University of North Dakota Advanced Rocketry Club

Avionics Team

Preliminary Design Review



Contents

[Overall System Architecture 3](#_Toc2765660)

[Electronics and Altitude Monitoring 3](#_Toc2765661)

[Avionics System Overview 3](#_Toc2765662)

[Position Monitoring 3](#_Toc2765663)

[Global Positioning System (GPS) 4](#_Toc2765664)

[Barometric Altimeter 4](#_Toc2765665)

[Doppler Effect Speed Measurement 6](#_Toc2765666)

[Inertial Navigation System 6](#_Toc2765667)

[Telemetry 7](#_Toc2765668)

[Recovery 9](#_Toc2765669)

[Engine Control and Monitoring 10](#_Toc2765670)

[Software 11](#_Toc2765671)

[Ground Control 12](#_Toc2765672)

[Tracking & Receiving 13](#_Toc2765673)

# Overall System Architecture

Our avionics system will be a low-budget, yet capable and safe platform. Since our exposure to harsh environment time is very short, we can reasonably deploy commercial off-the-shelf (COTS) hardware at a very low cost and low weight. For example, the Raspberry Pi (RPi) and BeagleBone Enhanced (BBE) single-board computers (SBCs) are both about the size of a credit card and weigh less than a typical smartphone. Both are fully featured Linux computers, and the BBE boasts a gigahertz processor with 2 embedded programmable real-time units (PRUs) along with an onboard barometer and 6-axis accelerometer.

Our responsibilities include active management of the engine and its systems, data acquisition, telemetry, abort systems, and recovery device deployment. Our rocket will not be using any form of guidance, navigation, and control (GNC) systems due to competition restrictions. We will achieve each of these requirements with appropriate hardware as described in upcoming sections.

# Electronics and Altitude Monitoring

While we’re still in the research phase of our design, we have several options being considered for data acquisition and position determination. Some of the solutions we’re considering include the Global Positioning System (GPS), using the Doppler effect to measure speed, Inertial Measurement Units (IMU), and more. Since we’re still so new to the design requirements and have a relatively tight budget we haven’t yet settled on a finalized design; however, we have a reasonably structured draft at this time and it will be presented below.

## Avionics System Overview

There are many parts that combine to form one avionics system. As outlined in our responsibilities above, we will have one or more subsystems to fulfill each of those requirements.

### Position Monitoring

Keeping track of the rocket is vital for multiple reasons, primarily safety. Knowing where the rocket is laterally and vertically allows mission controllers to quantify any risks that may be present and making an educated abort decision at any time. It also allows us to be certain that we’ve met our goal of reaching 100 kilometers, collect data for research purposes. However, determining where a rocket is can be quite difficult at times since it is not an often-solved problem. We plan to use a sensor-fusion approach to take data from multiple sensors/subsystems and use that data to provide an overall navigational picture.

|  |  |  |
| --- | --- | --- |
| **Type of device** | **What it provides** | **When it’s useful** |
| GPS receiver | Position (3D), velocity (3D) | Within CoCom limits |
| Barometer | Altitude, vertical speed | Approximately bottom third of the atmosphere |
| IMU/Inertial Navigation System (INS) | Acceleration (3D), can be integrated to provide velocity (3D) and position (3D) | Entire flight |
| Doppler effect velocity measurements | Velocity (3D) | Entire flight |

Table 1: Position determination data sources

#### Global Positioning System (GPS)

The obvious solution to target tracking, altitude and velocity monitoring is to use the GPS constellation. There are, however, some problems with this. The United States places export regulations on GPS receiver hardware to prevent foreign powers from using them to build inter-continental ballistic missiles (ICBMs). The result of these export regulations (called CoCom or COCOM, for Coordinating Committee for Multilateral Export Controls) is that the receivers automatically disable themselves outside of certain conditions. Most of the firmware manufacturers disable above 1000 knots OR 18 kilometers, however certain vendors will use an AND gate instead of OR. Unfortunately, any vehicle capable of reaching space (not even orbit, just 100km altitude) will *easily* break these conditions. 1000 knots is slightly under Mach 1.5 at sea level, and 18 kilometers is well short of the desired 100. Even with an AND-gated device, preliminary calculations from our propulsion team indicate that our vehicle will be crossing 18km at approximately Mach 1.57 (1770 ft/sec), disabling the GPS receiver.

