Rocket Telemetry - Hardware, Software Integration and Testing

Students: Darnell Richards, Dayja Young

Advisors: Dr. Farzad Moazzami, Dr. Richard Dean, Wondimu Zegeye

Morgan State University, Electrical and Computer Engineering Department

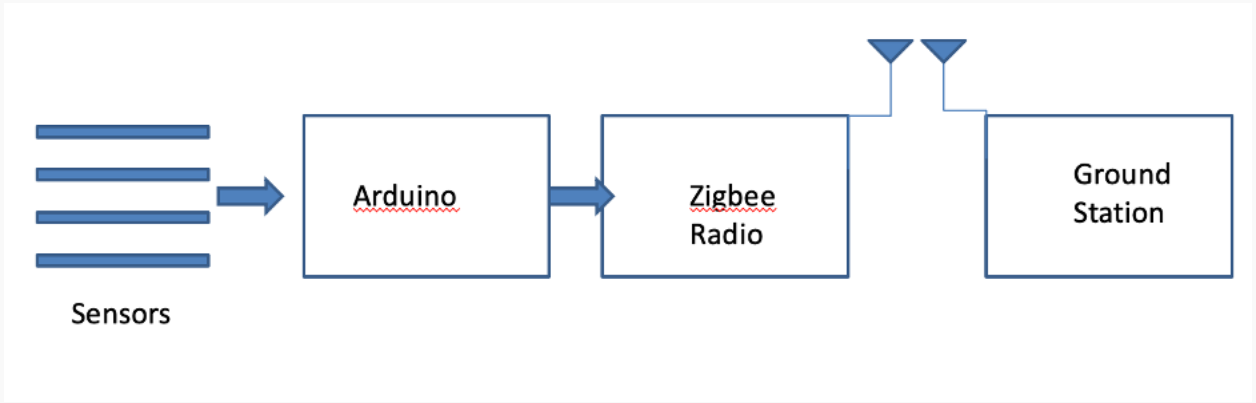
ABSTRACT

This paper is a continuation of the previous papers: “Rocket Telemetry - Physical and Functional Design”, and “Rocket Telemetry - Software and Functional Design”. The paper will capture our efforts of hardware, software integration and testing. The software will capture and transmit data from accelerometers and gyroscopes on the module to track the trajectory of the rocket after launch. Morgan State has received a $1.6 million aerospace grant that will allow the school to complete a liquid-fuel rocketry lab and to recruit and hire a faculty aerospace leader to create a world-class program in liquid fuel. The school is looking to build and launch a liquid fuel rocket that can reach 150,000 feet.

INTRODUCTION

Morgan State University's Wireless Network and Security Lab (WiNetS) is under the Electrical Engineering Department. The WiNetS lab has a focus on cyber security. We are tasked with creating a rocket telemetry module that provides measurements ranging from altitude, direction, heading, distance and location. The module is outfitted with a microcontroller, two digital radios, a sensor board, and a GNSS (GPS component). The telemetry package will consist of a ground unit and onboard unit that communicates by radio signals that will relay the information to the ground unit. The program implemented within the Arduino will allow the information to be displayed through the utilization of MatLab.

The purpose of the telemetry module is to provide comprehensive and readable data during the flight of the rocket. Our module will also provide data that can be used to account for any discrepancies during flight after it has been collected. Tracking the trajectory of the rocket is the major objective of the telemetry module to properly gauge the capabilities of the rocket and possibly any aerodynamic deficiencies. To accomplish this the use of sensors such as the accelerometers and gyroscopes are necessary to collect major deviations within its flight pattern.



