and potential PCB revisions.

4.3 Software Development

Software development has been focused on establishing closed loop motor control and interfacing the I2C peripherals. Both efforts build heavily upon the digital design team's work.

Closed Loop Motor Control

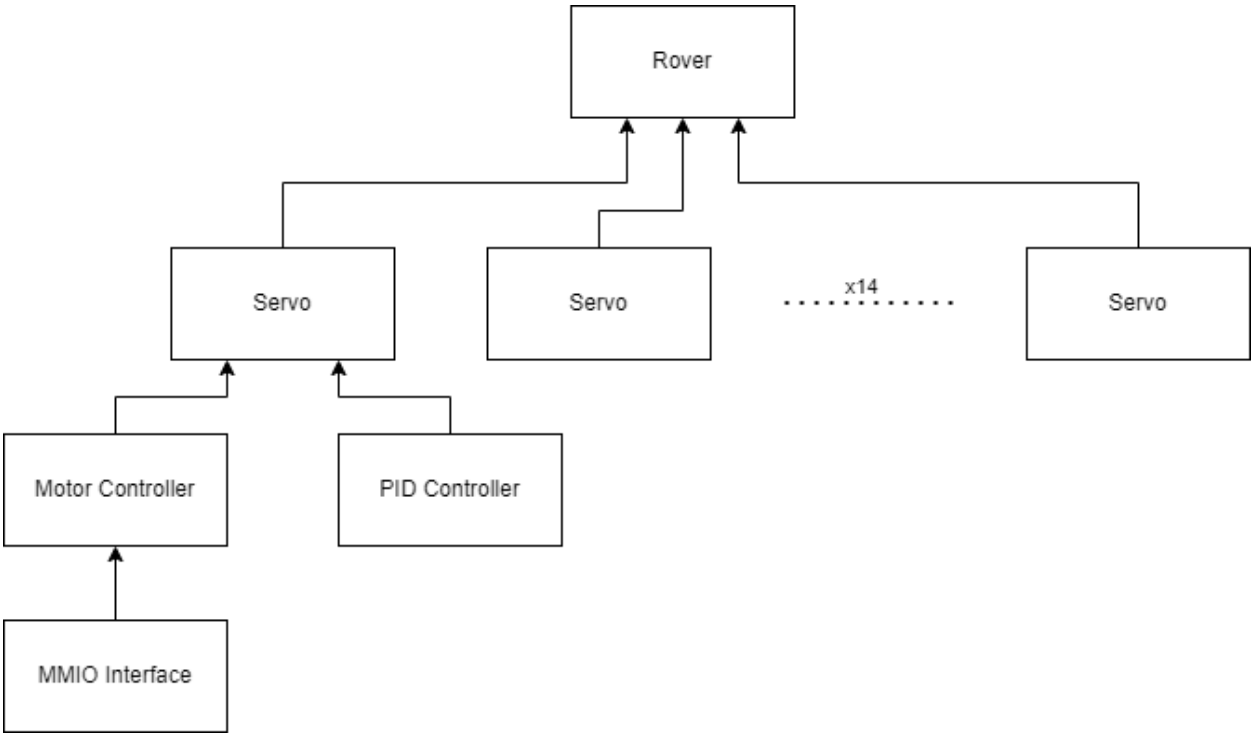

Figure 11: Inheritance Diagram for the Rover.

The digital design team created a motor controller Verilog module, which handles the real-time aspects of controlling a motor such as PWM signal generation and quadrature encoder decoding. However, to make use of that motor controller there must be a software interface. This interface was implemented using memory mapped I/O (MMIO), where certain memory addresses in the Kria SoM's RAM correspond to the control signals from the motor controller Vivado module. As seen in the inheritance diagram, the MMIO interface is inherited by the motor controller class. This is simply a software representation of the hardware module and allows for its simple use and configuration.

With motors being controlled in software, control logic needs to be implemented to create a closed loop motor system. In a closed loop motor system, a setpoint is issued by the user and the system moves the motor as quickly and accurately as possible to that setpoint. This control logic implemented here is called

a proportional-integral-derivative (PID) controller. The servo class unifies the motor controller's ability to read and control the motor's position with this closed loop control algorithm.

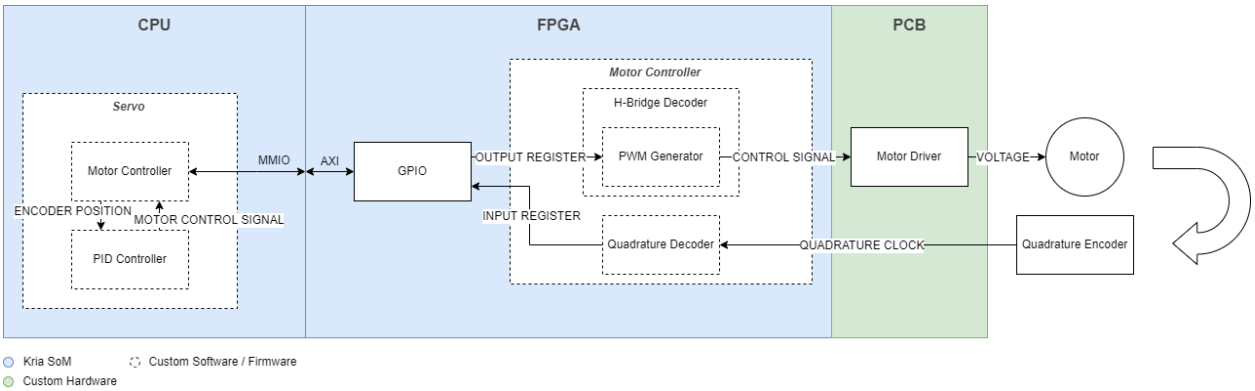

Figure 12: Diagram of SIMBA's Control Loop System.