There are some reports of devices that can easily have their CoCom limits disabled by a firmware “hack,” or modification, allowing them to continue functioning well outside of those limits. However, since this is a violation of international law, we will not be using this method.

What we can use GPS for is augmentation while within the CoCom limits. We plan on using the GPS receiver to cross-check the INS unit and provide any necessary calibrations on-the-fly. Since it provides 3-dimensional position, we can derive velocity and acceleration (although the latter may take a few seconds to gather enough data). This data provides augmentation and cross-referencing for INS and barometric subsystems up to the limit of 18km. We can use this data to determine errors between the systems, store those differences, and apply them over the rest of the flight.

#### Barometric Altimeter

A basic barometric altimeter has been used in aviation since 1928. Since the atmosphere’s pressure decreases in a predictable manner, we can determine the vehicle’s altitude above mean sea level as a direct function of atmospheric pressure. These functions have been well documented as they’re critical to aviation, and therefore we can implement them directly with very little work. Barometric sensors are cheap and readily available; the MS5803-14BA unit is rated all the way to and including 0 mbar of pressure, and there are breakout boards available for both Serial Peripheral Interface (SPI) and Inter-Integrated Circuits (I2C). In fact, the code to interpret the altimeter data is already available.

There are, unfortunately, restrictions on the altimeter’s use. Since the barometer requires atmospheric pressure to function, it will only be reliable during a portion of the flight. We estimate that it will be approximately the lower third of the atmosphere, however, we also plan on running complete vacuum chamber tests to determine exactly where the barometer becomes inaccurate. Looking at the International Standard Atmosphere (ISA) data that can be found online, it appears reasonable to expect the altimeter to stop functioning accurately at a little above 30 kilometers: its resolution is 0.2 mbar and the expected pressure at 32km is approximately 0.0085bar.

Using a barometer to aid GPS altitude calculations is quite common in aviation; a process called baro-aided receiver autonomous integrity monitoring (RAIM). In this procedure, corrections are made during the flight based on known pressure settings at different airports. From these corrections, a barometer can be used to accurately determine altitude above mean sea level with impressive precision & accuracy. For pilots, this is the primary means of determining altitude, almost to the exclusion of GPS altitude reports.

Comparing the barometric altimeter to using GPS altitude, certain restrictions apply to both and when they an be used. The GPS altitude tends to be less precise but will be accurate to a higher altitude. Barometric altitude, on the other hand, will be more precise but less accurate with higher altitudes. By varying the precision we read from the altimeter as altitude increases, it should be possible to read accurate and precise pressures up to 200,000ft, or almost 61 kilometers. A snippet from the ISA data table shows the critical part of the atmosphere where the altimeter is no longer precise enough:

|  |  |  |  |
| --- | --- | --- | --- |
| Altitude (ft) | Temperature (°Rankine) | Pressure (millibar) | σ (pressure@alt / sea-level-pressure) |
| 185000 | 462.5 | 0.741700 | 0.000350500 |
| 190000 | 455.0 | 0.606900 | 0.000286800 |
| 195000 | 447.4 | 0.494900 | 0.000233900 |
| 200000 | 439.9 | 0.402300 | 0.000190100 |
| 205000 | 432.4 | 0.325800 | 0.000154000 |
| 210000 | 424.8 | 0.262900 | 0.000124200 |
| 215000 | 417.3 | 0.211400 | 0.000099900 |
| 220000 | 409.8 | 0.169300 | 0.000080010 |

Since the barometric altimeter is accurate up to these altitudes, we can use it alongside the GPS to calibrate and correct the INS in the same manner.