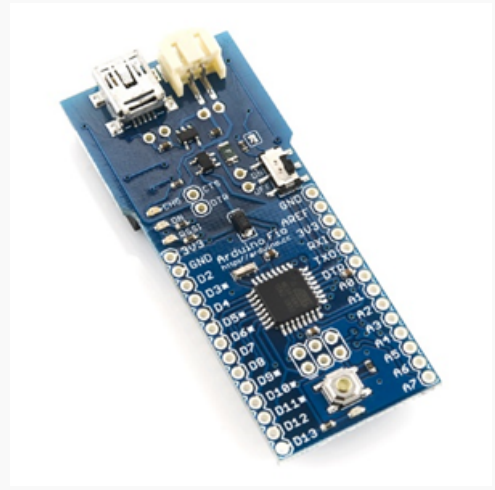
HARDWARE MODULES AND FUNCTION

Assembly of the rocket telemetry module is intended to be approached with the idea of constructing a system that can be compact, heat tolerant, and rigid to function during take-off and landing phases of the launch. Solder pins components were selected to achieve connections that would not be disconnected or undergo interference. The microcontroller: Arduino Fio, would utilize and be placed in the center of the module while being connected to a power source. This in turn, provides power to the other components wired to the controller. The digital radios: Xbee S2C, consist of a coordinator and a receiver. The coordinator would be the bridge to the router that is connected to the onboard module, that relays the information being recorded by the module to the ground station. The router relays information from the accelerometer and GNSS (GPS). The sensor board contains three axes of reference, and each axis contains one accelerometer, one magnetometer, and one gyroscope. The GNSS is the GPS component that allows for tracking of the module's velocity and position. Each component would be appropriately connected and tested before completely soldering and encasing.

The objective of the telemetry project is to create a light weight and optimally functional system. To achieve that, components size and weight were taken into consideration, so that the module does not contribute to any interference in the flight pattern rocket. Also, voltage and amperage operating values were also taken into consideration with consideration of the power supply being used and capabilities of the components in use. Total weight of the system is estimated to be 50 grams, and the total approximate current being drawn is 367 mA

Components

Arduino Fio v3 – The brain of the telemetry module, and an open-source micro-controller that allows a simple way to interface with sensors in the telemetry circuit. The Arduino is a versatile component that contains the required connections to interface with the sensors being utilized. The connections are inter-integrated circuit(I^2C) protocol for the Altimu-10, Universal Asynchronous Receiver Transmitter (UART) for the uBlox Max-M8Q GNSS receiver, and Serial Peripheral Interface Bus (SPI) for the Adafruit SD. The Arduino can be utilized at 8 MHz at 3.3 V , and 16 MHz at 5V. A higher clock speed would be preferred for faster computations, however the effective baud rate (Bits per second) is determined by the subdivisions of the clock. Since the internal clock of the Xbee is 16 MHz, it would be beneficial to match the Arduino, however this will lead to increased power consumption. The mass of the Arduino is 9 gram.



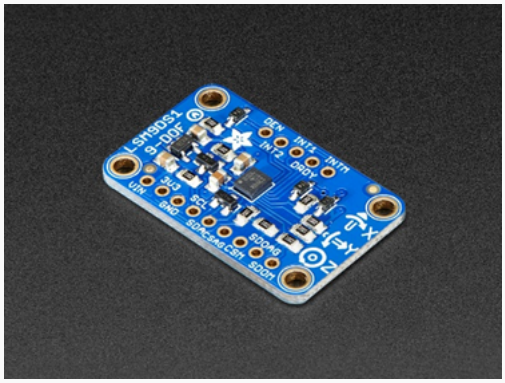
Xbee S2C – This is a remote communication device that allows for the passing of information at greater distances and higher altitudes. This is a line of sight transmission device that is said to reach a maximum distance of 1500 meters. Two Xbee radios are utilized, a Coordinator and a Router, to establish communication between a ground station and the module. The coordinator is the device that establishes the network to communicate to the router to receive the information from the router by way of a destination address specific to the device itself. Fortunately, this will be a point to point network so establishing an IP address is not required. The operating voltage for the xbee is within the range of 2.7 -3.6 V. The operating current to transmit is 117 mA and the

operating current to receive is 47 mA. The mass of the Xbee is 10 grams.

Adafruit Mirco SD Card Reader – This is an important device to read and record data for analysis and documentation. However, the actual SD card can be tricky to choose to ensure proper operating current consumption. The power consumption is not easily measured but is safely assumed through documentation from many other users that power demand should not exceed 150 mA. Thus, the Lexar 8 GB SDHC was chosen for the module. The Arduino is equipped with a library called SdFat.h that allows the microSD cards to achieve great writing speeds. The mass of the micro SD is 5 grams.

