Design of a Low-Cost CubeSat for Earth Monitoring and Data Transmission

Robert Hudson,[1] Christian Smith,[2]
Deanna Des Rivieres,[3] Tucker Barnes,[4] and Wout De Backer[5]
University of South Carolina, College of Engineering & Computing, Columbia, SC, 29201

CubeSats have long been used as an educational tool for students to learn the basics of space engineering design. With each CubeSat being designed for a specific mission, the cost and complexity of the design can vary greatly. This paper will detail the design of a CubeSat that can take a 480x480 pixel image of Earth from orbit every minute over a duration of 24 hours. The conceptual design phase of the project includes determining the initial ideas and configurations, including the process of analyzing the risks, successes, and failures of each design concept. Complexity of the design and data transmission rates were the critical constraints of the design that were not the initial mission requirements. These constraints, in addition to others documented, lead to the decision to use a Raspberry Pi logic board to control the CubeSat. The processing power cost, transmission speed, and overall simplicity allowed it to outperform other options explored when analyzed using a tradeoff table. The Raspberry Pi logic board interfaces with a camera capable of taking the required images, a power system containing enough battery power to last the required duration, the attitude determination and control system, and the communications system. The communications architecture will use the long-range capabilities of LoRa to transmit the images to the ground station terminal. The CubeSat will also analyze its own status by measuring data transfer rate, charge capacity, and data storage. This data will be sent to the ground station as well. The attitude determination and control system uses an active system that can control two-axes of rotation to maintain the camera's position toward Earth. Using two-axes control instead of three-axes control will reduce the complexity of the design and calculations. When in orbit, the CubeSat will rotate around two axes to keep Earth in frame. Given the requirements, the orientation of Earth within the camera frame is not critical. The orientational control system is not able to be tested with on-Earth conditions and is validated mathematically within this paper before arriving to orbit. Other systems, such as the communications architecture and the power control system, are validated mathematically and experimentally.

Nomenclature

CPU	= central processing unit
SBC	= single board computer
HAT	= hardware attached on top
LiPo	= lithium polymer
LoRa	= physical proprietary radio communication technique
I2C	= inter-integrated circuit
SDC	= serial data
SCL	= serial clock

[1] Undergraduate Student, USC Department of Aerospace Engineering
[2] Undergraduate Student, USC Department of Aerospace Engineering
[3] Undergraduate Student, USC Department of Aerospace Engineering
[4] Undergraduate Student, USC Department of Aerospace Engineering
[5] Assistant Professor & Faculty Advisor, USC Department of Mechanical Engineering, Senior Member

I. Introduction

Given the task of designing a CubeSat for theoretical deployment with a budget of $1250 and two months proved to be a difficult task. The following paper is a summary of the thought process and design options that were considered, chosen, and then produced and presented by the authors of this paper.

A CubeSat is a class of nanosatellite that utilizes a standard size and form factor. The standard size is 1U or 'one unit' measuring 10cm x 10cm x 10cm and scales as such, typically up-to 12U. The CubeSat platform and formfactor has been gaining popularity among government, industry, and especially academia. CubeSats fill a niche providing a lower cost option to engage in scientific research and observations outside of Earth's atmosphere. [1]

II. Design Considerations

The design of the CubeSat began with evaluating the mission statement, given as "Take an image of Earth from orbit every minute for 24 hours and send the data and the system's status back to Earth to be processed by a ground station". Each task was identified on its own and then research began on how each task can be completed and the challenges that might be encountered. This process can be shown below in Figure 1.

