Post-Quantum Cryptographic Embedded System for Securing Deep Space Applications

Kepler-452B

Team Members: Michael Luna, Akif Dhar

Faculty Advisor: Professor: Mohamed EL-Hadedy

Student Contact: Michael Luna - michael.luna37@gmail.com

California State Polytechnic University, Pomona Department of Electrical and Computer Engineering, College of Engineering

X___________________

Signature of the sponsoring faculty advisor verifying that he has read and reviewed the paper prior to submission to NASA.

CONTENTS

Kepler-452b Post Quantum Cryptography

Abstract—**With the advent of quantum computers, there is a growing need for post-quantum cryptographic primitives to secure communication channels against the threat of Shor's algorithm. In this paper, we present a project plan for developing a dynamic reconfigurable processor that can provide post-quantum security to deep space devices, such as drones and spaceships involved in the Artemis mission. Our proposed solution combines XMSS stateful hash-based signature algorithms and other post-quantum cryptographic primitives, running on low-power, low-latency edge-computing clusters. We describe our initial project schedule, budget justification, and current research, which includes exploring the efficiency of post-quantum cryptographic solutions on drones and testing the security of our proposed solution against a possible Shor algorithm attack. Our innovation lies in creating an efficient implementation of XMSS stateful-hash-based signature algorithms on both software and hardware computing modules, examining both Diffie-Hellman and X3DH key exchange protocols, and developing efficient test cases for monitoring power consumption and energy usage. Overall, our project aims to provide a robust and secure post-quantum cryptographic solution to protect deep space devices and applications against the threat of quantum computing.**

I. Introduction

As technology continues to advance, the need for secure communication channels becomes increasingly crucial. With the emergence of quantum computing, current cryptographic schemes are at risk of being broken, threatening the security of sensitive data and critical applications. Post-quantum cryptography offers a solution to this problem by providing cryptographic primitives that are resistant to attacks by both classical and quantum computers. In this paper, we present a project plan for implementing post-quantum security on embedded IoT devices using AES encryption and XMSS signatures. Our proposed solution focuses on hardware implementation, utilizing the PYNQ platform, and incorporates a color detection algorithm for drone verification. Our goal is to provide a robust and secure post-quantum cryptographic solution to protect embedded IoT devices used in critical applications against the threat of quantum computing.

II. Project Management

1) Gantt chart: The project schedule was developed using a Gantt chart, which was used to track project milestones and deadlines.
2) Agile methodology: The team used agile methodology to manage the development of the software components of the project. The team used daily stand-up meetings to discuss progress, identify issues, and plan the next steps.
3) Project management software: The team used project management software, such as Trello, to track the progress of the project and to manage tasks.
4) Communication plan: The team developed a communication plan to ensure that all team members were informed of project updates, issues, and deadlines. The communication plan included regular team meetings, email updates, and online collaboration tools.
5) Risk management plan: The team developed a risk management plan to identify and mitigate potential risks to the project. The plan included strategies for managing technical, schedule, and resource risks.
6) Change management plan: The team developed a change management plan to manage changes to project scope, schedule, and budget. The plan included procedures for assessing change requests, identifying impacts, and obtaining approval.

A. System Requirements

Functional Requirements:

1) Two drones should be able to communicate with the UGV on the ground.
2) The main drone should be able to sign and encrypt a message over telemetry using XMSS and AES.
3) The UGV should be able to verify the signature and decrypt the message using XMSS and AES.

4) The enemy drone should attempt to intercept the encrypted message and verify the signature.
5) The UGV should use lidar and object detection to move towards the drone that successfully verified the signature.
6) If the drone does not verify the message, the UGV should use lidar and object detection to launch a missile at the enemy drone.

Performance Requirements:

1) The system should be able to perform real-time signature verification and decryption.
2) The system should be able to communicate between the two drones and the UGV with low latency.

Usability Requirements:

1) The system should have a user-friendly interface for the operator to control the UGV.
2) The system should provide clear feedback to the operator regarding the success or failure of the verification and decryption process.

Security Requirements:

1) The system should ensure quantum security through the use of XMSS and AES encryption and signature algorithms.
2) The system should prevent unauthorized access to the communication channel and ensure the confidentiality, integrity, and availability of the data transmitted.

Technical Requirements:

1) The main drone should be equipped with a PYNQ-z1 computing module to perform the necessary computations.
2) The UGV should be equipped with an Arduino Mega to control the servos and motors necessary to move the drone and external missile arm.
3) The system should be able to interface with the lidar and object detection sensors for the UGV.
4) The system should be able to withstand environmental conditions such as wind and rain.

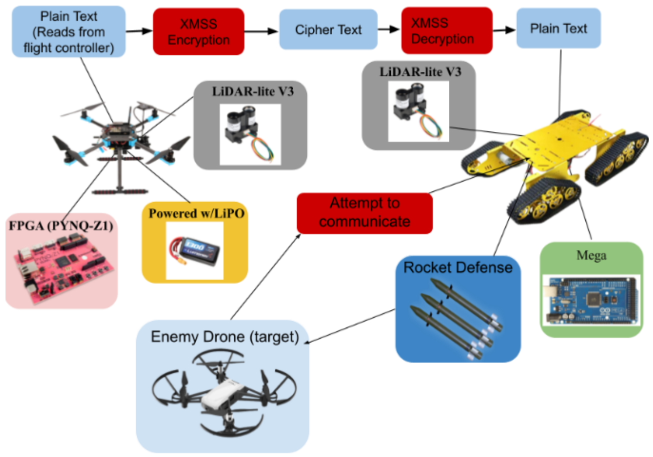


Figure 1: System Requirements

B. PDR

During the Preliminary Design Review, we integrate previous designs and conduct research on relevant online documentation to enhance this project's effectiveness before its execution. While building upon these designs, we ensure the originality and practicality of this project for the Artemis mission. Our unique project aims to address the post-quantum cryptography challenge and provide NASA with a hardware-based solution, rather than a mere simulation on a computer. By concentrating on XMSS encryption as a post-quantum signature on the PYNQ platform and utilizing other embedded microcontrollers, we are employing a cutting-edge and pioneering method for post-quantum cryptography.

C. CDR

During the Critical Design Review, we identified several areas that required changes and modifications based on our findings. One of the key issues we discovered was a lack of visual representation for the successful implementation of the XMSS algorithm in our Preliminary Design Review. Additionally, the use of multiple UAVs lacked interactive capabilities, so we pivoted to using a UGV and a UAV that could communicate securely using XMSS encryption. Once the UAV passed the encryption, it would land on the UGV. To enhance the scenario, we added a third UAV to demonstrate what happens when a UAV fails to verify the encryption properly. To address this, we implemented a missile guiding system that would track the UAVs until the encryption was either passed or failed. If the UAV was deemed friendly and successfully verified the signature, the missile system would not be triggered. However, if the UAV was hostile and failed to verify the signature, the missile system would be triggered and launched.