Altimu LSM9DS1- This is an accelerometer, magnetometer, and gyroscope breakout board with three axis. Each axis contains one of each sensor. The accelerometer is responsible for measuring vibration or the acceleration of the module. The gyroscopes are used to provide stability or maintain a reference direction in navigation systems. Each sensor can be set with specific ranges to provide accurate reading depending on the size and force applied to the module itself. The

optimal power output of the breakout board is 2.5-5.5V and 6 mA.



GNSS Receiver (GPS)- This component tracks the velocity and position of the module. This is useful for studying the trajectory of the rocket throughout its parabolic flight. It is noted that this component can provide a continuous transmission which would cause issues for the information from the other sensors transmitting data simultaneously, but since the u-blox MAX-M8Q uses a byte based system it can parse the information being received. The consumption of the GNSS is 22mA and 3-5.5V.

.

Power Supply – Given the range of most of the component voltage consumption, a 3.3V battery was chosen to power the module. It is the best option to provide a constraint to reduce rapid consumption of the power source, prevent part failure, and provide sufficient power to use throughout a launch. At the time of the test a power supply with a voltage regulator was used for continuous use.

SOFTWARE MODULES AND FUNCTION

Every piece of hardware requires software to tell each component what to do. The overall purpose of the software is to transmit and receive data. The software is made up with programs for the rocket segment and the ground station segment. This software is used to transmit data between the two using the languages Arduino, C++, Python and MatLab. These programs will operate by sending and receiving GPS sentences using a main loop along with some data handling code.

To have a completed telemetry model we must have running software. Using the Arduino code that outputs the GPS Sentence. It uses an Arduino IDE Software. The Arduino has limited computational capacity along with limited memory. UART pins use Rx for GPS and Tx for XBee.There are three major functions within the code which include reading the GPS UBX sentence, calculating attitude, and outputting the data. The number of libraries used is limited to avoid stackoverflow. Libraries used are SDFat, SPI, WIRE, NeoHWSerial, L3G, LSM303, and LPS. The data is self-contained in 32 bit intervals which makes the data readable by the union of a byte array and a structure. The GPS sentence is an output if-else statement.

The other part of the code includes receiving the GNSS, UBX, NAV, and PVT sentences. To read the PVT, the code runs without an interrupt connected to the Rx UART pin. The interrupt code is in the algorithm. The main loop of this code runs at 50hz, every 20 milliseconds it reads data from the Polou Altimu sensors. The main loop uses the AHRS algorithm to calculate attitude. If the code prints “true” it updates the output sentence and sets the flag to “false”. The system prints a stream of 80 bytes from the structure to the XBee and SD card.

The ground segment consists of a telemetry base station which uses a C++ code. This station records and reads data. The code was developed in a multi-platform IDE for C++ programming. The telemetry station had three main parts which include: receiving and logging off the message, converting data, and plotting vectors and plotting UI. This function logs the sentence into a binary file then passes it onto the vector creating thread. The main thread displays the graphs and options of the UI. This allows users to choose which plot to display. Another code used was Python, to convert binary to CSV code because of built in structure casting and printing. This code allows for easier post processing.

The final code used is a MatLab data handling code. MatLab was used for figure plotting. The code takes inputs from a comma separated value file (CSV), extracts data, and applies factors to get measured parameters. The code also obtains time vectors for GPS and sensor data values. This script was used for post processing and analyzing data.

HARDWARE SOFTWARE INTEGRATION

The telemetry hardware module was constructed with the use of proven methods and modern components. With the Arduino Fio being the center of the module, the connection of the other components are dependent on the interface in use. The module being implemented uses the inter-integrated circuit (I^2C) to communicate with the other components in the simplest method. The pins SDA and SCL of the arduino are used for integrating the accelerometer. The communication between the accelerometer and arduino are programmed to transmit and receive 3-dimensional information and display onto a window accessible through the python programming language. The accelerometer data is also utilized in MATLAB to analyze the information through graphs and charts. This is useful to make adjustments to all necessary equipment.