As seen in Figure 12, implementing closed loop control in software was the last step in a long chain of different components – all of which need to work correctly together. The motor turns the encoder, whose position is decoded by the hardware motor controller, and read by the software motor controller. Then the PID controller determines a new motor control signal which is fed into the software motor controller, decoded by the hardware motor controller (using the equations in Figure 9), and actualized by the motor driver IC on the PCB to move the motor.

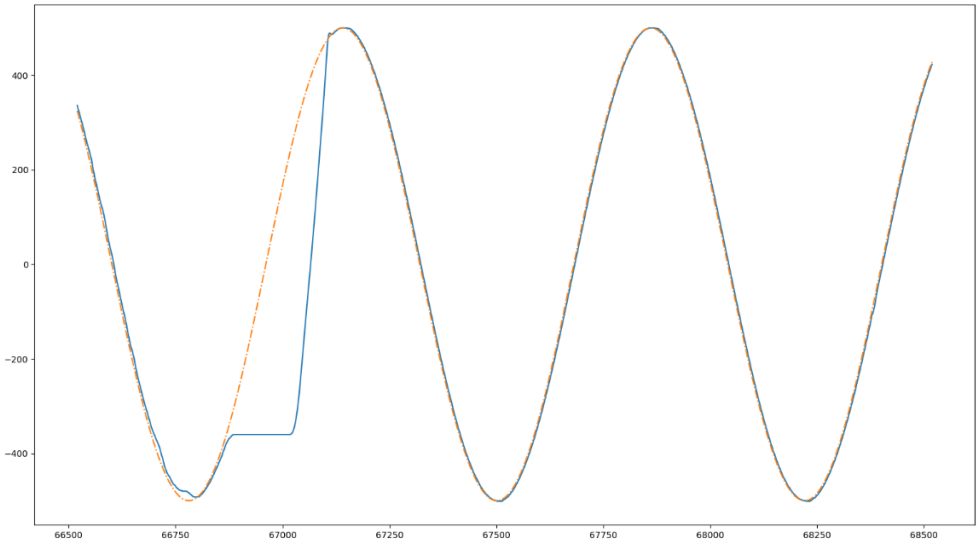

Figure 13: PID Controller Plot for Motors.

When this system was working, we were able to test it by creating a demo application involving real time plotting of the setpoint position vs actual motor encoder position. As you can see in Figure 13, the controller is able to recover and resume operation despite external influences such as holding the wheel in a fixed position and then letting go.

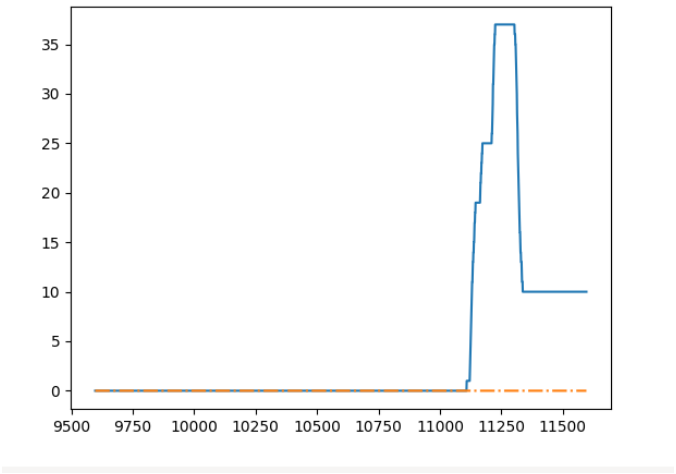

Figure 14: P Controller Plot for Encoders.

However, before the demo application was able to produce results like that, the PID loop needed to be manually tuned. Initially, the P gain is turned on and the I, D gains are turned off. The P stands for proportional, and it produces a corrective motor control signal proportional to the error between the desired setpoint and the measured motor position. Tuning the P gain resulted in the above behavior seen in Figure 14, where after moving the wheel, it would come close back to the zero position but not all the way there. This is because the error was small enough that the resultant corrective signal was not strong enough to move the motor all the way back to zero.

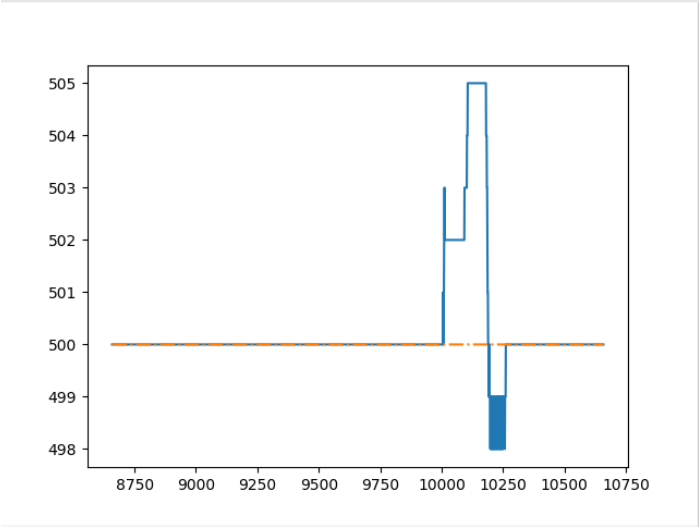

Figure 15: PI Controller Plot for Encoders.

