Talos: A Beginner's Exploration of Avionics and Telemetry for Model Rockets

Nathan D. Alspaugh, Samuel J. Correa

December 29, 2024

Abstract

In any aircraft or rocket, there are two systems present that are essential to any flight, the avionics and telemetry systems. The avionics system is responsible for controlling the flight path, monitoring the rocket's orientation, and collecting data. The telemetry system is responsible for transmitting this data to the ground station. In this paper, we present Talos, a project that aims to provide a beginner's perspective on the design and implementation of avionics and telemetry systems for model rockets. We document the process of learning to design and build Talos, from the initial planning phase to the final implementation and testing. We discuss the challenges faced, the lessons learned, and the future directions of the project. We hope that this paper will serve as a useful resource for students and hobbyists interested in exploring the field of avionics and telemetry for model rockets. Our results show that it is possible to design and build a functional avionics and telemetry system for model rockets using specialized components and open-source software that are accessible by any high school student. We believe that Talos has the potential to inspire others to explore the exciting world of aerospace engineering and rocketry.

1 Introduction

The avionics and telemetry systems are critical components of modern rocketry as they enable precise navigation, monitoring, and allow for the collection of data to later undergo optimizations. Being dedicated students, passionate for aerospace and mechanical engineering, we decided to explore these topics by building an avionics and telemetry system for model rockets. Talos, named after the mythical giant bronze automaton, is a project that aims to provide a beginner's perspective on the design and implementation of such systems. This paper documents the process of learning to design and build Talos, from the initial planning phase to the final implementation and testing.

2 Background Research

The term Avionics refers to the electronic systems used in aircraft and spacecraft. Avionics systems are responsible for navigation, communication, flight control, and monitoring. In the context of model rockets, avionics systems are used to control the flight path, collect data, and through telemetry, transmit to the ground station. This is made possible using a combination of Flight Control Systems, Air Data Systems, and Inertial Sensor Systems. Telemetry systems are used to transmit data from the rocket to the ground station using wireless communication protocols.

The Flight Control System employed in Talos is based on the auto stabilization (or stability augmentation) system described in [2]. This system uses a combination of sensors to measure the rocket's orientation and adjust the control surfaces to maintain a desired flight path.

The Air Data System is responsible for measuring the rocket's altitude and airspeed. The system should compute these values from a combination of pressure and temperature sensors.

The Inertial Sensor System is used to measure the rocket's orientation and acceleration. This is achieved using sensors that can measure rotation (gyroscopes), sensors that can measure acceleration (accelerometers). In conjunction with the Air Data System, the INS can be used to compute the rocket's velocity vector information.

For the Telmetry System, we decided to use the LoRa (Long Range) protocol. LoRa is a long-range, low-power wireless communication protocol that is ideal for transmitting telemetry data from the rocket

to the ground station. The LoRa protocol is based on spread spectrum modulation techniques, which increases reliability by "spreading" the signal over more frequencies, and is capable of transmitting data over long distances (up to 10 km) with low power consumption. This makes it ideal for use in model rockets, where power consumption and range are critical factors. It is also very immune to the doppler effects[3] which is important for rockets that are moving at high speeds.

3 Methodology

The Talos project was divided into three main phases: Planning, Design, and Implementation. Each phase involved a series of tasks and challenges that needed to be addressed.

3.1 Planning

The planning phase involved defining the project scope, setting goals, and identifying the components needed. We started by researching existing avionics systems and telemetry solutions to understand the requirements and challenges involved. We then defined the key features of Talos, such as flight control, telemetry, and data logging. We also identified the components needed, such as sensors, microcontrollers, and communication modules.

3.1.1 Breakout Board or Custom Printed Circuit Board (PCB)?

After thorough research, we decided to create our own custom PCB for the Talos project. This was heavily impacted by the space constraints of the rocket, as well as the need for a powerful microcontroller with custom sensors that normal breakout boards (Arduino or ESP32) do not have. When we started the project, we had no experience with PCB design so we had to learn how to use EasyEDA, a free online PCB design tool.

3.1.2 Microcontroller Selection

We decided to use the STM32F405 microcontroller for the Talos project. This microcontroller has a powerful ARM Cortex-M4 core (180 mHz), plenty of I/O pins, and built-in support for various communication protocols. We chose this microcontroller because of its speed and versatility (see figure 1), as well as the availability of development tools and libraries. The STM32 Family has also been used in other aerospace applications, such as the CubeSat project[8].