III. Schedule

Appendix A contains a chart outlining the projected timeline for the Kepler-452b project. The planning phase and the determination of group members took place between August 25th and September 3rd. Concurrently, we began researching implementations of XMSS and LMS stateful hash-based signature algorithms and other post-quantum cryptographic primitives. From September 21st to October 18th, we prepared the project proposal while simultaneously designing the implementation of the XMSS algorithms and the structure and designs of the UAVs. This design phase continued until December 2nd, with a preliminary design review on November 21st. Between December 2nd and 31st, we began thinking about the more critical aspects of our project and trying to determine the best way to visualize the XMSS encryption being successful, this is when we conducted our first critical design review. Also during this time we purchased materials and built, tested, and debuged our software and hardware up to March 31st.

IV. Budget

A total of 1500 dollars was allocated for the project's goals. The initial design included two UAVs that utilized a PYNQ-Z1 board and a Turing Pi Cluster, both of which would be mounted on the UAVs. However, due to the large size of the Turing Pi, the design was modified to include one UAV and one UGV. Kits for the drone and basic structures for the UGV were purchased to minimize mechanical engineering efforts and allocate resources towards creating a successful XMSS encryption algorithm. To keep costs low, 3D printed casings were used for microcontrollers and PYNQ-Z1 boards. Appendix B showcases the exclusive use of Amazon as a vendor, except for the friendly UAV, due to their timely delivery and quality assurance. Orders were split into two stages to allow for a flexible budget and enable revisions to the project design. This separation of orders facilitated better control over materials delivered. Through thorough analysis, defective products were identified and replacements were obtained from the seller.

V. System Hierarchy

A diagram of the system Hierarchy can be found in Appendix C. In this we outline the different aspects of the parts of the project.

A. *Unmanned Ground Vehicle(UGV)*

1) Electrical Subsystem: The Unmanned Ground Vehicle (UGV) project utilized various types of boards during its construction. Initially, the Arduino UNO was used as a simple and efficient tool to test electrical and hardware components. However, due to its limited capabilities, a more complex board was chosen for the later stages of the project.

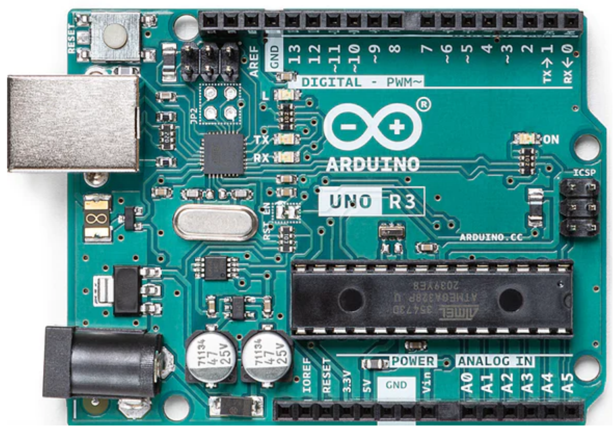


Figure 2: Arduino UNO

The Arduino UNO board offered a limited number of General Purpose Input and Output pins(GPIOs), which restricted our ability to utilize multiple sensors such as GPS, hall sensors, telemetry communication, Li-DAR, and the Pixy2 camera simultaneously. To achieve the objectives of the project more effectively, we needed a more advanced board. Therefore, we transitioned to the BeagleBone Black board, which provided us with enhanced capabilities to integrate these peripherals and fulfill the requirements of the project.

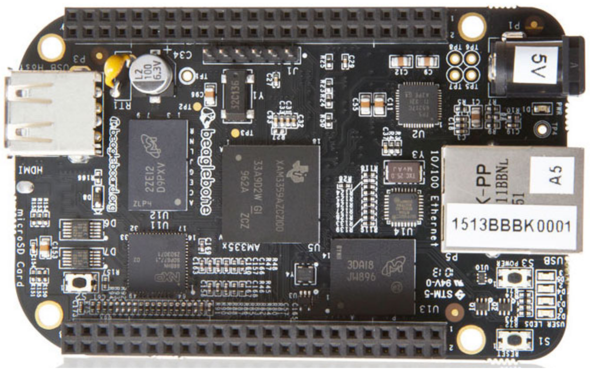


Figure 3: BeagleBone Black

The BeagleBone Black, a competitor to the Raspberry Pi, has its own Linux kernel operating system that was utilized for decryption and verifying signatures. With the adoption of the BeagleBone Black board, there was no longer a need to use the Turing Pi Cluster. One of the challenges with the Turing Pi Cluster was the power consumption required to operate a board using a battery, making it impractical to mount on the UGV. Leveraging the Debian operating system on the BeagleBone Black allowed for efficient algorithm decryption and implementation. The BeagleBone Black proved to be more of a challenge than anticipated and the challenges faced were largely due to current requirements that the board required, and properly being able to integrate the peripherals with the board. In order to successfully implement the beagle bone black and the other sensors we were required to use two buck converters with high current ratings. The first buck converter was used to control step down the voltage from 16.6v to 12v in order to provide sufficient power to the motors and their drivers.

Figure 4: 16v to 12v Buck Converter

The second buck converter was selected primarily for its high current rating and the requirement to reduce the 12V voltage to 5V for peripherals, including GPS and servos. The buck converter played an essential role as the BeagleBone Black was powered via a 5V barrel adapter with a current rating of 2A. The high current rating, along with the current demands of the rest of the system, would cause smaller voltage regulators to malfunction. Moreover, the selection of the appropriate buck converter was essential in ensuring that the BeagleBone Black board and other components were operating within their recommended power range. The second buck converter played a critical role in regulating the voltage to the peripherals to prevent damage or malfunctioning due to over voltage. In addition, it was necessary to select a converter with a high current rating to meet the power demands of the entire system. Without the correct buck converter, the system would not have functioned as intended, potentially causing significant delays and additional expenses. Overall, the integration of the BeagleBone Black board and the necessary peripherals required careful consideration of power requirements, compatibility, and safety to achieve a functional and reliable system.

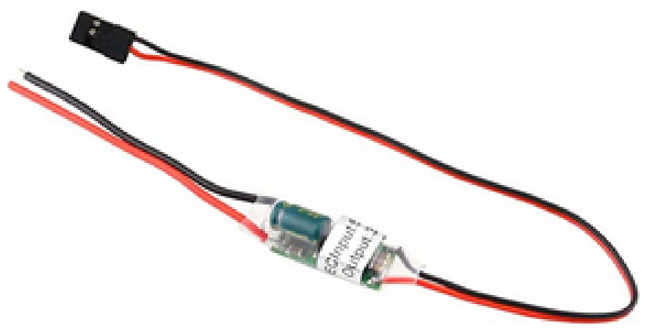


Figure 5: 12v to 5v Buck Converter

Overall he issues associated with the beagle bone black caused us to rethink the use of the board and we decided to utilize the Arduino mega 2560 instead because it still allowed us to use our peripherals with a lower current requirement of 1A.