To fix that issue, the I gain was turned on and tuned. The I stands for integral, and it produces a corrective motor control signal whose strength is equivalent to the integral of the error over time. In this way, when the motor is almost at zero but is still off by a little, there is an error term that gets integrated and the strength of the corrective motor control signal increases until it is enough to move the motor all the way to zero. This results in a plot like that shown in Figure 15.

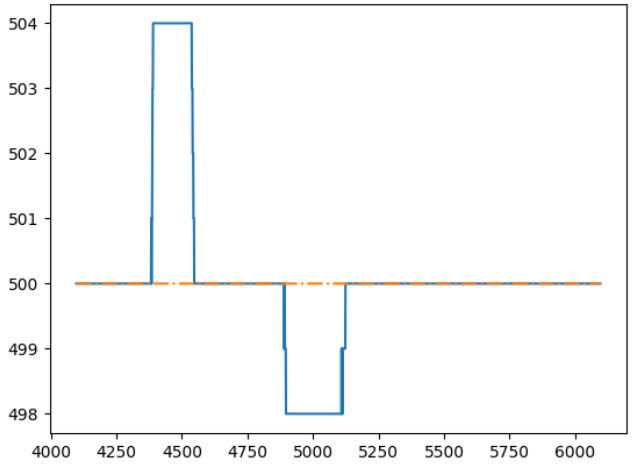

Figure 16: PID Controller Plot for Encoders.

Finally, as one might have seen in Figure 15, the motor overshot the zero position on its way back down. To prevent this overshooting, or overcorrecting by the PID algorithm, we had to tune the D gain. The D stands for derivative, and it reduces the strength of the control signal when the motor is approaching the setpoint too fast. After tuning all three gains the resultant plot shown above was obtained, where after moving the wheel, it immediately returned to zero position and did not overshoot its target. These tuned PID gains were used in the demo application mentioned previously.

GPS and IMU

The former GoScout system included Python implementations of rover control using GPS and IMU data. Using a Raspberry Pi, the software for communications and controls will be tested prior to being ported to C++ for the Kria SoM. The project implemented simple movement control onto an FPGA and CPU and is the most recent iteration of the rover project. The GoScout digital and hardware design will provide the foundation for SIMBA along with SIMBA C++ GPS and IMU implementation and other new features being built into the new Kria SoM.

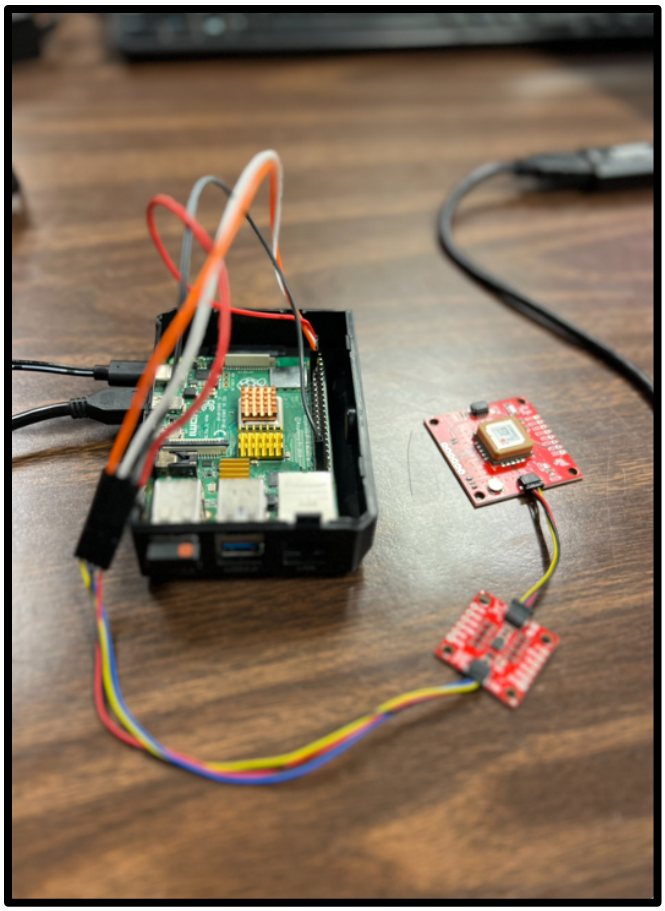

Figure 17: Test Setup of GPS (SAM_M8Q) and IMU (ICM-20948) Module for New Code Migrations.

During the rover's prototyping phase, we committed to meticulously gathering data to not only validate the rover's requirements but also to solidify its expected operational behavior. An integral part of this process involves the use of GPS and IMU systems, which will become beneficial in navigating during autonomous missions.