The microSD is a component that handles the storage of data and allows the module enough data space to process data being transmitted and received throughout the various stages of the rocket's launch. The connection for the communication between the arduino and card reader is the SPI protocol. The connection using SPI consists of four pins Serial Clock (SCLK) (output from master), Master Output, Slave Input (MOSI) (output from master), Master Input, Slave Output (MISO) (output from slave) and Slave Select (SS) (active low, output from master). The SdFat.h library is called within python to write a programmable script to optimize processing speeds in order to utilize small amounts of memory.

The global Navigation satellite system (GNSS) module utilizes the serial pins on the arduino to create the circuit. The connection is made from the transmit port (Tx) from the GNSS to the Recieve (Rx) pin onto the arduino. The python program script depends on the region of operation in accordance with GNSS committee protocols. This circuit allows for an additional method of observing trajectory and tracking the precise position of the rocket when grounded.

The Xbee radio transmitter is the bridge for ground communication to the rocket module. The connection to the arduino from the receive (Rx) to the transmit (Tx) on the xbee. The ground station Xbee is connected via USB connection, and is operated through the UART programming interface and python programming language. The configuration of the xbee can affect the code necessary to maximize the transmission by upgrading the antenna to increase the line of sight (LOS) range of the transmitters.

The most challenging part of our hardware and software development was debugging the Arduino Fio and its sensor code. The biggest error was actually a hardware error. We had to change the pins from the Arduino Fio to match what would have been a regular Arduino Uno. Because of this we also had to change the corresponding pin numbers in the code. It took many tries before actually understanding this. Working with Arduino Fio was pretty challenging because most of the altimu connections were explained for regular arduinos.

RESULTS

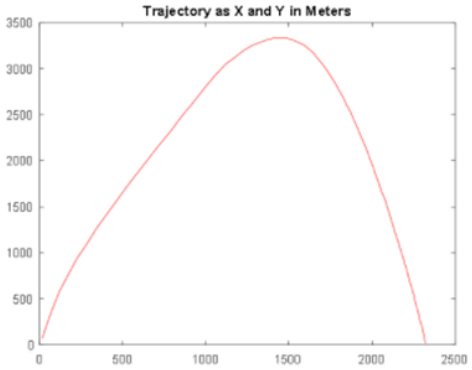
By using basic trajectory physics such as projectile motion and the force of drag we can create an environment to see how the rocket would behave in perfect conditions.
Distance in the X-direction: Dx(t) = x0 + Vx0(t) + .5Ax(t)2
Distance in the Y-direction: Dy(t) = y0 + Vy0(t) + .5Ay(t)2
A = Acceleration = dV/dt = F (force)/M (mass)
The Force do to Drag: F_d= (pACV^2)/2
P = density of medium = 1.23 Kg/m^2
A = cross section area = 2 sq meters
C = .1 for the shape of the rocket's nose (a pointed shape)
Force (unto the rocket): Fp (Propulsion) – Fg (gravity) – Fd (Drag)

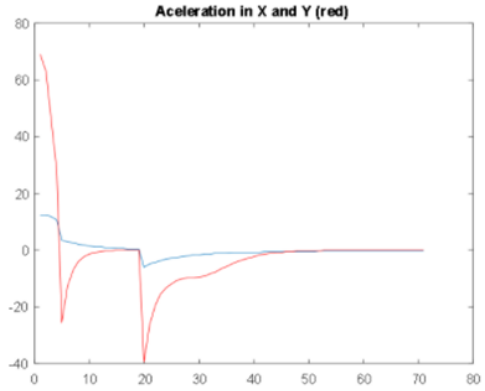
Also, the effect of the rocket's mass due to fuel loss:
If t <= T(2)

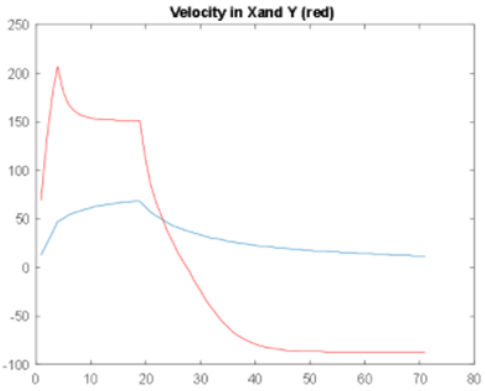
```
            Tk(t) = (1-Ratio*(t/T(2))); %Lose Mass
            M(t) = Mass_full*Tk(t);
Else  M(t)=M(t-1);
end
Mass=M(t0;
```