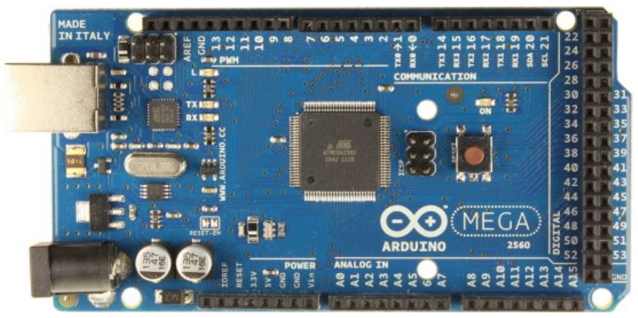


Figure 6: Arduino Mega 2560

The use of the Mega allowed us to still operate all of our peripherals but with a lower current requirement. The Mega board was a crucial addition to the system, as it allowed for the integration of multiple peripherals without exceeding the maximum current capacity of the system. By utilizing the Mega board, the current requirements for operating the peripherals were significantly reduced, enabling the system to function more efficiently and reliably. Additionally, the Mega board provided a broader range of general-purpose input/output pins, allowing for the integration of additional sensors and components with ease. Furthermore, the Mega board was compatible with the existing software and firmware that we utilized with the Arduino UNO, which meant that the transition to this board was seamless and required minimal changes to the overall system design. Overall, the Mega board was a crucial component in the successful integration of the peripherals and enabled the system to operate within the desired power range, providing reliable and accurate data for the UGV. For the motor drivers, we utilized two L298N motor drivers that were driven by the Mega.

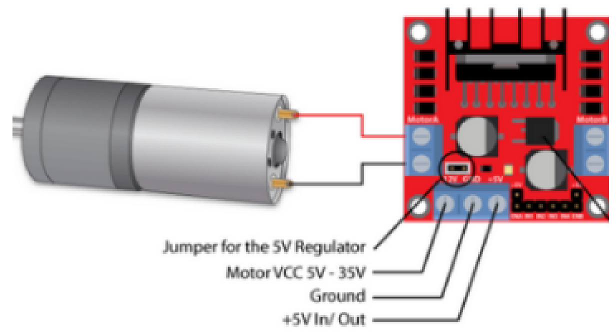


Figure 7: L298N Motor Configuration

This type of motor driver allowed for us to control the forward and backward control of the motors, and utilized one Pulse Width Modulation pin from the Arduino Mega. We used a configuration from figure 7 to control the DC motors.

2) Software Subsystem: For the Software for the UGV, we focused on creating simple and efficient software that would effectively implement all of the systems we required. The main IDE that we used was the Arduino IDE. When utilizing the Arduino UNO and the Arduino Mega the Arduino IDE was easy to implement and test our code. When we were using the beagle bone black we saved all of our information on the Cloud 9 IDE associated with the beagle bone black. The most important software system we used were the LiDAR and the Pixy2 camera. The Pixy2 camera was used for object tracking using color components.

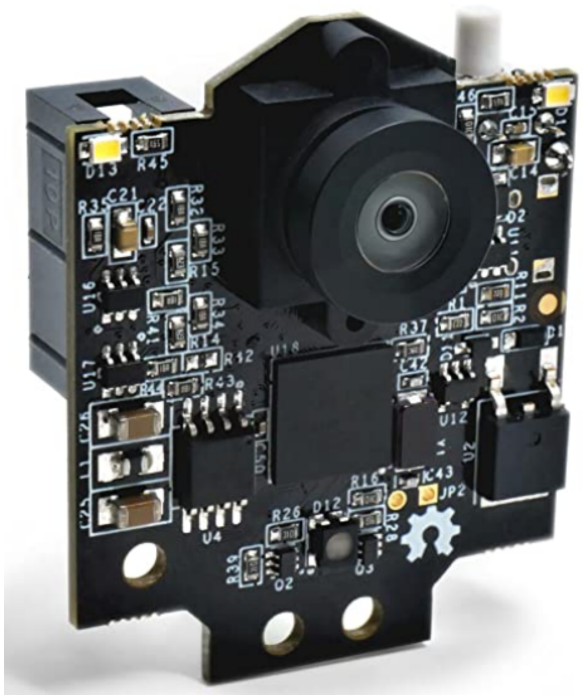


Figure 8: Pixy2 Camera

The LiDAR Lite v3 camera was a key component in effectively tracking and detecting objects from long range. Unlike the Pixy2 camera, which requires sufficient light and can only detect objects within a limited range of approximately 10 feet, the LiDAR sensor was critical for detecting incoming aerial vehicles at distances of up to 40 meters. The integration of the LiDAR sensor enabled the UGV to accurately and reliably detect objects in its path, providing essential data for the navigation and avoidance systems. The use of the LiDAR sensor also allowed for precise mapping and localization of the UGV in real-time, providing essential information for successful autonomous navigation. Overall, the incorporation of the LiDAR sensor proved to be an essential aspect of the UGV's sensory system, enabling it to operate effectively and safely in various environments and scenarios.

Figure 9: LiDAR Lite v3

The UGV was equipped with a pan and tilt mechanism that incorporated both the Pixy2 camera and LiDAR sensor. This system served as a critical precursor to the missile launch sequence, detecting and determining whether incoming UAVs were hostile or friendly and taking appropriate action. The Pixy2 camera, along with the LiDAR sensor, provided essential data for object detection and classification, enabling the UGV to make informed decisions and respond quickly and effectively to potential threats. The pan and tilt mechanism allowed for precise positioning and tracking of the Pixy2 camera and LiDAR sensor, ensuring comprehensive coverage and accurate detection. Overall, the integration of the pan and tilt mechanism, Pixy2 camera, and LiDAR sensor provided a robust and reliable defense system for the UGV, enabling it to operate autonomously in various scenarios and environments.

3) Mechanical Subsystem: The UGV features a mechanical pan and tilt mechanism that was specifically designed to provide sufficient space for the incoming drone to land on the platform. This particular design was chosen to allow for the missile system to be launched from the side. In adherence to safety protocols, the missile system was not activated on campus or in any populated areas.