3.1.3 Sensor Selection

Inertial Measurement Unit (IMU)

We decided to utilize the LSM6DM IMU sensor for the Talos project. This sensor combines a 3-axis accelerometer and a 3-axis gyroscope in a single package. The LSM6DM is a MEMS (Microelectromechanical Sensor) capable of measuring acceleration and angular velocity in all three axes (see figure 2), making it ideal for measuring the rocket's orientation and acceleration. It also includes a thermometer to measure the temperature inside the rocket.

Pressure Sensor

For the pressure sensor, we decided to use the BMP580 (see figure 3). This sensor is capable of measuring pressure and temperature, which can be used to calculate the rocket's altitude and airspeed. It is also very accurate, more so than the more commonly used BMP280.[4]

3.1.4 Communication Module

For the telemetry system, we decided to use the LLCC68 LoRa module and more specifically the DL-LLCC68-S-433 package. We use this package specifically because it transmits at 433 MHz, whose frequency is used for long range communication.

Figure 1: Table of STM32 Microcontrollers[7]

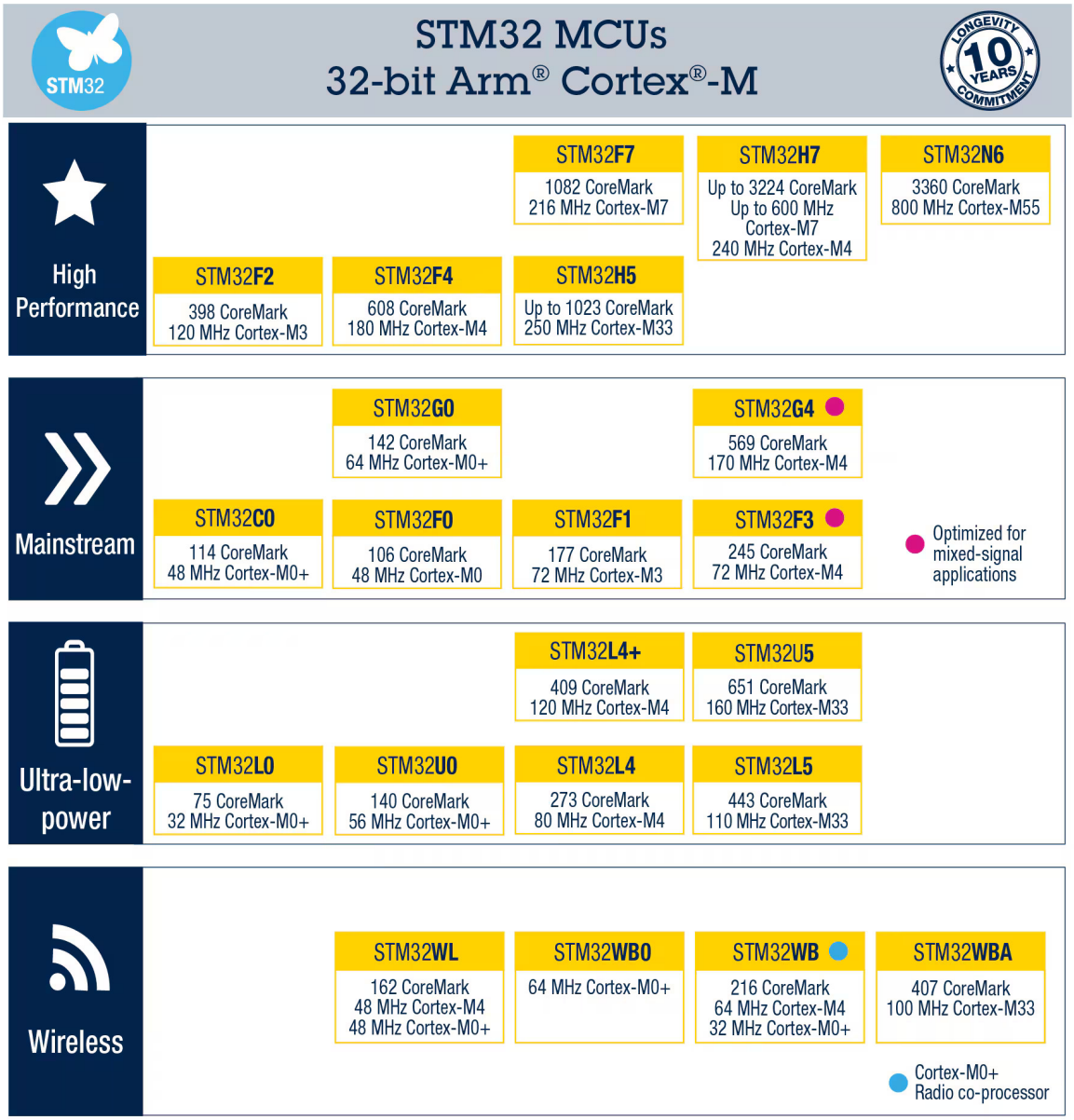


Figure 2: LSM6DM IMU Sensor[5]

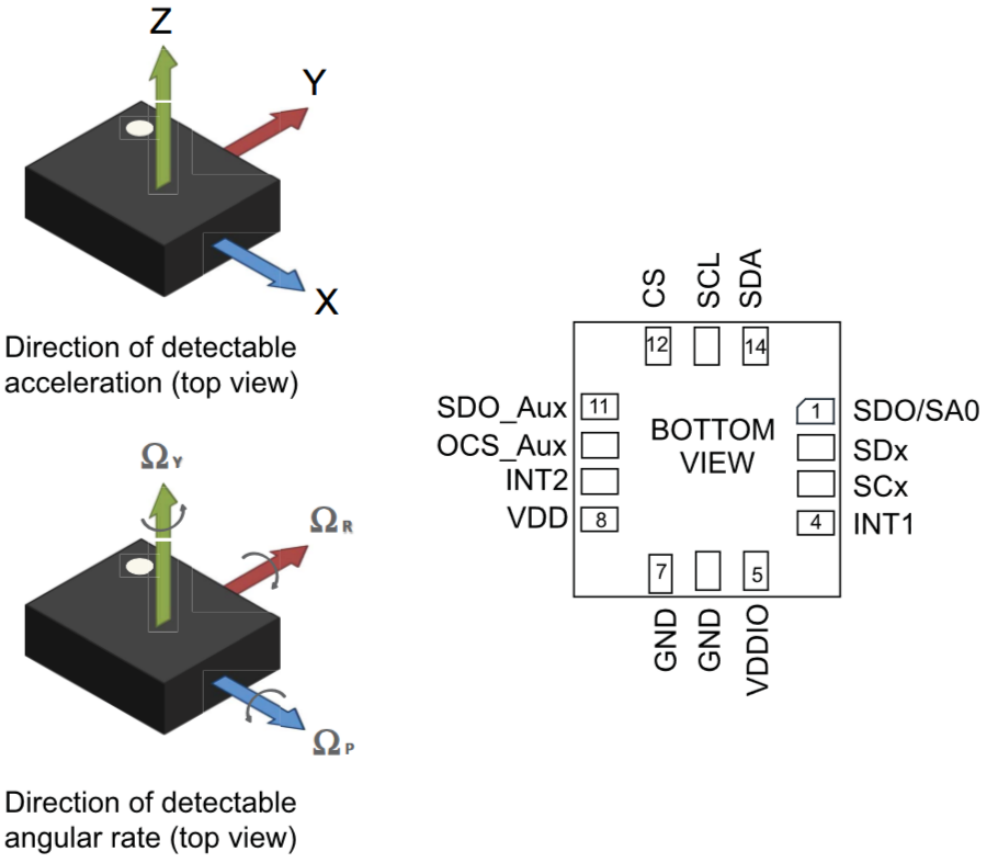
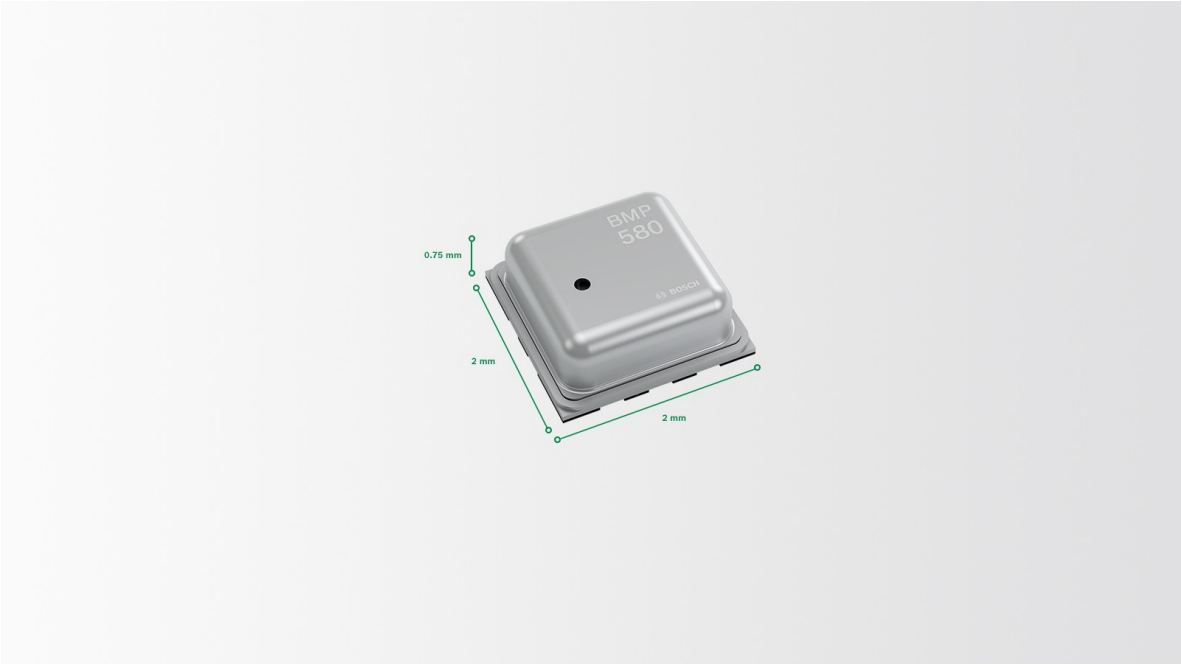


