Preliminary Design Document For RockSat-X Payload - Hephaestus

Helena Bales, Amber Horvath, and Michael Humphrey

CS461 - Winter 2017

February 9, 2017

Abstract

The Oregon State University (OSU) RockSat-X team shall be named Hephaestus. The preliminary design of our project shall be outlined in this document. The mission requires that the payload, an autonomous robotic arm, perform a series of motions to locate predetermined targets. The hardware shall be capable of performing the motions to reach the targets. The software shall determine the targets and send the commands to the hardware to execute the motion. The combination of the hardware controlled by the software shall demonstrate Hephaestus's ability to construct small parts on orbit. This document will focus on the implementation of the software, but shall include necessary project context including hardware.

Approved By _____ Date _____

Approved By _____ Date _____

Approved By _____ Date _____

Approved By _____ Date _____

Contents

1 Introduction

1.1 Document Overview

1.1.1 Helena Bales

1. Target Generation
2. Arm Movement
3. Arm Position Tracking

1.1.2 Amber Horvath

1. Emergency Payload Expulsion
2. Program Modes of Operation
3. Target Success Sensors

1.1.3 Michael Humphrey

1. Telemetry
2. Video Camera
3. Data Visualization and Processing

2 Technologies

2.1 Target Generation

2.1.1 Requirement Overview

The software shall generate points to be used in testing the Hephaestus arm. The points will constitute the total test of the arm, and should therefore include points representative of standard and edge cases. These points shall be used as targets for the arm body.

2.1.2 Solution Design

The points shall be generated in 3-D polar form, including an angle from normal, a radius, and a height. The angle shall be in the range of 0 and 359 degrees. An angle of zero degrees shall be in the direction of payload deployment. The radius shall be the distance from the arm's attachment to the base to the generated point. The height of the point, for the purpose of target generation, shall be constant. However the points will always be stored in a triple of angle from normal (θ), radius (r), and height (h).

The test points that are generated shall represent a sample of points over the range of motion required of the arm. As such the points should be at the extremes of where the arm can reach, in

the middle of the arm's range, and close to the arm base. Showing this full range of motion and the accuracy with which the range can be achieved will show the viability of construction on orbit.

The test points shall be generated prior to the launch. The test points will be generated by using a random number generator to pick a number in a range defined by which case the point is designed to test. For example, a point intended to test the arm's ability to reach near the base would generate an angle around the normal, a radius close to zero, and a height of zero. In this way, the generated test point will test a functionality of the arm. The test points will be generated prior to launch in order to insure that the points adequately cover the desired tests.

2.2 Arm Movement

2.2.1 Requirement Overview

The software shall control the movement of the arm body assembly. The position of the tip of the arm shall be tracked in the coordinate notation described in section 2.2 above. The software shall rotate the arm body assembly in a full 360 degrees. The software shall additionally control the movement the height of the arm body assembly. The arm should descend and touch the baseplate of the payload at any rotation.

2.2.2 Solution Design

The movement of the arm shall be generated by a custom system where the movement of the arm is generated based on the current position and the starting position. The position of the tip of the arm shall be stored in 3D polar coordinate form, as described above. The position shall be denoted as point p and shall be the location of the tip of the arm. The arm shall generate a series of commands for the motors to perform to go from p to the target, t_n where t_n is the n-th target. This command shall include a rotation command to rotate the entire arm to face the target θ value, and commands to each of the arm joint motors to bring the tip of the arm, p to the target radius, r, and height, h.