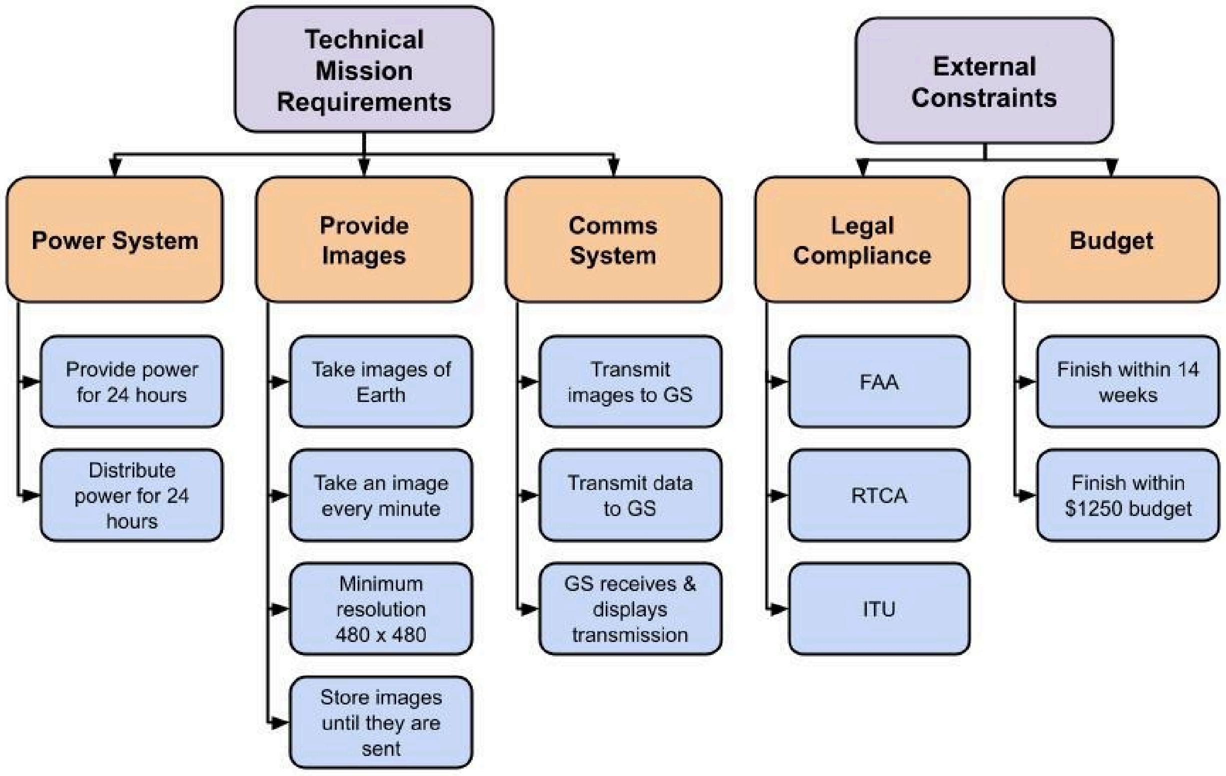
Technical Mission Requirements
External Constraints
Power System
Provide Images
Comms System
Legal Compliance
Budget
Provide power for 24 hours
Distribute power for 24 hours
Take images of Earth
Take an image every minute
Minimum resolution 480 x 480
Store images until they are sent
Transmit images to GS
Transmit data to GS
GS receives & displays transmission
FAA
RTCA
ITU
Finish within 14 weeks
Finish within $1250 budget

Fig. 1 Requirements Discovery Tree

A. CPU Selection

The first design decision that was made was selection of a CPU. With the consideration of budget and time constraints imposed by the project it was opted to select an off-the-shelf CPU. This was then broken down into microcontrollers or an SBC. It was from here that an SBC would work better than most commercially available microcontrollers as it would enable continuous control of all systems. The main failure of a microcontroller for the CubeSat operation is the distinct lack of multitasking capabilities. If the mission was run on a microcontroller unit, it is expected that the backlog of queued tasks would ultimately cause the CPU to overheat or crash altogether. The SBC option was explored in a more in-depth manner. Store-bought SBCs included Raspberry Pi, Rock 3C, Odyssey x86, Vision-Five 2, and Orange Pi. The choice of the SBC was based on the availability of documentation for the processor. Given its popularity, the Raspberry Pi SBC was chosen as it had the most amount of documentation for the applications of this mission [2][3][4]. Raspberry Pi SBCs can be broken down further into multiple processors of various sizes and strengths.

Given the lack of size and weight requirements for this mission, a stronger but larger processor could be chosen.

This resulted in the choice of using the Raspberry Pi 4 CPU. The Raspberry Pi 5 was considered for its increased processing capabilities but the documentation surrounding the operating system was considerably less than what was available for the Raspberry Pi 4.

Once the CPU was chosen it was possible to begin looking for compatible components to complete the remaining mission tasks. The design decision was made to use the standard HAT interface for Raspberry Pi devices. The HAT interfaces directly with all 40 pins on the CPU and allows for passthrough for the addition of multiple HATs per device.

B. Imaging System

With the main purpose of the CubeSat being to visually monitor Earth, the imaging system was a critical part of the design process. After searching through many options for cameras, it was decided to simply use the RaspberryPiCam3, which was designed specially for use on the Raspberry Pi [5]. It was also decided to select a model with a 120-degrees wide angle lens. This selection allows for the image to cover 1,920,000 km^2 of land with the determined orbit (discussed further in section I). While the default resolution of the camera is 12 megapixels, the desired image quality is 480 x 480p which is 230 kilopixels. This will help reduce image file size making it possible to transmit the images in a time efficient manner. To further help with this, the images will be reduced in quality when saved to the Pi4's internal storage. The file size of the images captured ranges from 5-10 kb, being well within the budgeted size to allow for complete transmission before the next image is captured. This extra time allotment allows for implementation of forward error correction to help ensure that the images are received as intended.

C. Control System Design

A key part of the mission is to be able to consistently take photos of Earth during the 24 hour mission. To keep Earth in frame, a control system is required to rotate the satellite's view. With the current selected orbit, the satellite will have to rotate a full 360 degrees every 1.542 hours.