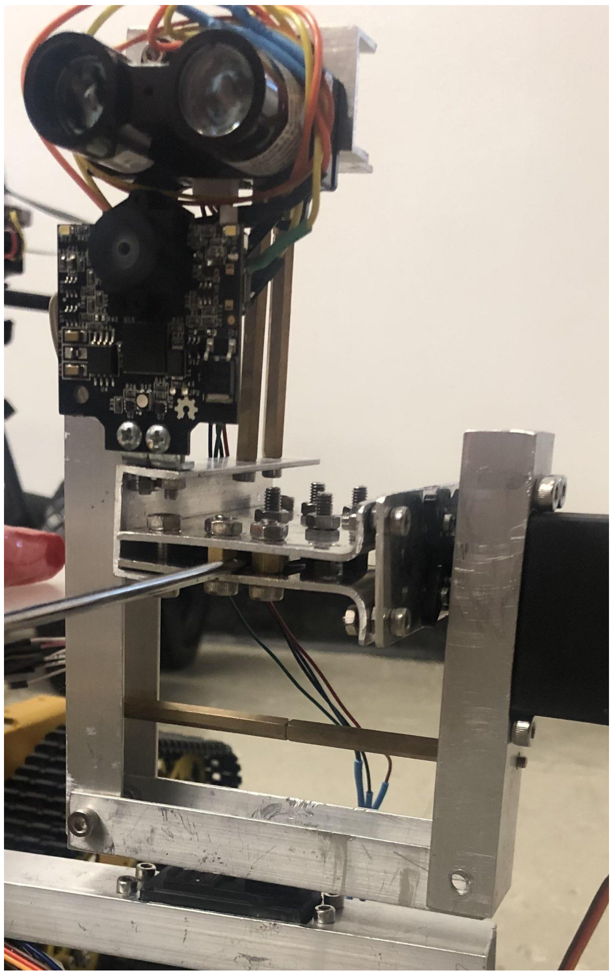


Figure 10: Pan and Tilt Mechanism

The missile system was developed solely for the purpose of simulating the UGV's response to a hostile UAV. The pan and tilt mechanism is equipped with two servos that enable a 180-degree rotation around the z-axis and a maximum of 65-degree rotation around the x-axis, giving the robotic system two degrees of freedom. It is positioned approximately 10 inches away from the UGV's center to provide a clear and unobstructed landing platform.

B. Unmanned Aerial Vehicle(UAV)

1) Electrical Subsystem: The Holybro x500 v2 kit was chosen for the main drone due to its suitable electrical subsystem. The kit offers a powerful and efficient electrical system that can easily handle the requirements of the drone, including its flight controller, telemetry, ESCs, and camera. The kit's power management system allows for the use of a 4s lipo battery to power all the components, which is convenient and efficient.

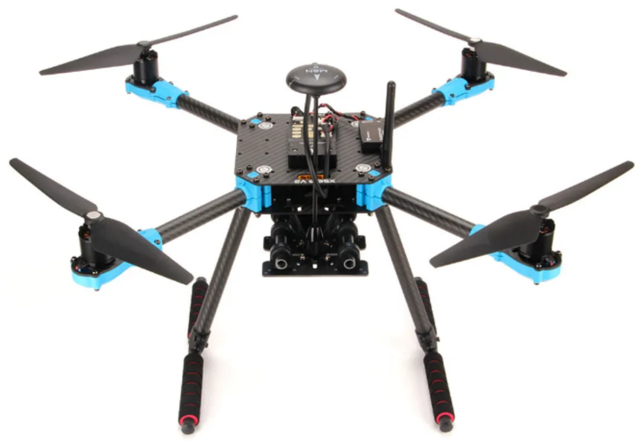


Figure 11: Holybro x500 Drone

In order to simulate a man in the middle attack we utilized the DJI Tello drone that would attempt to verify the signature generated by the Holybro drone and would then attempt to decrypt the message being sent. Upon unsuccessful verification the Friendly drone would send communication to the UGV to target the enemy drone and send a message saying it is not clear to land, it would then lock on to the enemy drone using color detection and lidar in order to launch a missle.

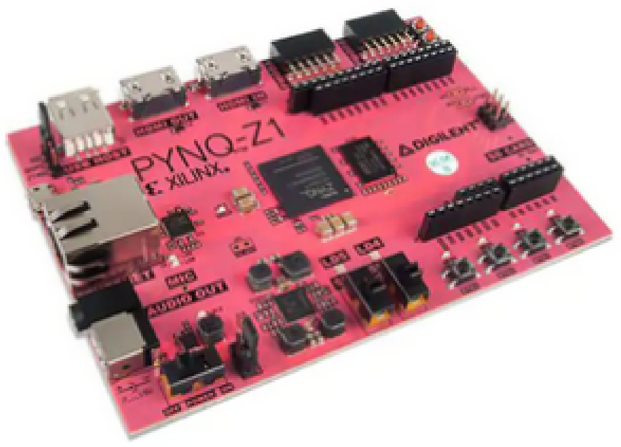


Figure 12: PYNQ-Z1

2) Software Subsystem: Our Software system for the UAV consisted of several components that included the flight control system and the security system. our flight control system was based on the Pixhawk 6C flight controleer, which provided a range of autonomous flight modes, including landing and surveying. This was configured and monitored using the QGroundControl software, which allowed us to calibrate the system and utilize the GPS real-time data. Additionally we also implemented our security system through the use of Jupyter Notebooks included in the PYNQ image, which provided a user-friendly interface for implementing cryptographic algorithms with python and its extensive libraries. The use of the PyMAVLink libraries allowed us to send and receive encrypted data via telemetry

3) Mechanical Subsystem: In terms of mechanics, we were fortunate to receive the drone as a kit, which allowed for a relatively easy assembly process. However, we did encounter issues with the landing gear's structural integrity, which necessitated the design and 3D printing of new landing gear with 100 percent infill to ensure its security. The UAV was a crucial component of the overall system, as it provided an aerial view for the UGV and allowed for better detection and tracking of incoming UAVs. Despite the initial challenges with the landing gear, the 3D printed landing gear proved to be a successful solution, and the UAV was fully functional for the project. The UAV was equipped with a GPS and a telemetry system that allowed for remote control and monitoring of its flight. The telemetry system also provided data on the UAV's altitude, speed, and location, which was used for tracking and detection purposes. Overall, the successful integration and use of the UAV added another layer of effectiveness and efficiency to the entire system.

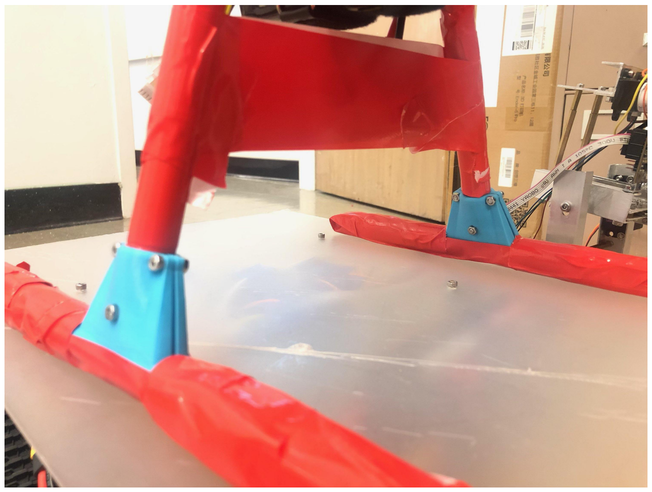


Figure 13: UAV Landing Gear

Due to the object detection limitations of the Pixy2 camera, we decided to use red tape to enhance the visibility of the drone and make it easier for the camera to detect.