The movement of the arm shall be constrained in several ways in order to prevent damage to the hardware. The first constraint on movement is in the height of the arm. The movement shall be limited by the heights of the arm such that it will not collide with the top or base plates. This means that at no point should the height of the deployed arm exceed the height of the half can. This measure is meant to protect the hardware in case the payload gets stuck in any position and must be retracted. The second limit to the movement is in the rotation of the arm. The arm should never be allowed to perform more than a single full rotation. This safety measure is meant to keep the wiring of the arm from becoming tangled. The final safety measure that limits the movement of the arm is in the speed and torque allowed for the motors. Both of these values must be limited in order to insure the safety of our payload and the rocket. The velocity of the arm must be limited in case of collision to limit damages. The torque is limited to prevent damage to the arm, the payload, the rocket, and the motors. If the arm gets stuck, we will be able to detect it by measuring the torque that the motor must apply in order to move the arm. If the torque increases dangerously, we can stop, unstick the arm, and continue with the operations.

2.3 Arm Position Tracking

2.3.1 Requirement Overview

The position of the arm shall be tracked using the same coordinate system described in the Target Generation requirement. The position of the arm shall be calculated using the known start position and the rotation of the motors.

2.3.2 Solution Design

The position of the arm shall be tracked using the motor movement to calculate p and p_{m2}. The initial position of the arm shall be defined in advance, and a sensor will be placed at that location. From there, the position will be tracked from the movement of the motors. The arm will recalibrate by returning to the initial position in order for the error to not increase over time. The position of the arm shall be denoted p, the location of the tip of the arm. From the coordinate p, the location of p_{m2}, the center of the middle joint of the arm, will be calculated. The height of p_{m2} will be calculated from the triangle made of the two arm sections, L1 and L2, and the radius of point p. From there the radius of the point p_{m2} can be calculated using the triangle of L1, h_{m2} and the radius of m2. Finally, the σ of p_{m2} shall be the same as that of p. Using this method will allow for the extra condition that point p_{m2} should never exceed the height of the can. Constrain rotation to not go all the way around.

The position of the arm will be verified after the flight by using visual confirmation from the video camera. The purpose of tracking the position of the arm is to verify the accuracy of the arm on orbit. Since this will determine our ability to determine mission success, it is important that we have several methods of verifying out results. This design allows us to know where we want to be by storing the values of p, the position of the tip of the arm, that occur during the motion of the arm. We can also know where we are compared to where we started by storing the motion applied by the motors. We can know where we actually are through the triggering of sensors. Finally, we can verify the sensor data using the video camera.

2.4 Emergency Payload Expulsion

2.4.1 Requirement Overview

The software shall eject the arm upon system failure. System failure in this case is defined as the arm becoming lodged or stuck in a state where it is unable to retract. The software will enter Safety mode (defined in section 2.5.2) and attempt to retract the arm. If it is unable to complete this step, the system will continue attempting to eject the arm until ejection is completed

2.4.2 Solution Design

Upon entering the Shutdown state, the system should succeed in closing the arm, the Arm Assembly Body should be retracted, and the On-Board Computer (OBC) should be powered off. A time shall be implemented to detect whether the arm is stuck. The timer shall determine this by checking if a certain amount of time has passed between the last arm movement and the last request for an arm movement. If arm movement requests are not being met by arm movements, and the system stalls past a certain amount of time, the system should send a signal to enter the Safety state so

that the arm can be ejected, as it is most likely caught in a bad extended position. The system shall determine shutdown was not completed correctly (as seen in state 7 defined in section 2.5.2) by determining that one of these requirements was not met. The system shall determine the arm is not contracting properly by the amount of torque that the motor is applying, as failure to contract will require more torque. The system will fire an interrupt signal from the AVR interrupt library, notifying the system to transition to Safety mode. Safety mode will attempt to contract the arm once more by calling the arm movement function. The arm movement function will take a coordinate to move the end of the arm to. The arm is equipped with sensors that can determine if the arm is folded or not so if the sensors determine that the arm is folded, then safe shutdown should be possible. The emergency retracting operation is completed by turning off all the motors in the in the arm except for the motor pushing the whole metal plate the arm is attached to in and out of the payload. With those motors turned off, the joints of the arm will be flimsy and can be pulled into the payload by retracting the metal plate. In the case of contracting the arm, the tip should point inwards to the center of the canister. If it is unable to do so, it shall continue attempting to eject. The system shall initiate the arm ejection sequence by turning on the motor in control of ejecting part of the arm and turning off all other motors. The system shall also clean up any memory leaks and ensure all telemetry ports are closed upon sending the data that an emergency ejection was required. In the post-mortem analysis, information regarding the arm's expulsion will be useful. The system shall, upon receiving a signal that ejection is required, send a log description of the current polar coordinates of the arm, the time elapsed since last arm movement request, and what state the system was in prior to being sent to the Safety state. The system will continue attempting to eject the arm until the system detects that the metal plate has successfully returned into the payload. The system shall determine this by a pin being set from low to high upon entry into the payload. If the arm is unable to be ejected safely, the arm will be stuck outside the canister and the mission shall be counted as a failure.