Initially it was thought that a full 3-axis control system would be required to be able to take clear images of the Earth. This idea evolved as it was realized that some assumptions could be made. There was no design constraint for image orientation, so automatically the axis that aligns through the camera lens does not need controllability. It was then further assumed that the satellite would be deployed into orbit with no rotational forces acting on it. If this is assumed, then control of the pitch axis (toward and away from the sun) is not needed. Thus, the only axis that needs controllability is the one that must be rotated about to keep Earth in frame.

To do this, the satellite has one reaction wheel installed inside the shell. The reaction wheel has a 93 gram disk attached to a DC motor. The motor is controlled by the Adafruit DC & stepper motor HAT which allows for bi directional control with 8-bit speed resolution. This motor controller was chosen over the other smaller and cheaper option because it allows for manual readdressing of the I2C address that it uses to communicate with the RaspberryPi making it easier to set up and program. These motors, when in a zero-gravity environment, turn on to produce torque with the weighted disk. Since there are no external forces holding the satellite in place, the equal and opposite reaction slowly rotates the satellite in the opposite direction.

The control system uses the Waveshare Sense HAT for Raspberry Pi. It uses onboard gyroscope chips to measure angular displacement during the mission. The sensor is calibrated at launch to track the angle of its orientation throughout the mission. With the satellite rotating at approximately 0.0011318 rad/s relative to Earth, the control system and motors will have to counter this rotation. Assuming that the satellite is to be deployed above the north pole and north is positive, it can be assumed that the desired orientation can be predicted with equation 1 shown below. It takes orbit time (in seconds) and yields the required angle that the gyroscope must align to relative to the start angle. [6]

$$Theta = 360 * \sin(\ 0.5 * (1/5553.5) * \mathrm{t})$$

D. Power System

One of the larger design constraints is that the system must function for a minimum duration of 24 hours. To achieve this goal, it was decided to use a large LiPo battery bank in conjunction with solar panels. Battery selection was rather simple as off-the-shelf parts were bought due to time constraints. Sourcing a high capacity rechargeable battery that would be able to fit within the 1.5 U design (discussed further in section H) did not prove to be difficult as there were limited options. When selecting the battery, it was narrowed down to three different sizes and capacity batteries. It was chosen to go with the Samsung 50S 21700 5000 mAh 25A Battery as it yielded the highest capacity for the lowest percent volume required for the battery bank.

The next requirement was to select a battery charger. Initially the HiLetgo TP4056 LiPo charger had been selected. This would have been more than capable of supporting the estimated current produced by the solar panels and adequately charging the batteries. Since the Raspberry Pi requires a constant operating voltage of 5 V the power depletion of a battery overtime had to be taken into consideration. It was during this process that the AdaFruit PowerBoost 1000 Basic - 5 V USB Boost @ 1000 mA from 1.8 V was found. This charge controller/boost converter combination allowed reduction of the original design while also operating at a 90% charge/discharge efficiency. The controller also has built-in load sharing which means that it can charge the batteries and power the Raspberry Pi at the same time increasing effective efficiency. This charger also works with voltages as low as 1.8 V which will work well in case the installed solar panels are ever obstructed during the mission.

The solar panels that were used in the design are 1.2-Watt 6 V solar panels from Voltaic Systems. According to the panel data sheet, the expected power output is estimated to be 1.09 W at 5.89 V. With the design containing 3 panels, the expected power input can be around 3.27 W. Accounting for the 90% efficiency of the charger, the expected 2.944 W of power is accessible to the Raspberry Pi.

The batteries have a full capacity of 20 Ah and at a nominal voltage of 4.2 V, they have a power capacity of 84 Wh. With the solar panels producing 2.9 W, or 69.6 Wh over the 24 hour mission duration it can be further assumed that the total power available during the mission is 153.6 Wh. The loaded current draw of the selected motors is 0.16 A, with a nominal voltage of 3 V, the motors peak power usage is 0.48W, assuming it operates continuously (which will not be required), it will consume 11.52 Wh over the course of the mission. The idle power drawn from the RaspberryPi4 is 2.85 W, while the tested under-load power consumption is estimated to be around 5 W. It was assumed that the Pi will spend 50% of the mission duration under load; the total power consumption of the Pi during the mission comes out to 94.2 Wh. The transmit power consumption of the LoRa hat is 0.5 W, assuming that 50% of the mission time is transmitting data, the power consumption is 6.6 Wh. The Waveshare Sense HAT that is being used as the gyroscope has a current draw of 3.11 mA only using 0.26 Wh throughout the duration of the mission. This brings the total estimated power consumption, while overestimating power draw, to 112.58 Wh. This means the current power system can supply 36% more power over the duration of the mission than needed.