The GPS module within the rover interfaces via the I2C communication protocol while formatting the messages according to the u-blox protocol, which is specifically designed for GNSS applications, enabling effective communication with satellites to obtain essential data [8]. This includes verifying the presence of active satellites and accurately determining geographical coordinates (longitude and latitude) with a precision of up to seven decimal places.

To ensure the reliability of these coordinates, our initial verification process involves inputting them into Google Maps. This step confirms their accuracy by correlating them with known locations, particularly the vicinity of the capstone lab at Cal Poly EE, which is the primary site for our testing. Advancing this verification process, we also utilize the Google Maps API to generate static map images pinpointing the rover's GPS-determined position. These map representations provide a clear and visual affirmation of the GPS module's accuracy, as exemplified in Figure 18, which illustrates the rover's location via a Google Maps image.

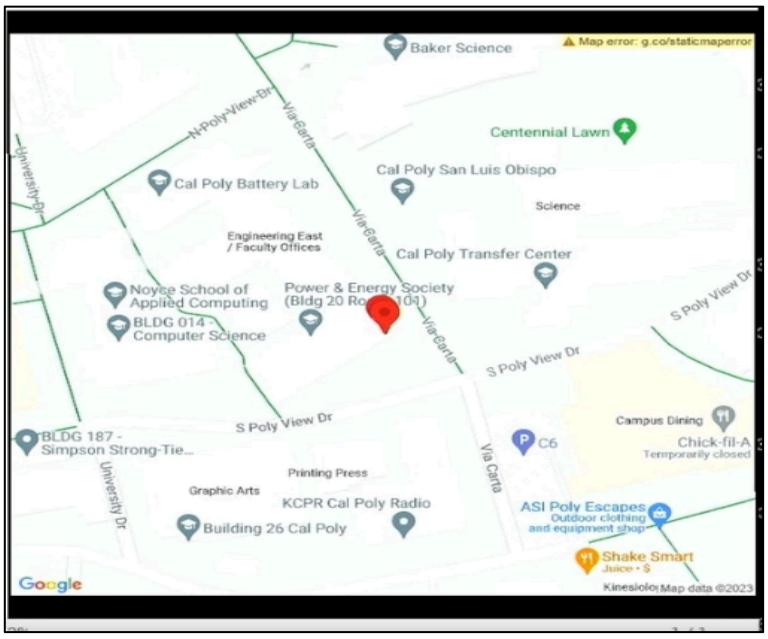

Figure 18: Google Maps Images with Markers Generated from GPS Coordinates.

Incorporating the GPS was a significant milestone, and alongside this, the IMU C++ libraries also demonstrated their efficacy. This was evidenced through the successful retrieval and confirmation of nine critical data points encompassing acceleration, magnetometer, and gyroscope readings across the X, Y, and Z axes. To facilitate this, various control registers were meticulously configured to accurately read and interpret the necessary bytes of data. This data was then dynamically plotted in real-time through a dedicated C++ library compatible with Matplotlib, providing a visual depiction of the three measurements.

For functional testing, the IMU module was manually manipulated — rotated and moved — to actively observe the corresponding fluctuations on the graph. This real-time responsiveness is illustrated in the first three figures found in the appendix. While these graphs currently serve a limited purpose, as the rover has yet to traverse around the campus or similar environments, they lay the groundwork for future data visualization when the SIMBA rover commences its mobility.

In summary, the ongoing tests with the IMU and GPS have yielded a preliminary level of accuracy, offering valuable insights into what can be anticipated in the final system. These insights are particularly instrumental in shaping our strategy for the rover's control loop. The full-scale validation and calibration of this control loop, crucial for precise movement, will be conducted once the entire system is seamlessly integrated onto the rover. This final step will be pivotal in ensuring the rover's operational efficacy and reliability.

PetaLinux

With the software team setting up C++ libraries for I2C communication with the GPS and IMU, the digital design team needed to incorporate a new I2C bus to the block design which can be seen in Figure 10. The GPS and IMU were originally tested on a Raspberry Pi so that the software team was not blocked by the firmware or hardware teams. However, when trying to incorporate them into the rover, several roadblocks were hit. Since the new I2C block is routed through the FPGA, it is essentially adding a new

hardware bus to the K26 SoM that the operating system (OS) is not aware of. So, a modified version of the Linux operating system must be created. This was the only option that the digital design team had because the onboard I2C bus, which was not routed through the FPGA, was not exposed for connection to external hardware on the Kria KR260 development board.

To create this new operating system, there is a program called PetaLinux [17] which can be configured with a specific hardware description from Vivado to create a bare-bones Linux operating system that will recognize the new I2C bus. It is a powerful tool, but it is also extremely complicated and lacks thorough instructional documentation. This software has been a block in trying to incorporate the GPS and IMU to the rover.

The first roadblock hit was a problem with the newest version of PetaLinux and the Kria KR260 board support package (BSP), both of which are version 2023.1. There are some missing dependencies in the board support package that PetaLinux looks for in version 2023.1, so it was necessary to downgrade to version 2022.2 for both items. After reinstalling and retesting the hardware description file in PetaLinux, a project was successfully built, and a boot image was created.