2.5 Program Modes of Operation

2.5.1 Requirement Overview

The software shall have the Modes of Operation necessary to insure the mission success. The software shall first deploy the payload, then the arm. Next the software shall activate the camera and perform a video sweep. The software shall then perform the science experiment. If the experiment fails, it shall return to observation mode. If observation mode fails, it shall return to idle. Once the experiment time has been exhausted, the payload shall shut down. If it shuts down correctly, everything will poweroff. If not, the payload shall attempt to retract again, or expel the payload from the rocket.

2.5.2 Solution Design

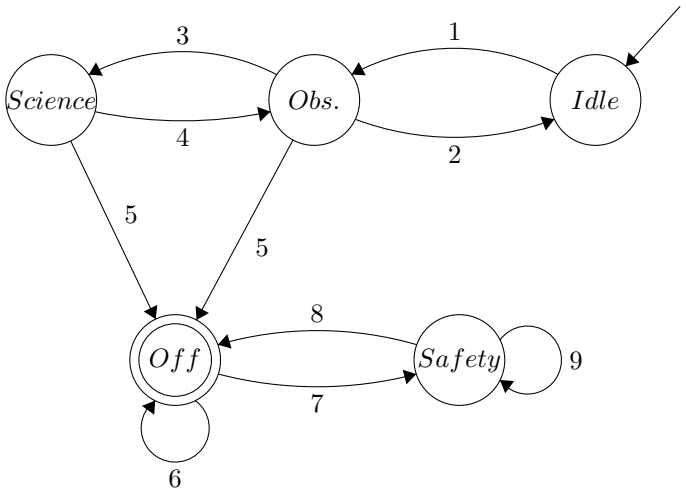

Diagram of software states of operation and transition between states [2].

Transitions between states occur as numbered:

1. **Appogee is reached.** The software shall activate when the power line goes to high at 28V. Observation mode shall be triggered when the OBC turns on. Observation mode will collect a sweep of the payload with the camera. This mode will ensure that the camera is operational during the more crticial parts of the mission.

2. **Error: Return to Idle.** If an error is encountered in entering Observation mode, the software shall fallback to Idle mode and retry. An error may occur if the payload fails to deploy correctly or if the camera fails to turn on. The system shall send a signal using the AVR interrupt library if the arm is not fully extended, as the arm is equipped with sensors to determine whether it is extended or folded. If the camera fails to turn on, the system shall be notified as the telemetry line will be sending empty data.

3. **Payload Assembly and Camera have been deployed.** The software shall enter Science mode once the payload assembly and arm have deployed and the camera has performed an observation sweep. Science mode will consist of the arm touching the sensors in the payload canister, and collecting data via the telemetry line. The whole mode shall be captured with the camera.

4. **Error: Return to Observation** The software shall return to observation mode if any error occurs in Science mode. An error may occur in Science mode if the arm fails to operate correctly and must return to default position. An error may also occur if the camera stops working. The system shall know if the arm fails as a timer can keep track of the time between an arm movement request and the arm actually completing the movement request. If too much time has elapsed between the request and the movement, the arm may be stuck. If the telemetry line stops receiving data from the camera, then the camera has stopped working and the system shall be notified via an interrupt.