Figure 3: BMP580 Pressure Sensor[1]



3.2 Design

After meticulously choosing parts, we decided to embark on the design phase. This phase involved creating the schematics and PCB layout for Talos. We started by designing the schematics in EasyEDA, which involved connecting the microcontroller, sensors, and communication modules. This was a huge challenge as we did not know how to connect each of the sensors. We had to learn how to read datasheets and understand the electrical characteristics of each component. Taking into account figure 2, we knew the basics of electronics, GND means ground, VDD means power, but the other pins were a complete mystery. Here are some terms that we had to learn:

- Resistor: A resistor is a passive two-terminal electrical component that implements electrical resistance as a circuit element.
- Capacitor: A capacitor is a passive two-terminal electrical component that stores electrical energy in an electric field.
- Serial Peripheral Interface (SPI), a way to communicate between devices along a single data line. Imagine it like a bus, where each device is the stop and the data is the passengers. We select the stop using the Chip Select (CS) line and then send or recieve the date through the MOSI and MISO lines (SDO and SDA). The clock is used to synchronize the data and make sure that the bus arrives at the same time that the passengers are ready to board.

  SDO: Serial Data Out

  SDA: Serial Data In

  SCL: Serial Clock

  CS: Chip Select
- Inter-Integrated Circuit (I2C), a way to communicate between devices along a single data line. Using the same analogy as above with the bus but this time the driver doesn't know the directions of the stops and has to ask the passengers where to go. The SDA line is used to send and recieve data, while the SCL line is used to synchronize the data.