Table 2: Comparison of GPS vs. barometer

|  |  |  |
| --- | --- | --- |
| **Metric** | **Barometer (MS5803-14BA)** | **GPS/GNSS (MKT3339)** |
| Size | 20 mm / 18 mm / | 25.5mm / 35mm / 6.5mm |
| Weight |  | 8.5g |
| Breakout board interfaces | 1xI2C; 1xSPI | 1xUART |
| Lateral position measurement accuracy | No lateral position provided | Dependent on available GNSS constellation. Optimum: 0.715m Worst-case: 7.8m |
| Vertical position measurement accuracy | 1mbar in 0.5millisec  0.6mbar in 1.1millisec  0.4mbar in 2.1millisec  0.2mbar in 8.22millisec | Typically, worse than lateral position accuracy. VDOP values are higher due to all measurement satellites being above the receiver. Near impossible to quantify. |
| Altitude performance degradation? | Yes – as altitude increases, barometric pressure derivative steepens | No – as altitude increases, more satellites come into view |
| Altitude Limits | Until lack of precision from pressure gradient | CoCom limits – reportedly OR-gated |
| Power draw |  | Navigation mode: 20 mA |
| Code available for use? | Yes | Yes |
| Data sampling frequency | SPI: 20 MHz  I2C: 400 kHz | UART: 10 Hz (not accounting for NMEA string processing) |

#### Doppler Effect Speed Measurement

The Doppler Effect is a well-documented phenomenon which causes the frequency of a radio transmission to shift as a function of velocity. For example, if a receiver in the path of a vehicle travelling at high speed is expecting to get a frequency of, for example, 433 MHz, it may in fact experience a frequency 65 Hz higher than that. The formula for calculating velocity given the other variables (initial frequency and received frequency) is available online and easily implementable in code as it requires no integration or derivation and is a constant-time function.

Even better, we likely won’t need any additional hardware to use the Doppler effect to our advantage for measuring speed. Our vehicle will be using a radio frequency (RF) link to send telemetry data back, and the Doppler effect speed measurement system can simply run on top of that physical hardware. It will likely be entirely a software product, simply running on the ground control machine to compute vehicle velocity. Since the vehicle does not have any need to know its velocity, there is no need for a return transmission to send that information back to the vehicle, further simplifying equipment requirements.

#### Inertial Navigation System

Inertial navigation systems have been used in high-fidelity aircraft for quite some time. The system relies on a set of accelerometers, preferably 6- or 9-axis. The angular accelerometers are used for determining orientation, by integrating the angular acceleration twice with respect to time to get current orientation. The same idea is applied to the linear accelerometers to determine current position. Since the starting position, orientation, and velocity (rotational and linear) are known, there are no unknown constants left to be solved for.

Further performance increases are given by precomputing the integration formulas and simply plugging in constants. This reduces the operation to constant-time functions, which are easy to compute and can be done in a fixed amount of time no matter the values (within reason). By using a 9-axis unit such as the BNO055, we can also gain access to a triaxial geomagnetic sensor which allows continuous orientation updates and correction.

Unfortunately, INS units sometimes suffer from drift errors. Since there are a certain set of constants only known at one point (launch), any error in the accelerometers is further magnified through the double integration. Unfortunately, most units are unique in their errors and therefore we will require additional testing to determine the best corrections to make.

There are several different accelerometer sensor-in-package (SiP) types available for use. The 9-axis BNO055 discussed earlier has the advantage of the geomagnetic sensors, but is a bit more expensive and an external unit. On the other hand, the 6-axis MPU-6050 unit lacks the magnetometer but is much smaller. Another advantage to this unit is that the BBE computer comes pre-packaged with an MPU-6050 unit on the board, linked to the processor already. There are some differences between the two, outlined below.

|  |  |  |
| --- | --- | --- |
| **Metric** | **MPU-6050** | **BNO055** |
| Size |  |  |
| Weight |  |  |
| Breakout board interfaces | I2C | SPI, I2C |
| Available on-board SBC | Yes | No |
| Sensor package | Triaxis linear accelerometer  Triaxis gyroscope | Triaxis linear accelerometer  Triaxis gyroscope  Triaxis geomagnetic sensor  Sensor fusion |
| Linear accelerometer type | MEMS proof mass |  |
| Gyroscope type | MEMS CVG |  |
| Geomagnetic sensor type | N/A |  |
| Linear accelerometer accuracy |  |  |
| Gyroscope accuracy |  |  |
| Geomagnetic sensor accuracy | N/A |  |
| Linear accelerometer sampling rate |  | 100 Hz |
| Gyroscope sampling rate | 8 kHz | 100 Hz |
| Geomagnetic sensor sampling rate | N/A | 20 Hz |
| Power consumption | 3.9 mA |  |
| Code available for use? | Yes | Yes |

## Telemetry

Having a data transfer link to the rocket is of utmost importance for making informed safety decisions on the ground. Engineers need data about engine performance, vehicle position, velocity, and acceleration; all sensor data should be transmitted to ensure that the vehicle is processing the data correctly in real-time. One key difference to understand is that telemetry is one-way: vehicle-to-ground. Ground-to-vehicle, by definition, are not telemetry data but rather a communications link. Since the only signal that may need to be sent to the rocket is an abort command, the communications link can have a much lower bandwidth than the vehicle-to-ground telemetry link, which must be able to transmit all the necessary data.