5. **Timer switches to end appogee period.** Once the time period for observation has ended, the timer line will go to low and trigger to Shutdown state. This state can be reached from either Observation or Science mode.

6. **Accept: Shutdown correctly** If Shutdown occurs correctly, the arm should be closed, the Arm Assembly Body should be retracted, and the OBC should be powered off. The arm will be have sensors to detect whether its closed or not, which can also be used to know whether it has been retracted into the body. Once the system has determined that this criteria has been met, it will power off.

7. **Error: Shutdown not completed successfully.** If an error occurs in the shutdown sequence, the software shall enter Safety mode. An error that could occur is the arm failing to close, the Body failing to retract, or the OBC not powering off. All these situations except for the OBC not powering off are handled through Safety mode.

8. **Payload is Shutdown correctly.** If the payload is Shutdown through Safety mode, shutdown can be completed. In Safety mode, the payload was either shut down correctly, retracted fully into the can with the arm open, or the arm was expelled safely from the rocket. The shutdown sequence consists of the arm closing, the Body retracting, and the OBC being powered off.

9. **Error: Payload is still deployed.** The software shall remain in Safety mode until the payload is either retracted correctly, retracted fully with the arm in the open position, or ejected safely from the rocket. Safety mode shall first try to correctly retract the arm, then retract with the arm open, then repeat attempting ejection until the payload is ejected.

2.6 Target Success Sensors

2.6.1 Requirement Overview

The software shall know whether or not the arm succeeded in touching the targets generated, as described in section 2.1. The sensors shall report back whether or not contact was made. This data can be used in post-mortem analysis to determine whether certain targets were faulty or whether the range of motion on the arm was faulty.

2.6.2 Solution Design

The payload shall be equipped with pre-placed sensors that the arm shall make contact with. The arm shall have generated targets as described in section 2.1. These coordinates shall be stored within the system and used as inputs for the function controlling the arms' movements, with the target position being where the tip of the arm should be located. The arm shall exert force to touch the sensor, and the sensor shall go high if contact is made. The sensors high or low signal shall be sent via the telemetry line. If the arm gets stuck during this process, it will enter Safety mode, as described in section 2.5. The telemetry data shall later be visualized using Python's UI package, TK.

2.7 Telemetry

Author: Michael Humphrey

2.7.1 Requirement Overview

The telemetry component shall report via telemetry all error codes and test results.

2.7.2 Solution Design

The telemetry component is responsible for collecting and sending data through the telemetry ports on the payload.

This component shall not be responsible for transmitting data generated from the temperature sensors. The temperature sensors will be wired directly to an analog telemetry port, bypassing the OBC altogether.

For each test the software successfully completes (see section 2.6, Target Success Sensors) this component shall output a code representing the test number to the telemetry port. There shall be two tests; one for each touch point on the payload. The tests shall be designed as a joint effort between the Hephaestus Structures, Robotics, Electrical, and Software teams.

The output shall be encoded as a four character binary string and transmitted simultaneously via a four parallel port pins. The binary string shall be transmitted at a rate of no more than 5,000 Hz. There shall be a delay of no less than .2 milliseconds between transmissions of codes. When no code is being actively transmitted, the telemetry shall output a code of '0000' to the telemetry pins.

In addition to transmitting codes via the telemetry lines, the software shall also store a log file with timestamps on an onboard SD card. The log file shall consist of a series of lines of text, consisting of a timestamp and a description. The timestamp shall be the number of tenths of milliseconds since the OBC powered on. The description shall be a sequence of ASCII characters of arbitrary length, and terminated with a carriage return character.

2.8 Video Handling

Author: Michael Humphrey

2.8.1 Requirement Overview

Video footage of the payload's operations shall be recorded and saved to the SD card.

2.8.2 Solution Design

The Hephaestus Electrical Engineering team shall design the payload such that the camera will power on and off at the appropriate times, as well as save footage to the SD card.