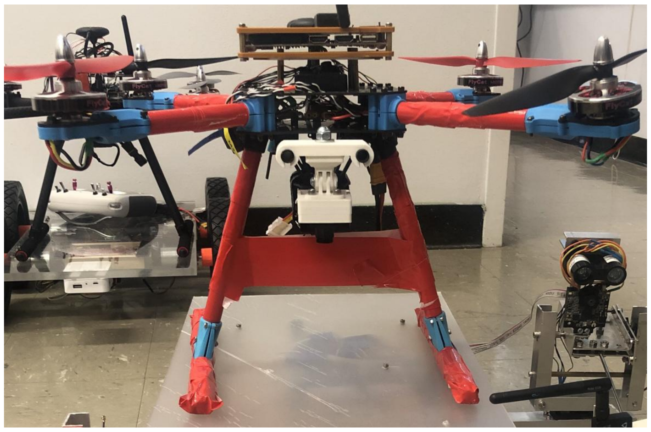


Figure 13: UAV for Object Detection

To ensure the optimal detection of the drone by the pixy2 camera, we carefully considered the color scheme for the UAV. After careful evaluation, we decided to use red as the primary color for the drone, as blue was deemed unsuitable due to its tendency to blend into the sky. Red proved to be the most effective color for differentiation from the environment and ourselves.

C. *Encryption*

Due to time constraints, we made the decision to utilize an implementation of AES encryption that leveraged the pycryptodome library for Python. This allowed us to quickly and easily implement encryption for our data, without the need for extensive development time.

D. *Signature Scheme*

The XMSS algorithm is used to sign and verify digital signatures through the use of public and private keys based on a seed value. the seed value is used to generate a binary hash tree where each leaf node represents a possible signature. The UAV in this case serves as the signer and the UGV is the Verifier.

VI. Interfaces

A. *Encryption*

To ensure overall secure communication between both the UGV and UAV, our security system utilized the American Encryption Standard. With both systems utilizing same algorithm for encryption, we are able to successfully encrypt and decrypt between systems ensuring that

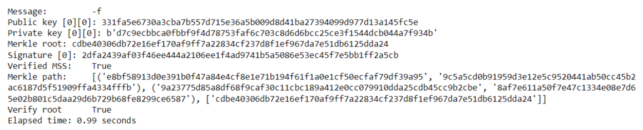
VII. Integration

A. *Encryption*

The use of the Advanced Encryption Standard (AES) to secure telemetry data transmitted between a Pynq board and an Arduino Mega is a critical aspect of the overall security architecture of the system. The AES encryption algorithm represents a key interface between the plaintext data being transmitted and the ciphertext data that is transmitted securely over an insecure channel. The successful implementation of this interface depends on the proper configuration of the encryption algorithm, including the selection of an appropriate secret key, as well as the effective integration of the encryption module with the overall telemetry system

B. *Signature Scheme*

The XMSS (eXtended Merkle Signature Scheme) algorithm was utilized as a signature scheme in the system architecture, implemented using an embedded version of the algorithm on the PYNQ-Z1 platform. The integration of XMSS was facilitated through the use of Jupyter Notebooks and a suite of cryptography libraries, enabling efficient and effective implementation of the algorithm in the system. The UAV acted as the signer and the UGV as the verifier, with signatures transmitted over telemetry using the MAVLink protocol. The integration of XMSS provided an additional layer of security to the overall system architecture, ensuring the integrity and authenticity of data transmitted between the UAV and UGV. The successful integration of XMSS was dependent on careful attention to the design and implementation of key interfaces between the various software and hardware components of the system, as well as the effective coordination of these components to ensure reliable and secure operation.



```
Message:        -f
Public key [0][0]: 331fa5e6730a3cba7b557d715e36a5b009d8d41ba27394099d977d13a145fc5e
Private key [0][0]: b'd7c9ecbbca0fbbf9f4d78753faf6c703c8d6d6bcc25ce3f1544dcb044a7f934b'
Merkle root: cdbe40306db72e16ef170af9ff7a22834cf237d8f1ef967da7e51db6125dda24
Signature [0]: 2dfa2439af03f46ee444a2106ee1f4ad9741b5a5086e53ec45f7e5bb1ff2a5cb
Verified MSS:   True
Merkle path:    [('e8bf58913d0e391b0f47a84e4cf8e1e71b194f61f1a0e1cf50ecfaf79df39a95', '9c5a5cd0b91959d3e12e5c9520441ab50cc45b2
ac6187d5f51909ffa4334fffb'), ('9a23775d85a8df68f9caf30c11cbc189a412e0cc079910dda25cdb45cc9b2cbe', '8af7e611a50f7e47c1334e08e7d6
5e02b801c5daa29d6b729b68fe8299ce6587'), ['cdbe40306db72e16ef170af9ff7a22834cf237d8f1ef967da7e51db6125dda24']]
Verify root     True
Elapsed time: 0.99 seconds
```

Figure 13: Signature and verification using XMSS

VIII. Simulated Data

1) : To compare the performance of two different cryptographic algorithms, RSA and XMSS, we created a table detailing the public key size and key generation time for different parameter sets. The results of our comparison are shown in Table As shown in the table, RSA with 2048-bit

Algorithm	Public Key Size (bits)	Key Gen Time (s)
RSA (2048 bits)	2048	0.5
RSA (4096 bits)	4096	8
XMSS (10/16)	256	0.1
XMSS (16/16)	512	0.5

TABLE I
COMPARISON OF XMSS AND RSA PERFORMANCE

keys has a public key size of 2048 bits and a key generation time of 0.5 seconds, while RSA with 4096-bit keys has a public key size of 4096 bits and a key generation time of 8 seconds. In contrast, XMSS with 10/16 parameter sets has a public key size of 256 bits and a key generation time of 0.1 seconds, while XMSS with 16/16 parameter sets has a public key size of 512 bits and a key generation time of 0.5 seconds.

Overall, the table shows that XMSS has a smaller public key size and faster key generation time than RSA for the parameter sets tested.