Although a project was successfully built and configured, when attempting to boot the Kria KR260 from the new OS image, the output from the board hung on “Starting Kernel….” This may be a problem with the current version of Ubuntu. The latest release of Ubuntu, version 22.04 LTS (which is what the digital design team was using), may not be supported by PetaLinux. The next step would be to set up a new virtual environment with a past version of Ubuntu and retry the PetaLinux build. Although the new PCB could be made to route the existing hardware I2C bus, PetaLinux would still be necessary for a real-time operating system (RTOS), so it is important that a successful boot image is created. The digital design team is currently blocked at this point in the incorporation of the GPS and IMU to the rover.

5 Management Plan

5.1 Development Teams

To facilitate the development of the SIMBA platform, three teams have been created to focus development efforts into three main branches – software, hardware, and digital design, as depicted in Figure 19 below.

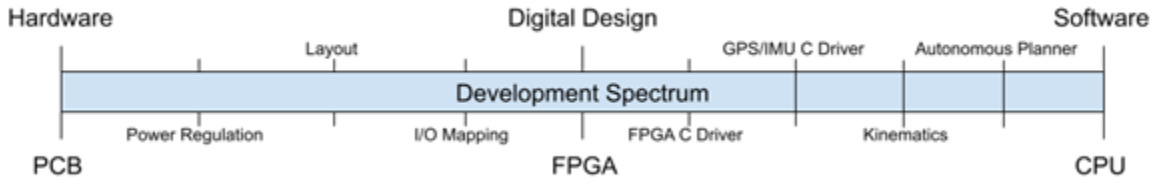

Figure 19: SIMBA Development Spectrum

On the development spectrum, project deliverables are placed in relation to the three main branches. Once defined, group members organized into teams according to their strengths and confidence in ability to complete tasks. Below are the resultant teams:

Hardware Team

Sepp Williams	Engineer, PCB Design

Digital Design Team

Ian Beck	Engineer, Firmware Design
Curtis Bucher	Engineer, Firmware Design
Braedan Kennedy	Project Lead, Software Design

Software Team

Brian Nguyen	Engineer, Software Design
Luis David Garcia	Project Liaison, Software Design

The hardware team is focused on the primary project deliverable of designing and producing a motherboard to connect all electrical components of the system. Only one engineer is currently required for this task as the initial deliverable consists of only a circuit schematic.

The digital design team is focused on the primary project deliverable of transferring motor control functionality to the new FPGA. This task requires the most research and development as the Kria KR260 platform is almost entirely different from the previous PYNQ-Z1 platform and most software development is dependent upon its functional operation.

The software team is focused on the primary project deliverable of updating and improving the C code responsible for interfacing with the FPGA and other peripherals.

All teams work together closely with each other to forward their primary objective. Some tasks require more collaboration than others – such as the development of the FPGA C driver which involves both the software and digital design team.

5.2 Management Positions

In addition to teams, there are two management positions – project lead and project liaison. The project lead coordinates the efforts of the three development teams, hosts meetings, keeps track of overall progress, and coordinates with the project liaison. This position requires in-depth knowledge concerning all technical aspects of the project. The project liaison is the public interface to the development team. Often called the client contact, the project liaison establishes and maintains communication channels with all external contacts such as the client, project manager, and other engineering teams. This position requires setting up meetings, sending emails, and communicating with the rest of the team.

Currently, there are no explicit financial management roles as this project is not expected to be making many purchases. Also, there are no explicit product verification roles as each team is responsible for producing very different products. When financial or product verification tasks are required, teams are expected to handle it themselves or with oversight from the project lead if needed.