2.9 Data Visualization and Processing

Author: Michael Humphrey

2.9.1 Requirement Overview

The data visualization and processing component shall provide visualizations for the collected data. This component shall be able to show whether the mission success criteria have been met or not. If the mission success criteria have not been met, this component shall show how and why they have not been met.

2.9.2 Solution Design

The component shall have a Graphical User Interface (GUI) written in Tkinter with graphs generated by matplotlib. The GUI shall consist of two graphs, a table, and a timeline. Each graph shall be a plot with analog data collected from each of two temperature sensors. The data from the temperature sensors shall be graphed with respect to time from apogee and actual temperature, if such a value can be determined. In the absence of a method to reliably determine actual temperature from the raw sensor data, then the data shall be graphed with respect to the raw value received from the sensor, with the graph scaled such that the lowest value recorded shall be the minimum y value, and the highest recorded value recorded shall be the maximum y value. The user shall be able to scale the graphs to view portions of the data as they see fit. The table shall consist of the name of each of a series of tests, the result of that respective test, and the time that test was completed. A result shall be either "passed", "failed", or "not completed". A result of "passed" shall be colored in green. A result of "failed" shall be colored in red. A result of "not completed" shall be colored in yellow. If the result of a test is "not completed", then the time of completion for that test may be omitted. There shall be two total tests. The tests shall be for if the payload can successfully touch each touch point sensor. The tests shall be designed as a joint effort between the Hephaestus Structures, Robotics, Electrical, and Software teams. The timeline shall be a visualization with time on the y axis, with significant events marked at various positions along the axis, according to when that event happened.

3 Conclusion

This concludes the design of our project. Further questions or concerns can be addressed to the authors of this document. This document may be subject to changes in the future as more design constraints are found, or designs are found to not work the way they were intended upon the creation of this document. The document shall be updated accordingly to account for these changes.

4 Glossary

apogee The point at which the rocket has finished its ascent and payloads are allowed to deploy. 11

ASCII American Standard Code for Information Interchange. Each alphabetic, numeric, or special character is represented with a 7-bit binary number. 128 possible characters are defined. 10

binary string An ordered sequence of 1's and 0's. 10

GUI Graphical User Interface 11

matplotlib A Python library for drawing and manipulating graphs. 11

OBC On-Board Computer 6, 8–10

OSU Oregon State University 1

payload A subsection of a rocket that is not essential to the rocket's operation. A payload is placed in a can, mounted on a standard base plate. A payload completes some specific task. 1, 10, 11, 13

plot An interactive window generated by the Python library matplotlib to display some dataset on an x and y axis. 11

5 Appendix

5.1 Mission Patch

Figure 2: Mission Logo [1]

5.2 Project Overview

The Hephaestus project is a Capstone Senior Design project for Oregon State University's 2016/2017 Senior Design class (CS461-CS463). The CS senior design project is one part of the overall Hephaestus project. In addition to the CS team, there is one team of Electrical Engineers and two teams of Mechanical Engineers working on this project through other senior design classes. The Hephaestus payload is a rocketry payload developed as part of the 2016/2017 RockSat-X program. The RockSat-X program is a year long program where groups of students develop rocketry payloads with

the help of the Colorado Space Grant Consortium and Wallops Flight Facility. The term "rocketry payload" refers to an experiment inside a section of the rocket. Each section of the rocket is called a can, and is a standard space that we can fill with an experiment. The Hephaestus payload shall take up half a can and shall be mounted on a standard base plate provided by Wallops. We, as the Hephaestus team, will create the hardware and software for the payload, then integrate it into the rocket before launch.