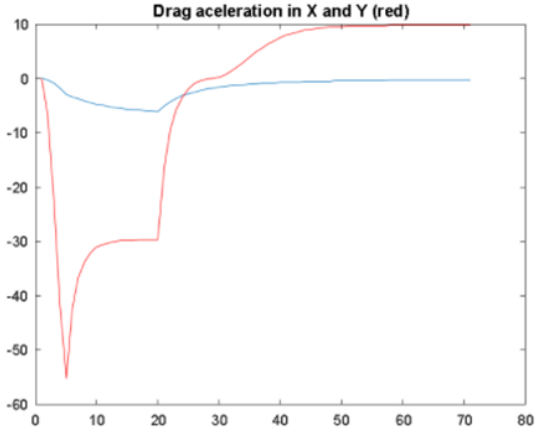
```
Mag X: -0.00 uT Y: 0.00 uT      Z: -0.09 uT
Gyro X: ovf rad/s       Y: ovf rad/s    Z: -83056512.00 rad/s

Accel X: 0.00 m/s^2     Y: 0.00 m/s^2   Z: ovf m/s^2
Mag X: -0.00 uT Y: 0.00 uT      Z: -0.09 uT
Gyro X: ovf rad/s       Y: ovf rad/s    Z: -83056512.00 rad/s

Accel X: 0.00 m/s^2     Y: 0.00 m/s^2   Z: ovf m/s^2
Mag X: -0.00 uT Y: 0.00 uT      Z: -0.09 uT
Gyro X: ovf rad/s       Y: ovf rad/s    Z: -83056512.00 rad/s

Accel X: 0.00 m/s^2     Y: 0.00 m/s^2   Z: ovf m/s^2
Mag X: -0.00 uT Y: 0.00 uT      Z: -0.09 uT
Gyro X: ovf rad/s       Y: ovf rad/s    Z: -83056512.00 rad/s

Accel X: 0.00 m/s^2     Y: 0.00 m/s^2   Z: ovf m/s^2
Mag X: -0.00 uT Y: 0.00 uT      Z: -0.09 uT
Gyro X: ovf rad/s       Y: ovf rad/s    Z: -83056512.00 rad/s

Accel X: 0.00 m/s^2     Y: 0.00 m/s^2   Z: ovf m/s^2
Mag X: -0.00 uT Y: 0.00 uT      Z: -0.09 uT
Gyro X: ovf rad/s       Y: ovf rad/s    Z: -83056512.00 rad/s

Accel X: 0.00 m/s^2     Y: 0.00 m/s^2   Z: ovf m/s^2
Mag X: -0.00 uT Y: 0.00 uT      Z: -0.09 uT
Gyro X: ovf rad/s       Y: ovf rad/s    Z: -83056512.00 rad/s

Accel X: 0.00 m/s^2     Y: 0.00 m/s^2   Z: ovf m/s^2
Mag X: -0.00 uT Y: 0.00 uT      Z: -0.09 uT
Gyro X: ovf rad/s       Y: ovf rad/s    Z: -83056512.00 rad/s

Accel X: 0.00 m/s^2     Y: 0.00 m/s^2   Z: ovf m/s^2
Mag X: -0.00 uT Y: 0.00 uT      Z: -0.09 uT
Gyro X: ovf rad/s       Y: ovf rad/s    Z: -83056512.00 rad/s
```