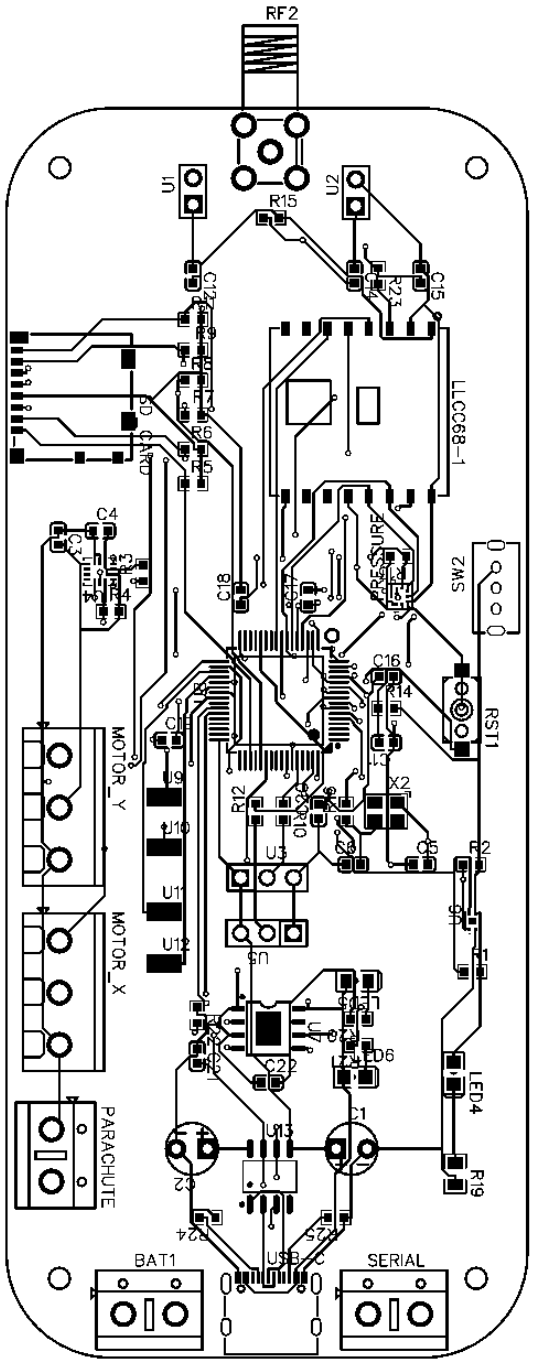
  SDA: Serial Data

  SCL: Serial Clock
- Interrupts, a way to notify the microcontroller that an event has occurred. This is like a bell that rings when the bus arrives at the stop.
- Universal Asynchronous Receiver-Transmitter (UART), a way to communicate between devices using two data lines. This is like a walkie-talkie, where one person talks and the other listens. The TX line is used to send data, while the RX line is used to recieve data.
- Pull-up and Pull-down resistors, resistors that are used to set the default state of a pin. Pull-up resistors are used to set the default state of a pin to high, while pull-down resistors are used to set the default state of a pin to low.
- Pull-up and Pull-down capacitors, capacitors that are used to filter out noise in a circuit. Pull-up capacitors are used to filter out noise in a high state, while pull-down capacitors are used to filter out noise in a low state.
- Decoupling capacitors, capacitors that are used to filter out noise in a circuit. These capacitors are used to filter out noise in the power supply.
- Cross-talk, a phenomenon where signals from one circuit interfere with signals from another circuit. This can cause errors in the data being transmitted.

Finally after reading all the datasheets for each component we created the schematic for the Talos project. The microcontroler schematic is shown in figure 5.

and the I/O schematic is shown in figure 6.

Figure 4: Talos PCB Layout



3.2.1 PCB Layout

After designing the schematics, we moved on to the PCB layout. This involved placing the components on the board and routing the traces to connect them. We had to consider factors such as signal integrity, power distribution, and thermal management. We also had to ensure that the board was compact enough to fit inside the rocket. This was a challenging task as we had to learn how to use the PCB design software and understand the constraints of the board. We had to learn how to route traces, place components, and not make any mistakes (which we did). Here is the final PCB layout for the Talos project (see figure 4).

We decided to put the microcontroller in the center of the board, with the sensors and commu-

nication modules around it. This was done to minimize the length of the traces and reduce the risk of interference. We also added decoupling capacitors near the power supply pins of each component to filter out noise. We also added pull-up and pull-down resistors to set the default state of the pins. Finally, we added a USB-C port for programming and debugging the board. Along the sides we took out pins to control tha parachute and vector thrusting system. We also added a battery connector to power the board with a 3.3V battery. Just in case, we exposed the serial lins so we could program the board in the event that the USB-C port failed.