E. Communications System Design

The communications system is tasked with sending and receiving the data prepared by the processor. LoRa radio communication was chosen with the range, bit rate, and complexity being considered [7][8][9]. LoRa communication is based on spread spectrum modulation techniques derived from chirp spread spectrum technology. For US communications, LoRa uses the license-free sub-gigahertz radio frequency bands US915 which operates between 902 -928 MHz. The bit rate and range of communications vary between LoRa transceiver and receiver products. For this design, a Waveshare SX1262 LoRa transceiver HAT is used. The transceiver has a radio rate range of 0.3 kbps – 62.5 kbps depending on range and package characteristics. The transmit power is 22 dBm with power consumption of 100 mA during transmission. The maximum range of transmission for this transceiver is 5 km. However, given the large nature of the data being transmitted, the maximum range allowed for transmission of the images within a reasonable time is ~500 meters.

The method of sending the data is vital to the proper operation of the CubeSat during its mission. The image taken by the Imaging subsystem must be processed into a format that the SX1262 LoRa transceiver can send and receive without high bit rate error or total failure. This constraint requires the CPU subsystem to process the image into hexadecimal code so that the image can be sent in packets of data. The maximum packet length that can be sent by the transceiver is 240 bytes. Of these 240 bytes, the leading 16 bytes and tailing 8 bytes will be used as an address for forward error correction techniques. With large packets there is substantial bit rate error that must be addressed. This error can be catastrophic to the mission by ruining any image that is transmitted. Further information regarding this error and its mitigation is presented in section G.

F. Ground Station

The design of the ground station was rather brief. It has one job and that is to receive the file packages from the CubeSat and process them yielding images or system data. With the LoRa HAT that was selected to be used as the transceiver, the simplest way to receive the data is with another Waveshare SX1262 915Mhz LoRa HAT. They are designed to work very well with peer-to-peer communication. The initial design called for using the UART to USB built into the HAT, then plugging the HAT directly into a laptop to use as the ground station. However, the software that comes with the LoRa HAT for interpreting the data proved difficult to use. To fix this issue in a timely manner, with the budget more fleshed out, it was decided to order and use a second Raspberry Pi as the ground station processing unit. This makes it easier to receive the data from the LoRa HAT directly to the I2C pins on the Raspberry Pi and run them through the python code to process the information. Using a Raspberry Pi at the receiving end also simplified the coding process as the team was more familiar with the environment and setup processes. With the Raspberry Pi also functioning as a standalone computer it makes it simple to hookup peripherals and a monitor to be able to read and display the received information as well as plenty of on-board space for storage.

G. Risk Assessment

The risks associated with this design are listed below:

1. Connection Failure
2. Continuity Failure
3. Short Circuit Failure
4. Single Event Upsets
5. Bit Rate Error
6. Solar Panel Structural Error
7. Solar Panel Operational Failure
8. Complexity Issues
9. Battery Failure
10. Structural Damage
11. Inclement Weather
12. Ground Station Failure
13. Active Control Failure
14. Software Failure

Connection failure accounts for the risk of faulty connections between pins because of an error during the setup of the CubeSat or pins loosening during/after launch. This error could result in loss of data transmission, data acquisition, and, in severe cases, total loss of control of the CubeSat. This risk will be mitigated by using soldering as the main wire connection technique during construction for all loose wiring.

Continuity failure accounts for the risk of damaged wires that limit the transmission of data between different pieces of hardware. Like connection failure, this error could result in loss of data transmission, data acquisition, and, in severe cases, total loss of control of the CubeSat. This risk will be mitigated by routine wire health checkups during the construction process leading up to the mission operation.

Short circuit failure accounts for the risk of the CubeSat system being overwhelmed and short circuiting. This can easily be a catastrophic failure for the mission if no proper considerations are given to preventing the system from being overwhelmed.

Single event upsets are extremely unlikely events in which a stray radioactive particle like a neutron collides with the system and flips a single bit to cause total system failure. Given the unlikelihood of this risk, no considerations were made to mitigating this risk.

Bit rate error accounts for the risk of the wrong bit data being sent from the CubeSat to the ground station. In some cases, bit rate error can be substantial to the point of not sending entire packets of data. This error can completely prevent the ground station from properly decoding the sent data into understandable images. This risk can be mitigated through forward error correction techniques. In this design, the forward error correction technique implemented will be using header and footer hexadecimal addresses to properly determine the start and end of the packages while ensuring that the contents are safe. Additionally, information such as the overall length of the packages can be sent within the header address that the ground station can check to ensure that the packages received are correct. If the packages are determined to have too much bit rate error, the ground station can request the CubeSat to either resend the image or the ground station can simply ignore the data and wait for the next transmission.