Since all data is being recorded on the rocket in case the telemetry link fails, it makes sense to simply insert the telemetry output process on the same hardware responsible for recording the data. This simplifies program design, since the telemetry process can listen to the data channels being recorded and pipe it to the radio transmission process. From there, the data is sent through an amplifier and finally to the antenna. The ground control station picks up the signal, processes and records the data as well as displaying it for the ground controllers to see.

Another potential safety feature that can be implemented easily is running a mirrored version of the vehicle abort determination software on the ground. This allows controllers to see immediately if the vehicle is going to/has initiated an abort sequence, and if so for what reason. It may also allow controllers to see a situation where an abort will be necessary in the near future and make that decision earlier than the automated systems would have determined, further improving safety.

As for the radio hardware required, we can use estimates of atmospheric attenuation by distance to determine the best frequency to use. First, we can narrow the search range down to the 1MHz to 1GHz range with a broad map of atmospheric opacity.

From this point, determining frequencies gets a bit harder. Atmospheric opacity charts for the desired range are very hard to find. Charts for the 1GHz to the 1THz range are common, showing that the limit as frequency goes to 1GHz from the right approaching very low numbers of dBm/km of approximately 0.01. From this point, the best option given the total lack of available data is to perform trials with the appropriate distance.

Since multiple students on our team (at least 2) are pilots, we plan to rent two aircraft from local FBOs (fixed base operators), and flying approximately 100km apart while carrying the appropriate test hardware. Since North Dakota is flat, and we can climb up to several thousand feet, achieving line of sight is entirely feasible. Not only does this allow us to test radio communications at the required distance, but atmospheric attenuation will be *more* than will be experienced by the launch vehicle. This is due to the thickness of the atmosphere – the total amount of air between two aircraft flying at, say, 5,000ft is much greater than the total amount of air between a ground station and a vehicle at 100km altitude. While visual contact will be impossible, aircraft holding at two points that are on the same line of latitude ensure that test antennae can be pointed directly east/west at the other aircraft.

There are still some bounds on what frequencies can be used. To summarize the 177-page FCC frequency band allocation, there are several interesting bands that we may be able to use. Several industrial, scientific, and medical (ISM) bands that are available for anyone to use. ISM bands are at various frequencies allowing us to pick one that suits us best with regards to ease of transmission and minimal atmospheric attenuation.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Frequency range (MHz)** | | **Center (MHz)** | **Type** | **Common uses** |
| 6.765 | 6.795 | 6.78 | A | Fixed & mobile service |
| 13.553 | 13.567 | 13.56 | B | Fixed & mobile service, except automobile |
| 26.957 | 27.283 | 27.12 | B | Fixed, mobile, except automobile & CB |
| 40.66 | 40.7 | 40.68 | B | Fixed, mobile, Earth exploration-satellite |
| 433.05 | 434.79 | 433.92 | Region 1 | Amateur service & radiolocation |
| 902 | 928 | 915 | Region 2 | Fixed, mobile except aeronautical & radiolocation |

**Type A:** Requires authorization by concerning administration (usually FCC)  
**Type B:** Must accept harmful interference in the ISM band

 Left to right: Regions 2, 1, 3

## Recovery

Our main avionics bay will be located near the top of the main body segment. Locating the unit here provides a short link to the sensitive sky-facing GPS antennae while allowing adequate reach to the communications antennae near the rear of the rocket. Running communications links down service tunnels primarily for fuel will allow for a hard-wired communications channel to the engine control unit.

A nominal flight would include burning the engine at full thrust until fuel depletion, then initiating an abort procedure. The abort procedure at any altitude consists of (in order) an immediate depressurization of the fuel tanks, coasting until vertical velocity reaches 0 (apogee), and starting the recovery device deployment loop. The recovery device deployment loop is the program responsible for monitoring the barometric pressure sensor and determining whether to deploy one or both parachutes.