3.3 Implementation

After designing the PCB, we moved on to the implementation phase. This phase consisted of ordering the PCB through JLCPCB and later programming the board. Although we talked with various professors before sending the board to be made, we still made small but critical errors. When we first recived the board, we encountered the first errors which was that firstly, the microcontroller would not turn on, and secondly, the serial lines were switchd up. Looking at the schematic in figure 6, in the USB/Serial section ther is a chip with the label CH340N. This chip is responsible for the USB to Serial convrsion. Part of that serial conversion is to swap the lines RXD and TXD because the output of the USB is the input to the Microcontroller. After reading the datasheet more carefully we discovered that we crossed the lines on the CH340N chip as well as the mictrocontroller, which resulted in the USB-C port not functioning correctly. We also missed two decoupling capacitors that were essential for a correct boot-up of the chip. After soldering the capacitors and crossing the RXD/TXD pins, the board booted up correctly.

3.3.1 Programming

We then proceeded to program the board using the STM32CubeIDE, a free IDE provided by STMicroelectronics[6]. We had to learn how to use the IDE, how to program the microcontroller, and how to interface with the sensors al in the C programming language. This was acheived throught the use of open source drivers that we ported to the STM32 MCU. After configuring all the drivers for the various sensors, we discovered another error. In the BMP540 we forgot two more decoupling capacitors that impeded the sensor from starting up. We are currently working on a resolution to this problem as the soldering technique required to fix the issue is very precise.

3.3.2 Testing

To tst the board, we designed and created an interactive web page that would display the data from the sensors in real time. We used the serial port to send the data through websockets to a web server written in Rust. We chose Rust because of its speed and low memory usage which is essential in any real-time application. We then used the websockets to send the data to the web page. The web page was written in React, Typescript, and TailwindCSS. We used React for the front-end because of its ease of use and the ability to create interactive web pages. We used Typescript for the back-end because of its type safety and the ability to catch errors at compile time. We used TailwindCSS for the styling because of its utility-first approach and the ability to create responsive designs. The web page displayed the data from the sensors in real time, including the rocket's orientation, altitude, airspeed, and temperature. The website is shown in figures 7 and 8 , and it can be accessed at `https://talos.notaroomba.dev`.

4 Results

After starting the project in August of 2024, we were surprised at how much we learned in such a short time. We learned how to design a PCB, how to program a microcontroller, and how to interface with sensors. We also learned how to use various tools and libraries, such as EasyEDA, STM32CubeIDE, Rust, C, and WebSockets. We also learned how to work as a team, how to communicate effectively, and how to solve problems together. We also learned how to manage our time, how to set goals, and how to achieve them. We also learned how to document our progress, how to write reports, and how to present our work. We also learned how to learn, how to research, and how to teach ourselves new things. We also learned how to fail, how to make mistakes, and how to learn from them. We also

learned how to succeed, how to achieve our goals, and how to be proud of our work. We also learned how to inspire others, how to share our knowledge, and how to make a difference. We also learned how to be curious, how to ask questions, and how to seek answers. We learned how to engineer.

5 Future Work

In the future, we plan to continue working on the Talos project and improve the avionics and telemetry systems. We plan to add more sensors, such as a GPS module, a magnetometer, and a barometer. We also plan on re-doing the layout of the PCB and place the IMU sensor in the center of the board. The sensors we chose could also be improved as new versions come out. For anyone who also wants to towk on this project, we recommend that you have a basic understanding of electronics, programming, and engineering design principles. Don't forget to double check your work and **ALWAYS** read through the datasheets.

References

[1] Bosch Sensortec. *Bosch Sensortec*. [Online; accessed December 29, 2024]. 2023. URL: https://www.bosch-sensortec.com/media/boschsensortec/products/environmental_sensors/pressure_sensors/bmp580/bosch-sensortec-bmp580-stage_res_1600x900.jpg.

[2] R.P.G. Collinson. *Introduction to avionics*. Springer Nature : Springer, 2012, pp. 1–2.

[3] Alexander A. Doroshkin et al. "Experimental Study of LoRa Modulation Immunity to Doppler Effect in CubeSat Radio Communications". In: *IEEE Access* 7 (2019), pp. 75721–75731. DOI: 10.1109/ACCESS.2019.2919274.