Solar panel structural failure accounts for the risk that the solar panel could take any kind of structural damage which could result in partial or total loss of power generation. This can be a result of space debris, issues during launch, or mishandling during construction. Mitigation of this risk involves careful storage and handling of the solar panels. Additionally, a structural analysis of the solar panels can be conducted to determine the acceptable loads that the solar panels can take. However, given that the use of solar panels is to act as a backup to increase the safety factor of the CubeSat power system, the failure of the solar panel system is not considered catastrophic.

Solar panel operational failure accounts for the solar panel failing to operate properly resulting in partial or total loss of energy generation. This failure can be caused by structural damage/failure, connectivity issues, and continuity issues. The mitigations of those risks are detailed prior.

Complexity issues account for the risk that the overall system is too complex to design within the time frame required. This could result in the requirement of an extended deadline, or the mission being canceled entirely. Mitigation of this risk was completed during the Conceptual Design phase by analyzing the complexity of each processor and subsystem to determine the simplest design without losing processing power.

Battery failure accounts for the batteries either running out of power or failing to provide power to the system. This would be a catastrophic failure for the mission given the reliance on battery power to power the CubeSat. Multiple health checks will be conducted periodically during construction and finalization before the mission starts to ensure the batteries have no obvious issues.

Structural damage accounts for the risk of damage to the chassis of the CubeSat which could result in small deformations or catastrophic deformations that damage the internal parts. This could be caused by mishandling of the CubeSat prior to operation or complications during operation such as stray space debris. Like the solar panel structural risk, mitigation involves the careful storage, handling, and creation of the structure of the CubeSat. Additionally, a structural analysis of the CubeSat was conducted to measure the load limitations of the design.

Inclement weather can decrease the bit rate transmission speed as well as the communication distance. This risk can lead to major issues during operation resulting in loss of data and loss of control of the satellite. It is difficult to mitigate this risk due to the uncertainty of weather. However, safety factors on the distance can be adjusted by setting the operating distance of the CubeSat as closer than the maximum limit of connectivity.

Ground station failure accounts for any complication that occurs with the receiver and ground station that can result in total loss of data acquisition from the CubeSat transmitter. Ground station failure can also involve improper coding of the post-processing functions that leads to runtime errors that may require on-the-fly debugging and adjustments. However, improper coding of the ground station is unlikely due to the simplicity of the post-processing code and the substantial allocation of debugging time during construction.

Active control failure accounts for the possibility of losing control of the CubeSat's orientation resulting in images not of the intended target. This can be seen as a major, if not catastrophic failure as the CubeSat will not be able to complete its intended mission. Mitigation for this risk involves intensive testing of the Active Control subsystem before mission operation to ensure that all software and hardware components are operating as intended.

Software failure acts as a "catch all" risk assessment for CubeSat's software health. It accounts for the possibility of failure of the developed code across all subsystems such as the CPU, ground station, active control, and imaging. Mitigation of software failure risks requires substantial debugging time to be allotted during the construction and finalization phase of the project.

H. Structural Design

The main structural elements used in the design were provided by a previous design team. This was utilized as a cost saving measure. The overall dimensions of the CubeSat were determined to be 10cm x 10cm x 15cm to be able to easily hold all the required components. This size CubeSat is known as 1.5 U. The bottom face of the CubeSat (one of the 10 x 10 faces) has a camera and three of the side faces have solar panels. The structure of the CubeSat consists of four 10mm x 10mm Maker Beams in the corners which are threaded through the middle to make fasteners easy to install. The CubeSat is also covered in a shell of sheet metal 0.5 mm thick. All these structural elements are made of aluminum. To properly model the CubeSat structure for analysis, the shell needed to have holes where fasteners are anticipated to be placed and holes for the camera lens or wires leaving the solar panels will thread through.

Since the CubeSat's primary production life will be in orbit around Earth, the highest loading the CubeSat will face is during launch. The G-forces the CubeSat will face will reach up to 3 times Earth's gravity, however a sustained force of 10 Gs is applied to consider less likely events like impacts [10]. These impacts were modeled in Abaqus to ensure sound structural design. With the small and light nature of CubeSats minimal deformations and overstress was found with a maximum deformation of the structural plates being 0.07 mm.

I. Orbit and Aerodynamics

One of the fundamental decisions in the CubeSat design was the choice of orbit. An altitude selection of 400 Km was decided on as it was a good balance of reduced angular velocity, which is important to reduce control system power consumption, and the practical cost of launching the satellite to a high altitude. While the launch cost was considered for the purposes of this paper as it is an application of theory while producing a physical object.