Since the abort procedure is the same as the nominal recovery procedure, we can simply use the abort signal to begin a normal recovery. This further simplifies program design by reducing overall code size and complexity. By implementing a simple velocity check (or even simpler, an altitude-changing-at-minimum-rate check) we can further use the same abort procedure to perform on-the-pad aborts. During an on-the-pad abort, we only want to safely shut down the engine (if it’s been turned on) and depressurize the tanks. Ejecting the nosecone and deploying parachutes would not be productive.

Deciding when to abort based on an avionics unit failure of any type is discussed later, in the risk assessment section. However, it is imperative that the recovery procedure does *not* depend on having a functioning communications link – that is, it can be triggered by an on-board abort signal. To do this, a dedicated program will run on both the vehicle and ground control to factor in available information and determine an automatic abort signal.

Because data such as current GPS position and integrated INS solutions are not available to the vehicle but rather on the ground, there are scenarios possible where an abort can be made by ground controllers before the automated system would have determined it. While it hasn’t been set in stone yet, we are weighing the pros and cons of having a separate SBC to run the abort decision-making algorithm. Since that process could be resource-intensive, having its own dedicated hardware would not only be a safety enhancement but also provide a platform capable of running a more in-depth analysis of all variables and make a more informed, and therefore safer, abort decision.

## Engine Control and Monitoring

The propulsion team is still working with us on finalizing engine design. However, we can still plan on how to control the necessary parts of the engine once we know exactly what those parts are. We plan to use an Arduino Nano to control the necessary servos and pumps, communicating with the brain via a hardwired interface, likely UART.

The engine controller would be responsible for exposing high-level function calls to the applicable interface. These calls would be something like setting engine thrust to a certain value or percentage; initiating or shutting down the engine or reading from sensors. Communication would use a serial protocol with the brain unit sending commands to the Arduino, which would respond appropriately both acknowledging the commands and notifying the brain when the actions have completed.

Safety can be improved by implementing a heartbeat protocol between the two devices. If the engine controller stops responding, it can be assumed that it has failed, and the abort procedure should be initiated. From the engine controller, we can also monitor the brain – if the brain stops responding, then the engine should be shut down. This is another argument for dedicated recovery hardware: if the brain fails, the recovery unit would be aware of that and able to initiate the abort sequence safely.

Some initial calculations show the vehicle exceeding our target altitude by a large margin, potentially endangering other space traffic such as satellites. To deal with this, the brain controller will monitor velocity and altitude to determine if we will overshoot the target altitude by a certain margin. If this margin is exceeded, the engine controller will be sent a shutdown or throttle-down command to prevent this from happening.

Flight Dynamics is still determining if the vehicle will encounter a maximum dynamic pressure (Max-Q) requiring the engine to be throttled down. If so, since there’s no way of measuring dynamic pressure on the vehicle (i.e. no pitot tube), we will have to use timing determine the throttle-down point and duration. To satisfy this requirement, the engine control unit (ECU) must be able to control the pumps and servos with enough accuracy and precision to throttle the engine safely.

# Software

In order to get the best and safest process, custom software is essential. It can be tailored to exactly what the engineering teams want/need, allowing faster and safer work. Custom software will be found everywhere from running on the brain of the vehicle to the ground control telemetry processor. It can be made more efficient than already existing software, which does not do exactly what we need – code that does exactly what it’s designed for is often better that code that can do many things including what we need.

Our software can be broken down into 4 main parts: telemetry, engine control, abort decision-making, and recovery. Each of the 4 processes will have their own hardware in one form or another and are described below. Since each of these elements are described in more detail above (per the Base 11 PDR template), only a brief summary of responsibilities & requirements is given below.