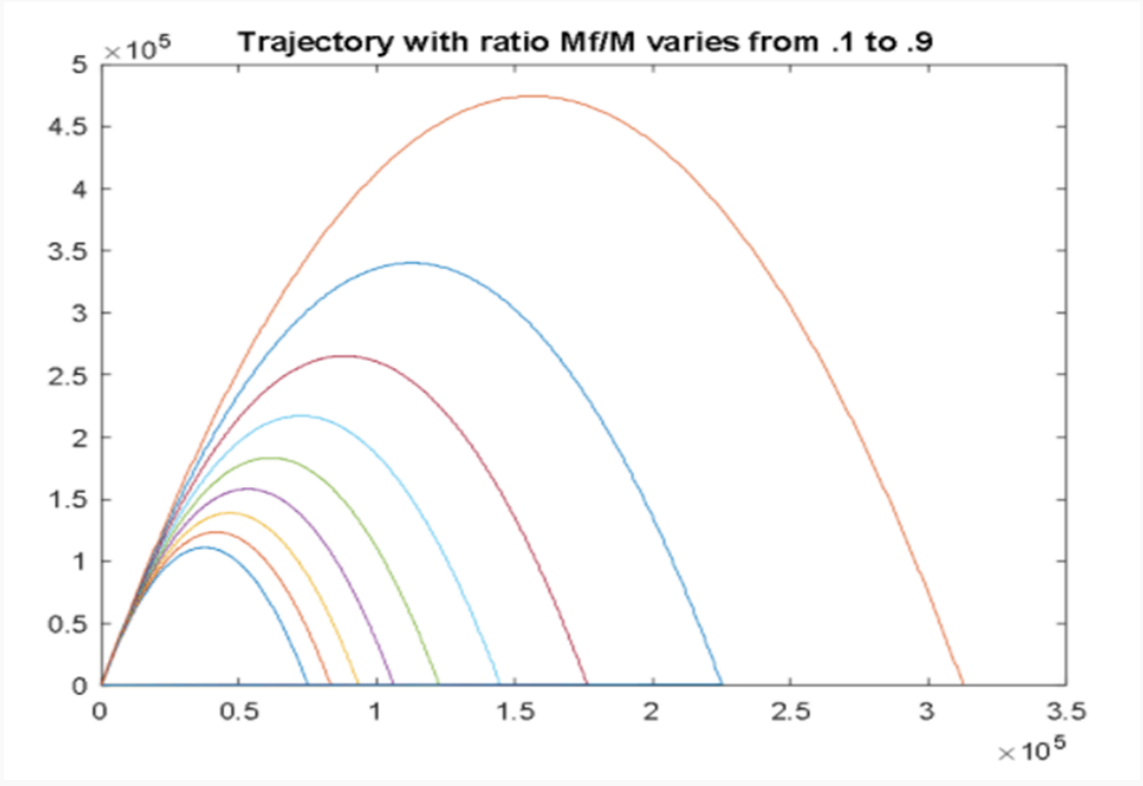
Simulation


Trajectory as X and Y in Meters
3500
3000
2500
2000
1500
1000
500
0
0
500
1000
1500
2000
2500


Aceleration in X and Y (red)
80
60
40
20
0
-20
-40
0
10
20
30
40
50
60
70
80


Velocity in Xand Y (red)
250
200
150
100
50
0
-50
-100
0
10
20
30
40
50
60
70
80


Drag aceleration in X and Y (red)
10
0
-10
-20
-30
-40
-50
-60
0
10
20
30
40
50
60
70
80


Trajectory with ratio Mf/M varies from .1 to .9
×10^5
5
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
0
0.5
1
1.5
2
2.5
3
3.5
×10^5

CONCLUSION

The module is curated with tried and tested methods with the intentions of making modifications specific to the rocket's requirements. Progression has been pushed back this past year due limitation of access to parts and labs, however the setup with establishing communications with the Arduino and radios have been successful. By late august or early September a full assembly will be operational and ready for possible modifications and intense testing under required conditions. The xbee communication has been tested and accomplished through the UART programming. It has been established that the 4000 ft limitation will need to be upgraded to meet the needs of 350,000 ft launch via antenna dongle.

FUTURE WORK

There are three phases set in place to complete the module and allot room for upgrades. Phase I, is the bench testing phase for all component integration and calibration for proper operation based on conditions and location. This will be accomplished by using a 3-axis tripod for preliminary testing to ensure calibration of the module. Testing will include static testing, vibration testing, and line-of-sight to ensure proper calibration and connection. Phase II, the module will be placed upon a drone for testing under movement conditions such as vibration, g-force, and line-of-sight limitations such connectivity under rapid movement and frequency traffic. The second phase's purpose will be to identify possible upgrades to improve distance and operation. Phase III, is the phase where the module is implemented onto the rocket.

REFERENCES