The location of the orbit was specified too. The CubeSat will be in a polar orbit meaning its orbit passes over or near both poles of the planet. Also, the plane of the orbit will be perpendicular to the direction of incoming light from the sun. This makes sure the CubeSat is exposed to solar photons allowing the solar panels to generate electricity for the entire duration of the mission. As Earth orbits the sun, the plane does not rotate meaning it will not continue to permanently be exposed to the sun, but this is not relevant in the time scale of the CubeSat's mission. The CubeSat will also be in a circular orbit, meaning it's a perfect circle and the distance from Earth never changes. The orbital velocity was calculated to be 7668.5 m/s with an orbit time of 1.5426 hours. The angular velocity of the CubeSat relative to Earth is found to be 0.00113 radians per second. This is the same rotational speed that the control system will have to counter to keep Earth in frame of the satellite's camera. Calculations for drag on the satellite were analyzed, but proved to be miniscule enough to be neglected.

III. Bill of Materials

After completing the project planning, preliminary design, and detailed design, the key required components have been identified. These components have been organized into a bill of materials and sorted by each system to help keep track of the budget and materials ordered. It contains detailed product names, subsystems that the components are needed for, price, number needed, and product shop location. The bill of materials also contains estimated shipping cost and taxes.

Since the technical advisor of this paper has overseen a previous iteration of the project, there were a lot of materials that were donated and then recycled into the project. The main example of this is the spare structural beams and hardware that goes with it. These parts helped keep the cost of the project low while imposing minimal design restraints as they are rather versatile. Some of the old hardware was also used for system testing while waiting for new components to arrive. Some parts used for system testing were a motor controller and the charge controller/boost converter.

J. Budget Breakdown

Below is the current budget breakdown with all parts that were used for assembly and testing. As previously stated, a lot of the parts for this project were sourced from in-house, from the students, or from previous design teams. The most expensive systems included the ground station and CPU as they each required their own Raspberry Pi 4 as well as Waveshare LoRa HAT. With the recycling of old structure materials, the cost to build the structure was kept low compared to previous teams. The total cost came out to $411.47, a mere fraction of the allotted budget. See Table 1 for the detailed budget.

Subsystem	Price (USD)
Camera	36.95
Structural	19.99
Control	71.44
CPU	90.45
Power	14.95
Ground Station	99.23
Testing	19.95
Misc	26.99
Total	379.95
Total (including tax and shipping)	411.446

Table 1 Budget Breakdown

K. Weight Budget

While the weight of the satellite is important for some cases, such as launch constraints, the design has no maximum weight imposed by the mission statement. This does not mean that it was designed without weight in mind, weight was just not a constraint of the design. The weight impacts the effectiveness of the control system discussed in section C. Since a higher orbit was chosen, the orbit time was increased thus decreasing the total amount of control needed. This is the case because the satellite must rotate a full 360 degrees for each revolution around Earth.

IV. Interfacing and Production

A large portion of the time taken once detailed design was completed was spent interfacing all the electronic components as well as writing code to instruct the various systems on what task they must do. This section of the paper will highlight some of these interfaces as well as discuss the internal layout of the satellite.

A. CPU Interface

The main interface connection that is used between the various microprocessors and the CPU is the I2C protocol. It is a bus interface connection protocol that is used to allow two devices to communicate with each other using serial data. All the various HATs used in the design communicate through this protocol to send and receive data to the Raspberry Pi. The Raspberry Pi only has one SDA and SCL pin. These are the only pins on the Pi that it uses for I2C communication. Since there are multiple devices feeding data through the same pins the protocol allows for addressing from each device to keep the data separate. This was a big factor in electronic selection as the design had to make sure that each device, if they had fixed addresses, was different from one another. This also made it easier to code each device, because despite them all running through the same set of data pins, could essentially be programmed as if they were entirely separate.

B. Code

The entire mission was once again broken down into a task and then the systems and components that were required to complete that task.

The first one was image capturing. The components identified were the camera, CPU, and transceiver. The Raspberry Pi 4 has a built-in port for a camera connection that allows for easy interfacing. There are code libraries online that are designed to work with PiCamera3 making the code to take a picture as simple as ‘picam.capture_file(“image.jpg”)’. The more challenging part was writing the code to convert the image into hex code so it can be saved and sent to the transceiver to broadcast. However, since the transceiver works through the I2C protocol, the transmission process involved sending the data packets to the SDA and SCL pins with the I2C address of the LoRa module.

The next system is the controls system. This one has more feedback loops, it must acquire data and then complete the task while checking the acquired data. This is done with the WaveShare Sense Hat which has a 3-axis gyroscope that outputs the orientation of the satellite relative to the initialization point. Since there is no rotational velocity acting on the satellite after deployment, the angles read from the gyroscope should remain constant. However, the point of the system is to rotate the satellite around its z-axis during its orbit to keep Earth in frame. In section C the equation for the desired orientation of the satellite was found to be a function of mission time in seconds. Using this equation the python code takes the output angle for the z-axis of the gyroscope and the desired angle and compares them. If the desired angle is less than the actual angle the code will send a signal through the I2C protocol addressed to the motor controller turning on the motor with a predetermined power in the code [11][12][13]. Once the motor starts moving the satellite it will keep going until it overshoots the desired angle. This is done to minimize the total number of times the satellite must make corrections to reduce computational power.