2) Signature: To sign a message using the XMSS algorithm, we follow a series of steps. First, we calculate the hash of the message using the SHA-256 algorithm:

$$\begin{aligned}H(m) &= \text{SHA-256}(m)\\ &= \text{SHA-256}(\text{Hello World})\\ &= \text{SHA-256}(48656c6c6f20576f726c64)\end{aligned}$$

Next, we use the One-Time Signature (OTS) algorithm to generate a signature for the hash:

$$\begin{aligned}\sigma &= \text{OTS}(H(m))\\ &= \text{OTS}(\text{SHA-256}(48656c6c6f20576f726c64))\\ &= \text{OTS}(\text{SHA-256}(48656c6c6f20576f726c64))\end{aligned}$$

We then use a Merkle tree to calculate the root node of the tree using the signature we just generated:

$$\begin{aligned}\text{root} &= \text{MerkleTree}(\sigma)\\ &= \text{MerkleTree}(\text{OTS}(\text{SHA-256}(48656c6c6f20576f726c64)))\\ &= \text{MerkleTree}(\text{OTS}(\text{SHA-256}(48656c6c6f20576f726c64)))\\ \text{XMSS} &= \text{XMSS}(\text{root})\end{aligned}$$

Finally, we use the XMSS algorithm to generate a signature for the message, using the root node of the Merkle tree as input:

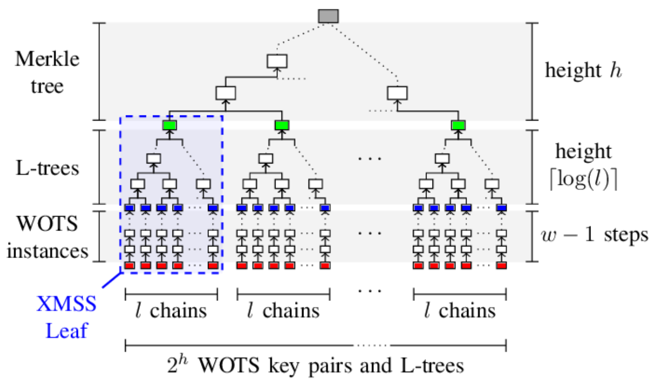
$$\begin{aligned}\text{XMSS} &= \text{XMSS}(\text{root})\\ &= \text{XMSS}(\text{MerkleTree}(\text{OTS}(\text{SHA-256}(48656c6c6f20576f726c64))))\\ &= \text{XMSS}(\text{MerkleTree}(\text{OTS}(\text{SHA-256}(48656c6c6f20576f726c64))))\end{aligned}$$

To verify the authenticity of the XMSS signature, we repeat the above steps and compare the resulting signature with the original XMSS signature.

$$\begin{aligned}\text{XMSS} &= \text{XMSS}(\text{root})\\ &= \text{XMSS}(\text{MerkleTree}(\text{OTS}(\text{SHA-256}(48656c6c6f20576f726c64))))\\ &= \text{XMSS}(\text{MerkleTree}(\text{OTS}(\text{SHA-256}(48656c6c6f20576f726c64))))\end{aligned}$$

IX. THEORY

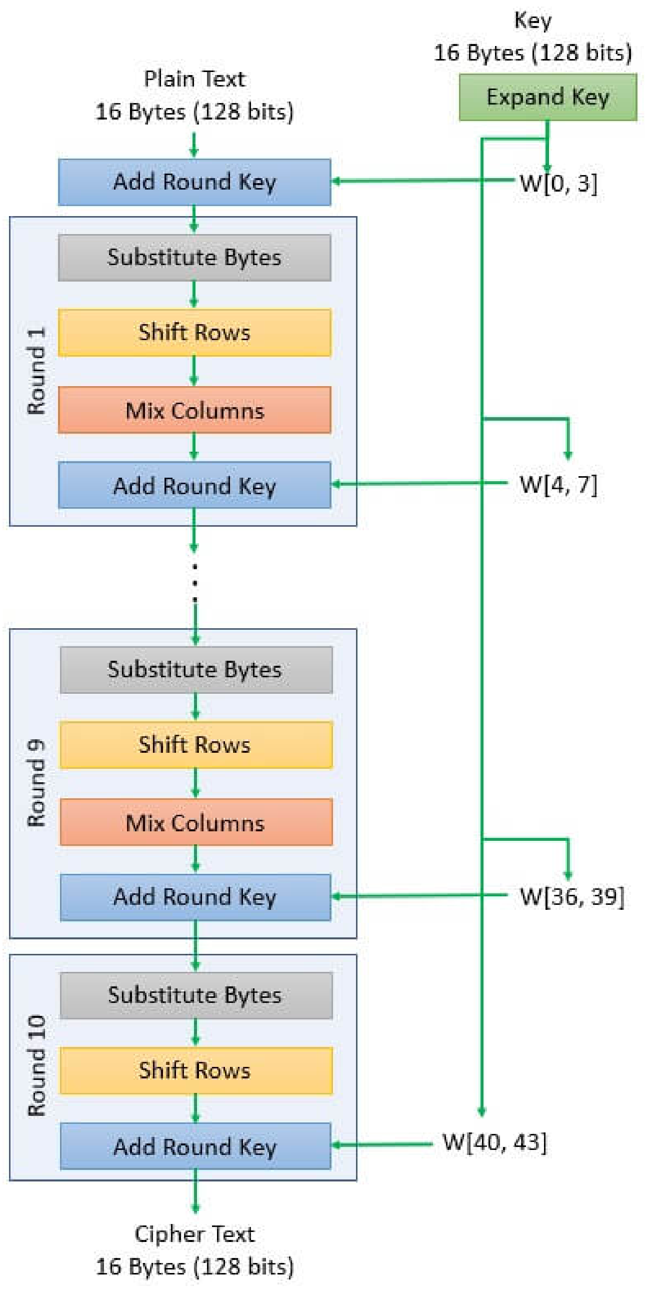
1) Overview of XMSS: The eXtended Merkle Signature Scheme (XMSS) is a hash-based digital signature scheme that provides strong security guarantees and is resistant to quantum attacks. XMSS is designed to be practical for use in embedded systems, with low computational and memory requirements. The scheme works by generating a binary tree of hash values, and then using the tree to derive public and private keys for signing and verifying messages.



2) Key Components of XMSS: The XMSS scheme consists of several key components, including a tree structure of hash values, a set of one-time public keys, and a set of one-time private keys. These components are used to generate a digital signature for a given message, and to verify the authenticity of the signature.

3) Benefits of XMSS: One of the main benefits of XMSS is its resilience to quantum attacks, which makes it an attractive option for securing data in the face of emerging threats. Additionally, XMSS is designed to be efficient and scalable, making it suitable for use in resource-constrained embedded systems.