6 Project Schedule

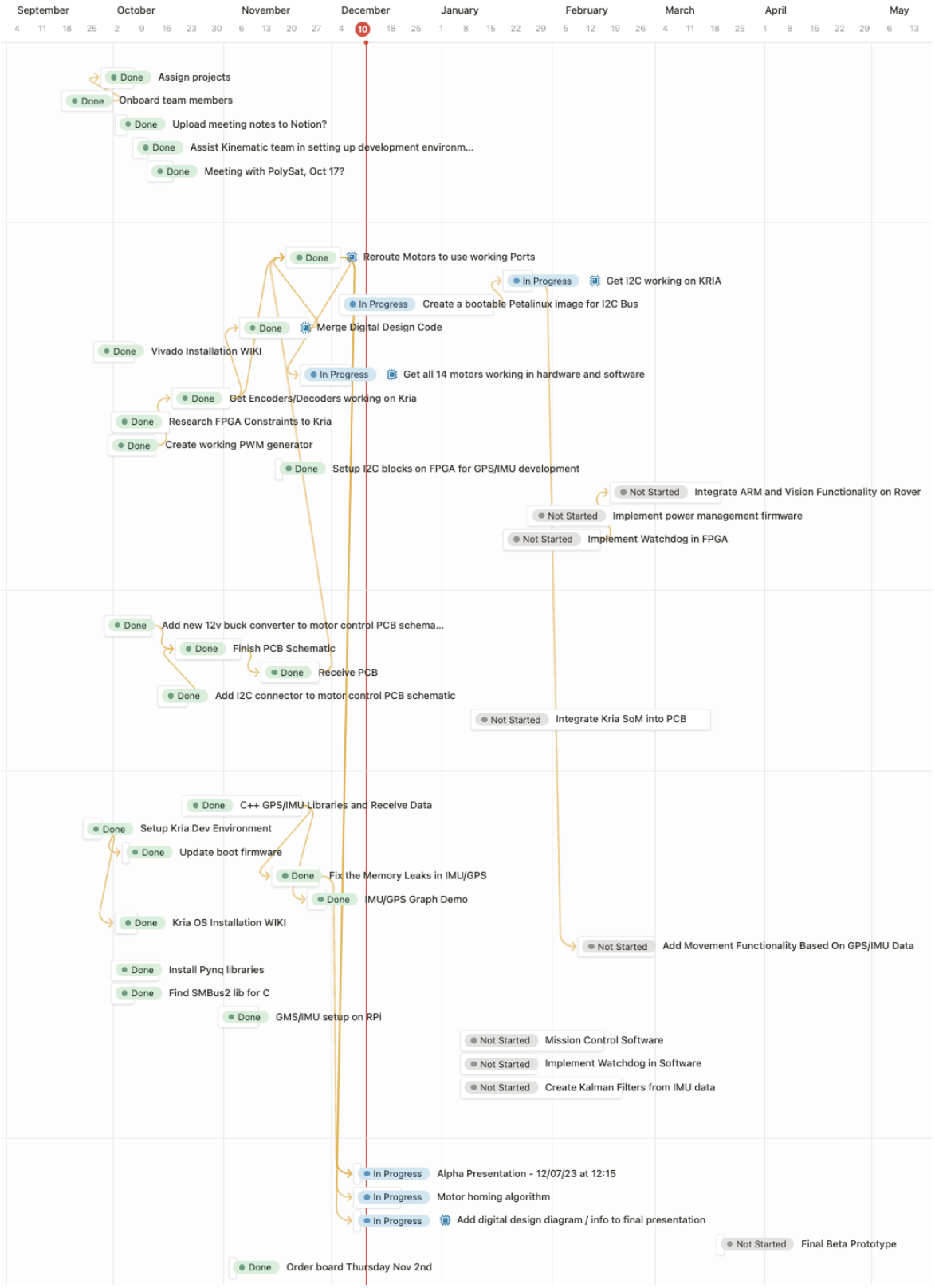

Figure 20: Gantt Chart.

Team Member	Tasks	Expected Work Hours
Braedan Kennedy	Researching Kria K26	8
	Begin development on mission control software	10
	Begin development on autonomous navigation software	10
	Implement watchdog timer in FPGA and software	3
Curtis Bucher	Debug existing motor control issues	6
	Adapt motion API to work with arm and vision software.	8
	Evaluate hardware acceleration for arm and vision teams	4-16
Ian Beck	Create bootable PetaLinux image that recognizes I2C bus.	8-10
	Incorporate GPS/IMU C++ libraries into development environment.	4
	Implement rover control functionality based on GPS/IMU data	16
Sepp Williams	Designed new motor control PCB schematic and layout with updated I/O and voltage regulator.	14 (Completed)
	Finished PCB manufacturing for two boards and validated I/O, voltage regulation, and motor control	12 (Completed)
	Design a full control PCB schematic and layout with K26 SoM	18-24
	Manufacture and test the new K26 SoM PCB	12-16
Brian Nguyen	Validate the IMU library on the KRIA board so that it is successfully integrated.	8
	Creation of Kalman filters for IMU acceleration, gyroscope, and magnetometer data.	12
	Create documentation for Kalman filters.	3
Luis Garcia	Validate the GPS library on the KRIA board so that it is successfully integrated.	8
	Creation of Kalman filters for GPS coordinate data.	12
	Coordinate with Tyler regarding any integrations he would like to see with Arm and Vision teams	2

Figure 21. Estimated Future Development Tasks and Hours Table.

7 Appendices

7.1 Accelerometer Data Plot

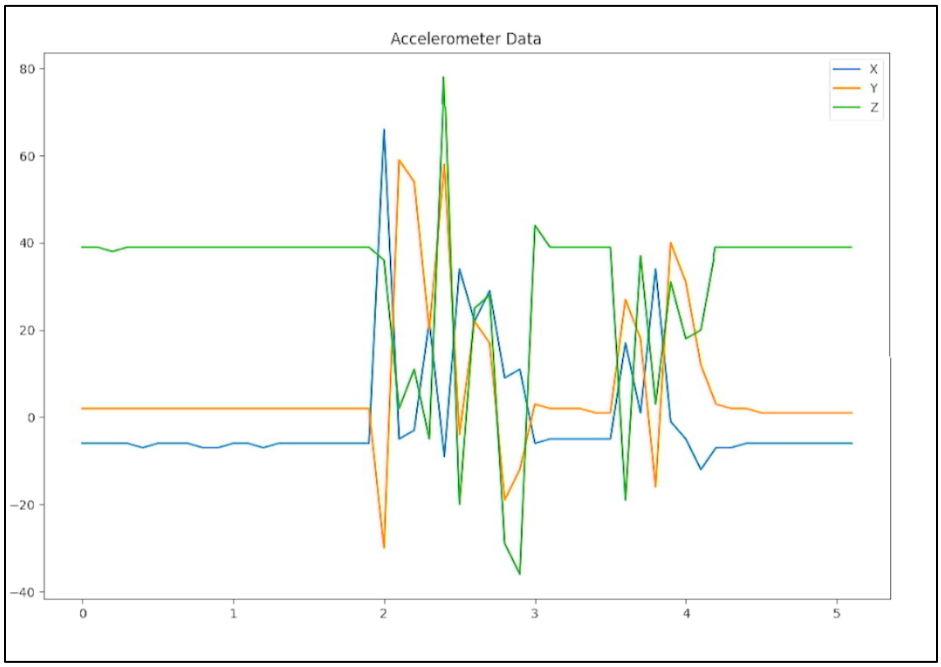

Figure 22: Accelerometer Data from IMU Displaying Acceleration of X, Y, and Z Axis in units of [m²/s].