The final system that required coding was the ground station. This system was the simplest to code as the code for receiving data through the WaveShare LoRa HAT runs continuously and can print all received transmissions to the terminal or print them to a .txt file. From here a second simple python code was written to take the .txt file and decode it yielding the .jpg image that was taken from space.

C. Physical Interface

Designing the assembly in CAD proved to have some challenges. It was decided to neglect wiring from the 3D models to reduce complexity and keep the models easy to parse. Below in Figure 2 is the assembly. The battery bank and charge controller are at the top of the satellite to keep its center of gravity in the center. The camera module sits with its lens flush through the bottom of the structure with the Raspberry Pi just a few millimeters above it. Then comes the HAT stacks with the motor controller, WaveShare Sense, and the WaveShare LoRa modules. On the right side of the internal structure is the motor and flywheel that rotate the satellite about its y-axis. It was placed about the y-axis moment of inertia to maximize its effectiveness. The solar panels can also be seen in the 3D model.

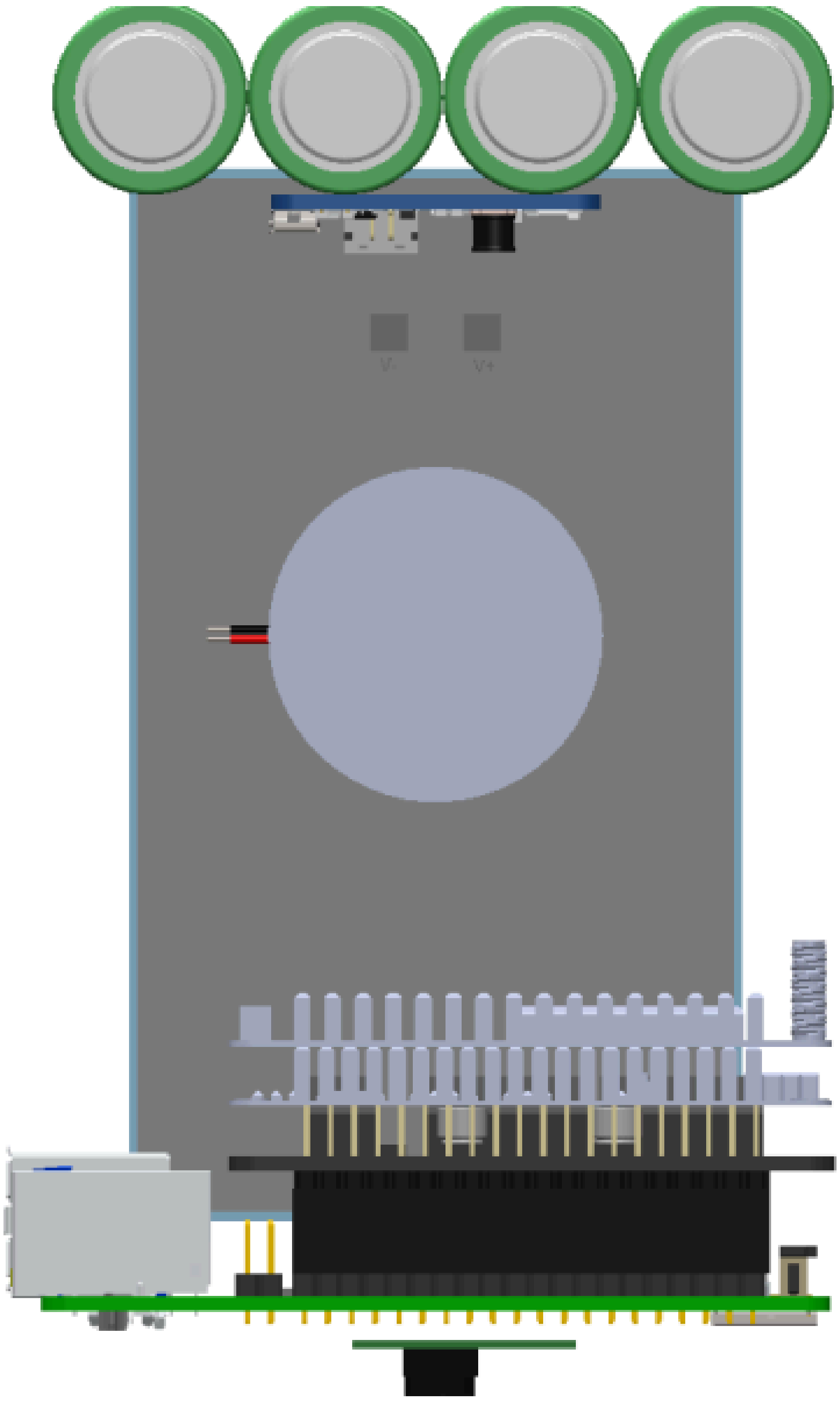

Fig. 2 Internal Electronics

D. Structure

The physical structure was built around the design of the electronics, so it is rather simple as previously discussed. Below in Figure 3 a 3D model of the structure can be seen with both a view of the pure structure and one where the structure is translucent to show how the electronics fit inside. All holes in the outside panels are for fasteners or for solar power wires to pass through into the structure.