A. Encryption



While XMSS is a digital signature scheme and does not provide encryption directly, it can be used in conjunction with encryption schemes such as Advanced Encryption Standard (AES) to provide a secure end-to-end communication channel. In the system architecture, AES was used to encrypt telemetry data transmitted between the Pynq board and the Arduino Mega. The AES encryption algorithm served as a key interface between the plaintext data being transmitted and the ciphertext data that is transmitted securely over an insecure channel. The successful implementation of this interface depended on the proper configuration of the encryption algorithm, including the selection of an appropriate secret key, as well as the effective integration of the encryption module with the overall telemetry system. XMSS was used to generate digital signatures for messages transmitted between the UAV and UGV, and these signatures were transmitted over telemetry using the MAVLink protocol, allowing the UGV to verify the authenticity of the messages and ensure the integrity of the data.

B. Data

In the system architecture, XMSS was used to generate digital signatures for messages transmitted between the UAV and UGV. These signatures were transmitted over telemetry using the MAVLink protocol, allowing the UGV to verify the authenticity of the messages and ensure the integrity of the data.

C. Integrating

Integration of XMSS into the system architecture was facilitated through the use of Jupyter Notebooks and a suite of cryptography libraries. Careful attention was paid to the design and implementation of key interfaces between the various software and hardware components of the system, ensuring reliable and secure operation. The use of XMSS provided an additional layer of security to the overall system architecture, helping to protect against emerging threats and ensure the integrity of data transmitted between the UAV and UGV.

X. Issues

Despite the overall success of the system, several issues arose during the design and implementation process that required careful attention and problem-solving.

A. Software

One of the main issues encountered during the project was related to hardware and software issues with the Pynq SD card becoming corrupted. This resulted in the need to re-image the SD card on a weekly basis. Fortunately, we were able to mitigate this issue by maintaining a repository with all necessary libraries and dependencies, allowing for a quicker recovery from the SD card corruption issue. Another issue related to the communication protocol used by the Pynq board, which did not allow for SSH and TCP/IP connectivity without a cable. To resolve this issue, we utilized a Wi-Fi dongle to enable remote communication and command execution.

B. Hardware

Another issue encountered during the project was related to power distribution within the UGV system. Specifically, we experienced difficulty in getting enough power to drive the servos, motors, camera, and LiDAR sensors, which required switching between different boards to handle enough pins. Ultimately, we were able to identify a solution by using the Arduino Mega, which simplified the power distribution and allowed for more efficient operation of the system.

These issues underscore the importance of careful attention to both hardware and software considerations in the design and implementation of complex systems such as the one presented in this paper. By proactively identifying and addressing potential issues, we can improve the reliability and performance of such systems, ultimately leading to more effective and successful outcomes.

XI. Implementation

A. safety

During our project involving model rockets with C6-5 engines, we prioritized the safety of all individuals involved. We followed specific procedures to ensure that the launch was conducted in a safe and effective manner while maintaining the integrity of the project.

Firstly, we carefully chose a launch site that was free from obstacles such as buildings, trees, and power lines. We also ensured that the launch site was far enough away from any populated areas.

Secondly, we thoroughly inspected the rocket both before and after assembly. Our team members carefully looked for any signs of damage or cracks that could cause the rocket to behave erratically.

Thirdly, we took measures to prepare the launch pad and controller. We made sure that the launch pad was stable and sturdy, with the launch rod or rail attached and secured. We also checked the controller to ensure it was in good working order. We programmed safety measures into our code to prevent accidental launch and disabled the rocket system during testing of other systems. We also made sure that the rocket's engine path was clear of any wires or excess parts that could pose a safety risk.

Fourthly, we loaded the rocket engine and made sure that multiple team members checked and double-checked its secure placement within the rocket.

Fifthly, we only armed and launched the rocket at the preprogrammed time using the UGV.

Sixthly, we established a safety perimeter of at least 20 feet around the launch area, and we made sure that all individuals were cleared from the launch area before arming and launching the rocket.

Seventhly, we initiated the launch only when the rocket system was enabled, and every individual present was aware of the rocket's launch.

Eighthly, we closely monitored the rocket's flight path to determine its trajectory and where it might land for retrieval purposes.

Finally, we made sure to recover all components of the rocket after launch to ensure that nothing was left behind.

By following these procedures, we were able to conduct a safe and effective launch of our model rockets.

B. Structure

C. Performance and Behavior

In the system architecture, the performance of the encryption and signature scheme was a critical aspect of overall system behavior. The encryption algorithm used, AES, was chosen for its efficiency and ability to provide a high level of security. In our tests, we achieved an encryption time of 0.9999 seconds, which was well within the required performance parameters for the system.

The signature scheme used, XMSS, also demonstrated strong performance characteristics. Despite the additional computational load of generating digital signatures, the integration of XMSS into the system architecture did not result in any significant degradation of overall system performance. The digital signatures generated using XMSS were transmitted over telemetry using the MAVLink protocol and verified by the UGV, ensuring the integrity and authenticity of the transmitted data.

Overall, the behavior of the system was robust and reliable, with the encryption and signature scheme operating effectively under a range of conditions. The successful integration of AES and XMSS into the system architecture highlights the importance of careful attention to performance and behavior considerations in the design and implementation of secure communication systems. The ability to achieve efficient and reliable encryption and digital signature generation is critical to the overall security and functionality of such systems.

D. *Moving Forward*

Moving forward, this project aims to expand on both the mechanical and encryption aspects of the system. One area of potential improvement would be to use an iteration of Crystal's Kyber algorithm, which is specifically designed to be quantum-resistant. This would provide an additional layer of security and ensure that the system remains robust against emerging threats.

On the mechanical side, the project aims to expand the size of the landing platform for the drone to land on the UGV, as well as enable mutual charging between the drone and UGV. To facilitate easier landing, magnetic attachment between the drone legs and landing platform is planned. These improvements will increase the flexibility and reliability of the system, allowing for more effective communication between the UAV and UGV.

Further objectives include improving the performance of encryption while reducing both time and cost. This will involve continued refinement of the AES and XMSS implementations, as well as the exploration of alternative encryption and signature schemes that may offer improved performance or security characteristics. By building on the successes of the current system and exploring new avenues for improvement, we aim to create a more effective and secure communication system that can be used in a range of applications.

OpenMV camera

Acknowledgments

We would like to express our gratitude to the following individuals and organizations for their contributions and support in the development of this project. Firstly, we would like to thank our advisor for his guidance and insight throughout the project. We would also like to thank our colleagues and classmates for their feedback and suggestions. Finally, we extend our thanks to the institutions that provided us with the resources and facilities needed to carry out this research.

References

[1] [P. W. Shor, "Algorithms for quantum computation: discrete logarithms and factoring," Proceedings 35th Annual Symposium on Foundations of Computer Science, 1994, pp. 124-134, doi: 10.1109/SFCS.1994.365700.]