7.2 Gyroscope Data Plot

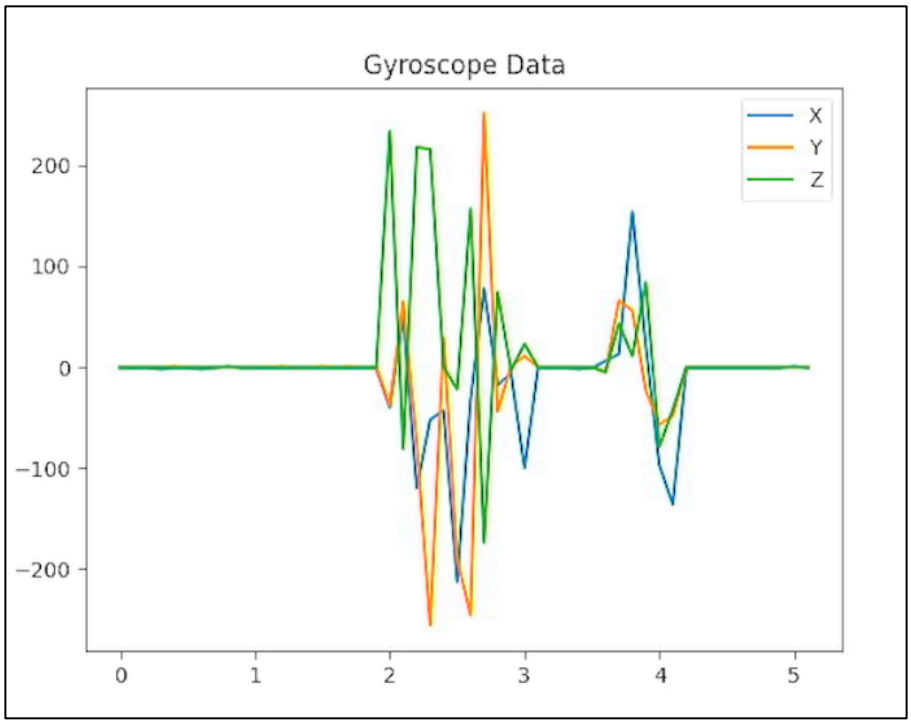

Figure 23: Gyroscope Data from IMU Displaying Rotation in the X, Y, and Z Axis' in units of [Degrees/s].

7.3 Magnetometer Data Plot

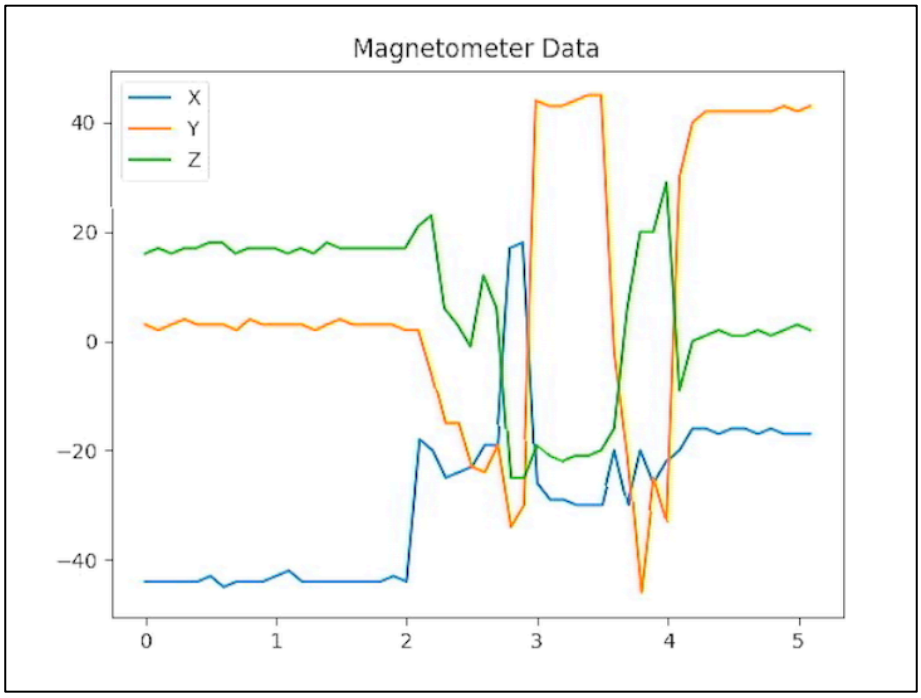

Figure 24: Magnetometer Data from IMU Displaying Acceleration of X, Y, and Z Axis in units of [µTesla/s].