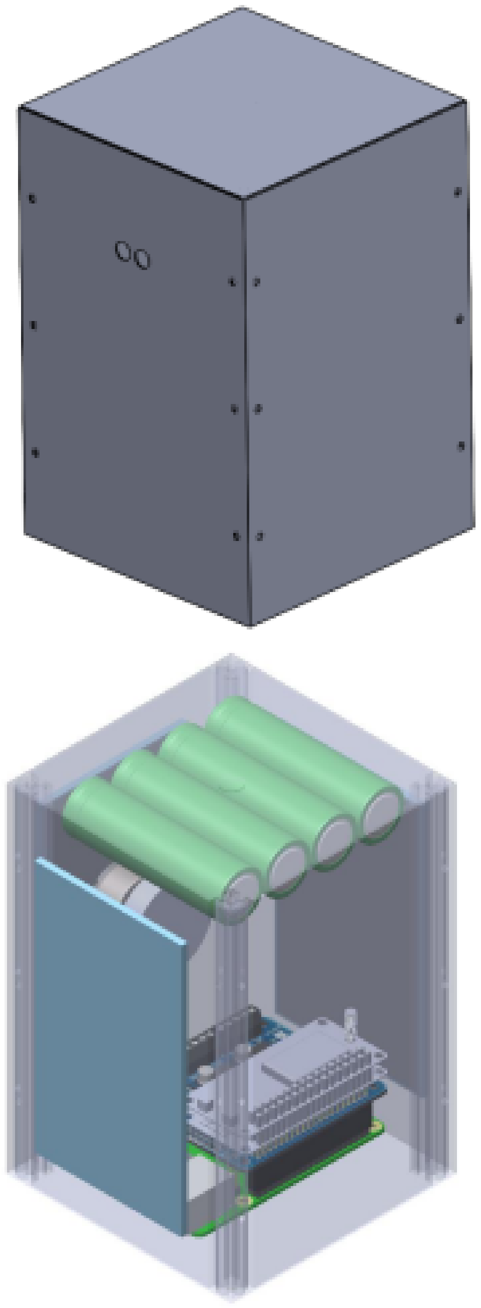

Fig. 3 Structure View

V. Conclusion

This CubeSat is capable of taking a 480x480 pixel image of Earth from orbit every minute over a duration of 24 hours. The communications architecture uses the long-range capabilities of LoRa to transmit the images to the ground station terminal. The CubeSat also analyzes its own status by measuring data transfer rate, charge capacity, and data storage. This data is then collected and sent to the ground station. Overall, the mission to produce a low-cost CubeSat meeting these requirements was a success. With the deep dive into microelectronics, programming, and wireless communication all the authors feel they have been able to advance their engineering knowledge and expertise and will continue to pursue new challenges that offer a chance to further scientific knowledge in the world. The design of this CubeSat may be small, but it has laid down the foundation for future research and experimentation.

VI. Acknowledgements

The authors thank Patrick Bailey and Shruti Jadhav for their guidance and support with navigating the early stages of the project. Their assistance helped to improve the design and allow for the successful creation of this CubeSat.

VI. References

[1] NASA, “What are SmallSats and CubeSats?” [Online].
[2] Instructables, "How to Build a CubeSat With an Arduino - Instructables," [Online].
[3] Arduino, "Arduino MKR WAN 1310," [Online].
[4] Meshtastic, "Meshtastic," [Online].
[5] Raspberry Pi. "Camera Module V2 - Raspberry Pi Documentation," [Online].
[6] Rawashdeh, Samir (2010). " PASSIVE ATTITUDE STABILIZATION FOR SMALL SATELLITES," Master's thesis, Department of Graduate School, University of Kentucky, Lexington, KY, USA. [Online].
[7] Arduino, "MKR WAN 1300 - LoRa Sensor Data," [Online].
[8] Arduino, "LoRaWAN 101: A Practical Guide to LoRa Communication," [Online].
[9] S. Projects, "Large Data Transfers with LoRa - Part 3," 2021, [Online].
[10] Oxford Reference, “G-force Quick Reference,” [Online]
[11] S. Projects,"SX12XX-LoRa GitHub Repository," [Online].
[12] M. Johnson, "A Simple Method to Estimate Signal-to-Noise Ratios for CubeSat Radios," NASA Technical Reports Server, 2015, [Online].
[13] Jirous, "Calculation Wi-Fi - Jirous," [Online].