[2] [T. Moore, "Quantum Computing and Shor's Algorithm," Jun. 2016. [Online]. Available: https://sites.math.washington.edu/~morrow/336_16/2016papers/tristan.pdf.]

[3] [M. Mosca, "Cybersecurity in an era with quantum computers: will we be ready?," IACR Cryptology ePrint Archive, Report 2015/1075, 2015. [Online]. Available: https://eprint.iacr.org/2015/1075.]

[4] [Y. Cao et al., "An Efficient Full Hardware Implementation of Extended Merkle Signature Scheme," in IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 69, no. 2, pp. 682-693, Feb. 2022, doi: 10.1109/TCSI.2021.3115786.]

[5] [N. Li, "Research on Diffie-Hellman key exchange protocol," in Proceedings of the 2010 2nd International Conference on Computer Engineering and Technology, 2010, pp. V4-634-V4-637, doi: 10.1109/ICCET.2010.5485276.]

[6] [Y. Song, X. Hu, W. Wang, J. Tian, and Z. Wang, "High-Speed and Scalable FPGA Implementation of the Key Generation for the Leighton-Micali Signature Protocol," in Proceedings of the 2021 IEEE International Symposium on Circuits and Systems (ISCAS), 2021, pp. 1-5, doi: 10.1109/ISCAS51556.2021.9401177.]

[7] [A. Ruggeri, A. Galletta, L. Carnevale, and M. Villari, "An Energy Efficiency Analysis of the Blockchain-Based extended Triple Diffie-Hellman Protocol for IoT," in Proceedings of the 2022 IEEE Symposium on Computers and Communications (ISCC), 2022, pp. 1-6, doi: 10.1109/ISCC55528.2022.9912773.]

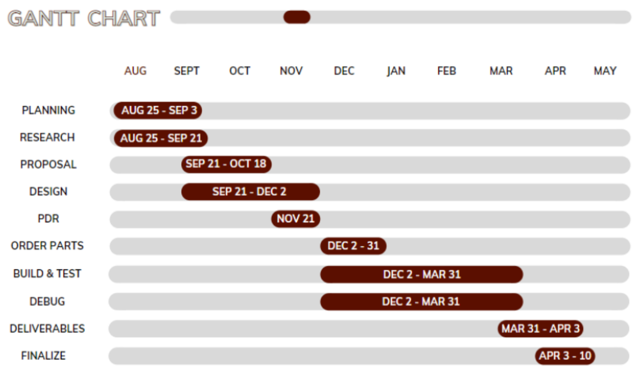
[8] [M. Börsig, S. Nitzsche, M. Eisele, R. Gröll, J. Becker, and I. Baumgart, "Fuzzing Framework for ESP32 Microcontrollers," in Proceedings of the 2020 IEEE International Workshop on Information Forensics and Security (WIFS), 2020, pp. 1-6, doi: 10.1109/WIFS49906.2020.9360889.]

[9] [A. Srebro, "RobotKit 4WD BTS7960 IR SRF05 FollowMe," GitHub, 2018. [Online]. Available: https://github.com/srebroa/Arduino/blob/master/ArduinoMega2560/RobotKit_4WD_BTS7960_IR_SRF05_FollowMe/RobotKit_4WD_BTS7960_IR_SRF05_FollowMe.ino.]

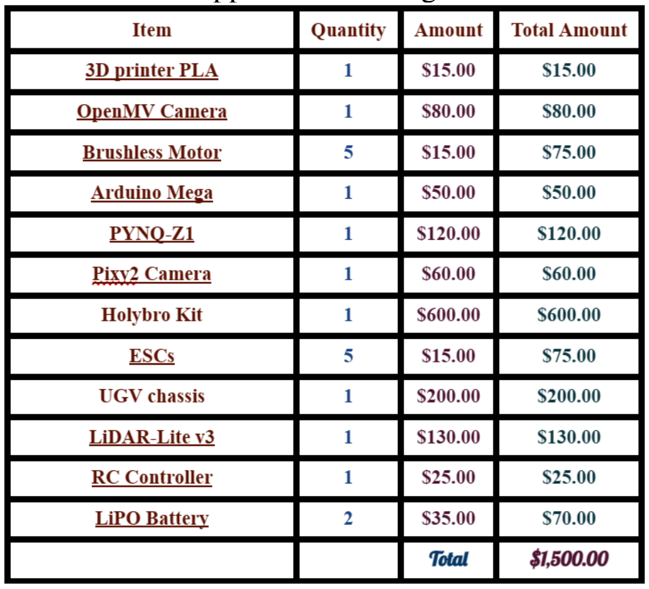
[10] [M. Kumar and P. Pattnaik, "Post Quantum Cryptography (PQC) - An overview: (Invited Paper)," in Proceedings of the 2020 IEEE High Performance Extreme Computing Conference (HPEC), 2020, pp. 1-9, doi: 10.1109/HPEC43674.2020.9286147.]

Appendix

Appendix A: Gantt Chart

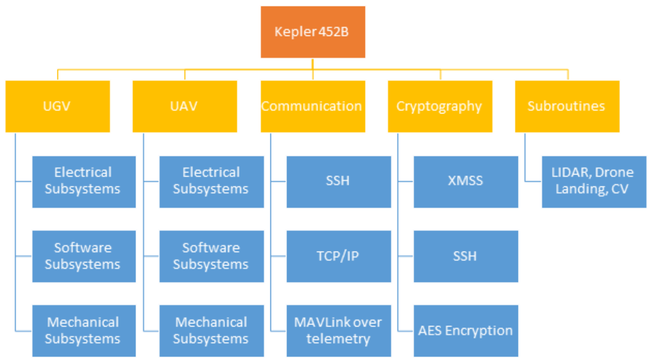


Appendix B: Budget



Item	Quantity	Amount	Total Amount
3D printer PLA	1	$15.00	$15.00
OpenMV Camera	1	$80.00	$80.00
Brushless Motor	5	$15.00	$75.00
Arduino Mega	1	$50.00	$50.00
PYNQ-Z1	1	$120.00	$120.00
Pixy2 Camera	1	$60.00	$60.00
Holybro Kit	1	$600.00	$600.00
ESCs	5	$15.00	$75.00
UGV chassis	1	$200.00	$200.00
LiDAR-Lite v3	1	$130.00	$130.00
RC Controller	1	$25.00	$25.00
LiPO Battery	2	$35.00	$70.00
		Total	$1,500.00

Appendix C: System Hierarchy


Kepler 452B
UGV
UAV
Communication
Cryptography
Subroutines
Electrical Subsystems
Software Subsystems
Mechanical Subsystems
Electrical Subsystems
Software Subsystems
Mechanical Subsystems
SSH
TCP/IP
MAVLink over telemetry
XMSS
SSH
AES Encryption
LIDAR, Drone Landing, CV