7.4 Bill of Materials

Component	**Cost per Part**	**# of Parts**	**Total**
PCB Manufacturing	$ 5.92	1	$ 5.92
0.1uF 50V Capacitor	$ 0.10	42	$ 4.20
22uF 63V Capacitor	$ 1.27	14	$ 17.78
30K Resistor	$ 0.10	14	$ 1.40
2K Resistor	$ 0.48	14	$ 6.72
DRV8871 Motor Controller	$ 2.24	14	$ 31.36
2x3 Pin Header	$ 0.15	14	$ 2.16
2x6 Pin Header	$ 0.29	4	$ 1.14
2.1mm Power Plug	$ 5.00	1	$ 5.00
40 Pin Raspberry PI HAT Header	$ 4.52	1	$ 4.52
XT60 Battery Connector	$ 1.50	1	$ 1.50
QWIIC Connector 4-Pin	$ 0.56	1	$ 0.56
Slide Switch 5A 120V	$ 3.45	1	$ 3.45

Pololu 12V 4.5A Buck Converter	$ 24.95	1	$ 24.95
Kria KR260 Robotics Starter Kit	$ 349.00	1	$ 349.00
Jumper Wire Male to Female 6" 28AWG Bulk	$ 1.95	1	$ 1.95
			$ 461.61

Figure 25: Build of Materials (BOM) of current iteration of SIMBA control electronics PCB and Kria KR260 Dev-board.

8 Sources

[1] B. Nguyen, B. Kennedy, C. Bucher, J. Williams, L. Garcia, I. Beck, “Solar Autonomous ROACH Background Research,” https://tinyurl.com/2erhdpp5 (accessed Oct. 26, 2023)

[2] T. Bovenzi, E. Picazo, S. Sehnert, “Cal Poly R.O.A.C.H,” Cal Poly Computer Engineering Department, San Luis Obispo, CA. June 2023.

[3] L. La Rocca, Melopero SAM-M8Q Arduino Library, https://github.com/melopero/Melopero_SAM-M8Q/tree/master (accessed Oct. 26, 2023).

[4] B. Alsadik, “Kalman filter,” Kalman Filter - An Overview, https://www.sciencedirect.com/topics/earth-and-planetary-sciences/kalman-filter (accessed Oct. 26, 2023).

[5] B. Nguyen, B. Kennedy, C. Bucher, J. Williams, L. Garcia, I. Beck, “Archetypes and Use Cases,” https://tinyurl.com/4k2sk4a3 (accessed Oct. 26, 2023)

[6] “SAM-M8Q module Easy-to-use u-blox M8 GNSS antenna module Smart antenna module for easy and reliable integration.” Accessed: Dec. 05, 2023. [Online]. Available: https://content.u-blox.com/sites/default/files/SAM-M8Q_ProductSummary_%28UBX-15027452%29.pdf

[7] “ICM-20948 Datasheet,” *TDK InvenSense*. https://invensense.tdk.com/download-pdf/icm-20948-datasheet/ (accessed Dec. 05, 2023).

[8] “u-blox 8 / u-blox M8 Receiver Description Including Protocol Specification.” Accessed: Dec. 05, 2023. [Online]. Available: https://docs.rs-online.com/8e7c/0900766b815aef70.pdf

[9] “SparkFun u-blox Arduino Library,” *GitHub*, Nov. 08, 2023. https://github.com/sparkfun/SparkFun_Ublox_Arduino_Library (accessed Dec. 05, 2023).

[10] “SparkFun_ICM-20948_ArduinoLibrary/src/ICM_20948.h at main · sparkfun/SparkFun_ICM-20948_ArduinoLibrary,” *GitHub*. https://github.com/sparkfun/SparkFun_ICM-20948_ArduinoLibrary/blob/main/src/ICM_20948.h#L34 (accessed Dec. 05, 2023).

[11] “Using I2C in C++ on Raspberry Pi,” *Raspberry Pi Stack Exchange*. https://raspberrypi.stackexchange.com/questions/33485/using-i2c-in-c-on-raspberry-pi (accessed Dec. 05, 2023).

[12] “I2C Device Interface — The Linux Kernel documentation,” *www.kernel.org*. https://www.kernel.org/doc/html/v5.4/i2c/dev-interface.html (accessed Dec. 05, 2023).

[13] A. de Freitas, “Matplot++,” *GitHub*, Dec. 04, 2023. https://github.com/alandefreitas/matplotplusplus (accessed Dec. 05, 2023).

[14] M. Guntupalli and M. Simek, “Xilinx Device Tree Overlay,” GitHub, https://github.com/Xilinx/linux-xlnx/tree/master/arch/arm/boot/dts (accessed Dec. 2, 2023).

[15] “Kria K26 SOM Thermal Design Guide (UG1090),” *AMD*, Nov. 19, 2021. https://docs.xilinx.com/r/en-US/ug1090-k26-thermal-design/Empirical-Correlations-to-Determine-the-Thermal-Solution-Performance (accessed Dec. 07, 2023).

[16] B. Kennedy, “SIMBA Digital Design Achitecture,” SIMBA Team, Nov. 2, 2023. SIMBA Digital Design Architecture.docx (accessed Nov. 2, 2023).

[17] Xilinx, “Petalinux Tools,” AMD, https://www.xilinx.com/products/design-tools/embedded-software/petalinux-sdk.html (accessed Nov. 18, 2023).