5.2.1 Project Phases

The project shall include several phases. The first is the design phase. The design phase shall last all of Fall 2016 term at OSU. In the design phase, we shall design the robotics, electronics, materials, and software. The design phase shall include presentations to the RockSat-X program, where there will review our designs. Following the design phase will be the implementation phase. In the implementation phase we shall last through June 2017. This phase shall include testing of the payload. We will perform testing both at OSU and at Wallops. At OSU we will be testing the payload functionality. At Wallops, we will be testing the structural integrity of the payload, as well as its resistance to vibrations, heat, and cold. Following the implementation phase will be the integration phase. This phase will occur at Wallops in July. This is the point at which our base plate will be integrated into the rocket as a whole, along with the other participating teams. The final phase will be launch. Launch will occur in Summer of 2017. The rocket shall be launched from Wallops Flight Facility. During the flight we shall send telemetry to the ground station at Wallops. The payload shall perform the experiment once it reaches appogee. The payload will hopefully be recovered post-flight.

5.3 Software State Diagram

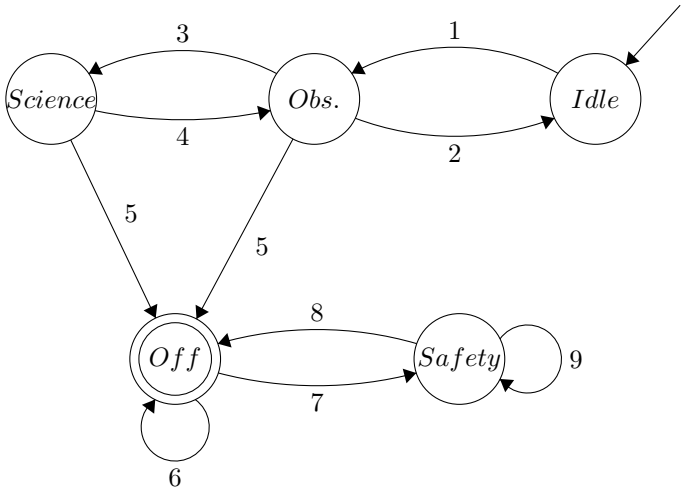

Figure 3 – Diagram of software states of operation and transition between states [2].

Transitions between states occur as numbered:

1. **Appogee is reached.** The software shall activate when the power line goes to high at 28V. Observation mode shall be triggered when the OBC turns on.

2. **Error: Return to Idle.** If an error is encountered in entering Observation mode, the software shall fallback to Idle mode and retry. An error may occur if the payload fails to deploy correctly or if the camera fails to turn on.

3. **Payload Assembly and Camera have been deployed.** The software shall enter science mode once the payload assembly and arm have deployed and the camera has performed an observation sweep.

4. **Error: Return to Observation** The software shall return to observation mode if any error occurs in Science mode. An error may occur in Science mode if the arm fails to operate correctly and must return to default position. An error may also occur if the camera stops working.

5. **Timer switches to end appogee period.** Once the time period for observation has ended, the timer line will go to low and trigger to Shutdown state. This state can be reached from either Observation or Science mode.

6. **Accept: Shutdown correctly** If Shutdown occurs correctly, the arm should be closed, the Arm Assembly Body should be retracted, and the OBC should be powered off.

7. **Error: Shutdown not completed successfully.** If an error occurs in the shutdown sequence, the software shall enter Safety mode.

8. **Payload is Shutdown correctly.** If the payload is Shutdown through Safety mode, shutdown can be completed. In Safety mode the payload was either shut down correctly, retracted fully into the can with the arm open, or the arm was expelled safely from the rocket.

9. **Error: Payload is still deployed.** The software shall remain in Safety mode until the payload is either retracted correctly, retracted fully with the arm in the open position, or ejected safely from the rocket. Safety mode shall first try to correctly retract the arm, then retract with the arm open, then repeat attempting ejection until the payload is ejected.

5.4 References

[1] Oregon State University RockSat-X Team, "Hephaestus Mission Patch," 2016. [Online]. Accessed: June 14, 2016.

[2] H. Bales and M. Humphrey, "Diagram of Software Modes of Operation," 2016. [Online]. Available: Hephaestus Requirements Document.