* Telemetry: Responsible for sending data back to the ground control station. It must be able to process all the available sensor data onboard the vehicle and format it in a manner that takes up the minimum bandwidth for transmission. We aren’t yet sure how much data there will be, but packetization and compression are being investigated to weigh the transmission efficiency versus the additional computing power required.
* Full-authority digital engine control (FADEC): Responsible for processing the requests from the brain controller and turning those requests into reality, the FADEC software will be required to run in real-time. Our first choice for hardware is the Arduino Nano, which provides a small, lightweight, and familiar environment meeting the real-time requirements.
* Recovery: As discussed earlier, the recovery process will have dedicated hardware to facilitate safe recovery of the vehicle. Primarily altimeter-based and single-threaded, the recovery software must also be run in real-time, making the Arduino Nano again a good option.
* Abort decision-making: There are multiple reasons to have an on-board abort decision. For example, if the telemetry link fails but FADEC and recovery remain, there’s no reason to abort the flight prematurely. A counterexample where abort would be necessary could be if the vehicle begins to rotate about the wrong axis – vehicle destruction could happen very quickly and therefore ground controllers may not have time to send an abort command back. An on-board abort controller could process the rotational acceleration real-time and initiate an abort much faster than a ground-based controller could.

# Ground Control

Ground control software is where most of the calculations will take place. The vehicle itself only needs to know altitude, orientation and engine parameters to facilitate a safe abort in the event of a communications link failure, so the rest of the calculations can be performed by much higher-performance computers on the ground. These on-site calculations will provide a live data feed for the launch team to monitor and make a decision about whether they want to abort the flight or not.

This also allows the vehicle avionics to offload unnecessary stress. Our ground control unit (GCU) will consist of a high-powered desktop computer, necessary radio receivers and antennae, a cellular phone modem (unless somehow Internet access is provided via a different method), and a suitably stocked snack drawer. The software running on the computer will be custom-built and designed to give the engineers all available data and system statuses in an easy-to-understand manner.

Current thoughts are that the most efficient system is to set up everything we need in a gutted van. This allows the setup to be mobile, tested at home and known to work anywhere. It can be used for multiple launches, including smaller outreach launches for hardware/software testing, and provides an easy method to transport any necessary electronics.

Certain functionality can be enhanced by having umbilical cables running to the rocket. These cables, although we aren’t yet certain what type of cable/interface to use (Ethernet, serial, a custom interface) would allow a direct line to the brain of the rocket pre-launch and enable safer on-the-pad aborts, launch control, and a much faster telemetry line for engine parameters. It also provides a much easier method of starting the avionics processes before the radio link is established without having someone physically at the vehicle.

## Tracking & Receiving

The vehicle’s omnidirectional antenna won’t be transmitting with enough power to be received by an isotropic antenna on the ground. Therefore, a more directional high-gain antenna such as a Yagi-Uda antenna is required. To keep the antenna oriented correctly, a “turret” will be built. This turret would include a rotational servo, for controlling azimuth, and another servo for controlling elevation. Between these two servos the turret can be pointed at any point in the sky.

Mounted on the turret will be a high-resolution, high-zoom video camera for a visual feed of the rocket. This provides a very simple manner for visual target tracking.

Target tracking will be accomplished with a differential radio antenna system. Mounting 4 directional Yagi-Uda antennas in a diamond pattern, slightly angled away from the center, will give the capability to track the telemetry signal from the vehicle. Comparing the receive volume from the two left-right antennas yields the direction to rotate in, and comparing the volume from the two vertically-aligned antennas gives the direction to elevate in.

Since radio processing is a complex task requiring sufficient hardware, each of the radio receivers will be connected to a dedicated controller responsible for determining the appropriate action for the turret to take. Many such radio receivers are designed to output to a USB port, so something with a 4-port USB bus is required. Ideally, the device would also have a multi-core processor to allow for one process to determine necessary actions and another to order their execution.

Some smoothing will be required to keep the unit from oscillating violently based on small signal variations. It will likely be easier to keep the target slightly above the vertical center, so that the entire system continually and smoothly increases in elevation rather than attempting to constantly focus on the vehicle and then immediately falling behind. The optical camera can be angled up slightly to compensate for this, but more experimenting will be required to determine optimal setup and smoothing.

While implementation can be difficult, mounting a compact radar on the turret will also give distance data. Interestingly, with distance, azimuth, and elevation, a 3D geometric position and altitude can be computed. This provides another redundant location mechanism should the primary methods fail on the vehicle. The ground control units will be able to provide sufficient computing power and speed to process the translation from raw data (elevation, azimuth, range) into a position (latitude, longitude, altitude). If it is found that a single unit is not able to compute the transformation on top of other tasks, another smaller unit, possibly an RPi 3B+, can be added as a middle-man responsible for the transformation of turret data into a location. As long as the Pi is not responsible for anything else, the available power should be plenty to run such a transformation.