[4] Bosch Sensortec. *Barometric pressure sensors*. 2024. URL: https://www.bosch-sensortec.com/products/environmental-sensors/pressure-sensors.

[5] STMicroelectronics. *iNEMO inertial module: always-on 3D accelerometer and 3D gyroscope*. [Online; accessed December 29, 2024]. 2017. URL: https://www.st.com/resource/en/datasheet/lsm6dsm.pdf.

[6] STMicroelectronics. *Integrated Development Environment for STM32*. 2019. URL: https://www.st.com/en/development-tools/stm32cubeide.html.

[7] STMicroelectronics. *STM32 MCUs 32-bit Arm Cortex-M*. [Online; accessed December 29, 2024]. 2024. URL: https://www.st.com/content/dam/category-pages/stm32-32-bit-arm-cortex-mcus/arm_cortex_mcu_portfolio_new.jpg.

[8] Bruce Yost. *State-of-the-Art: Small Spacecraft Technology*. Ed. by Sasha Weston. 2023, pp. 214–217.

Figure 5: Talos Microcontroller Schematic

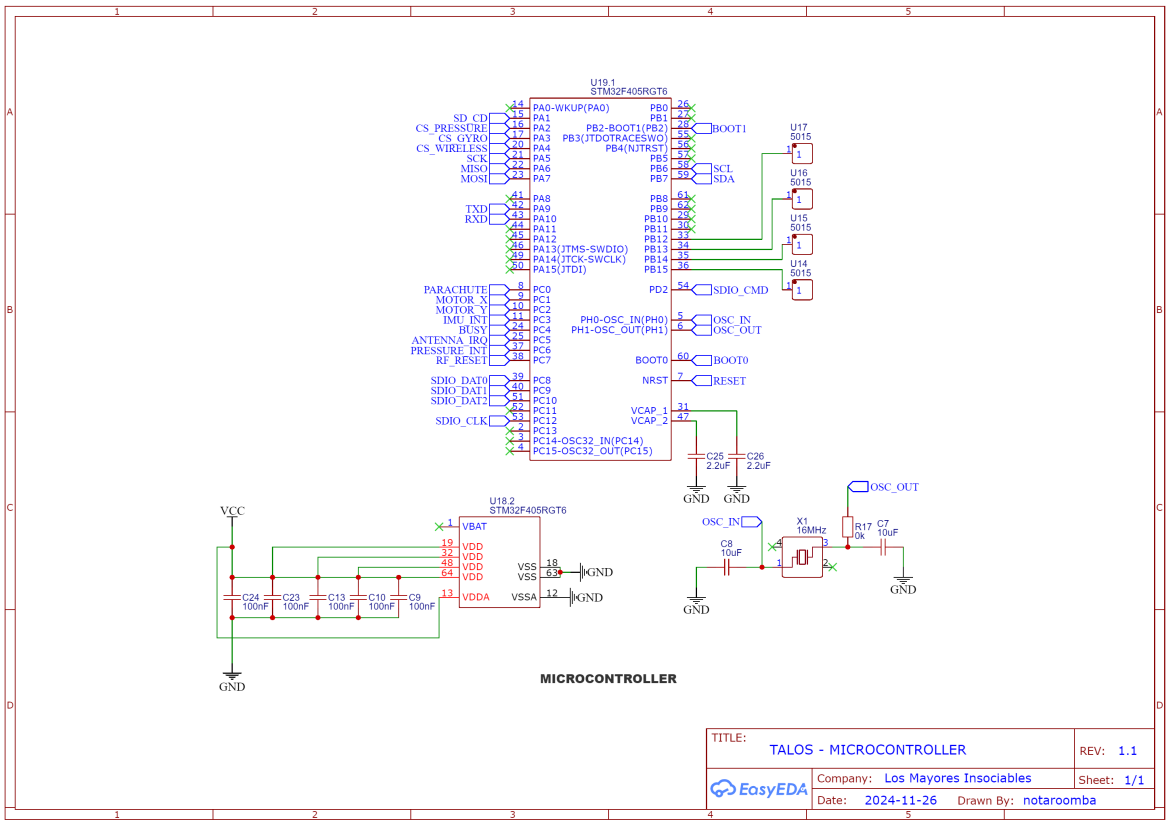


Figure 6: Talos I/O Schematic

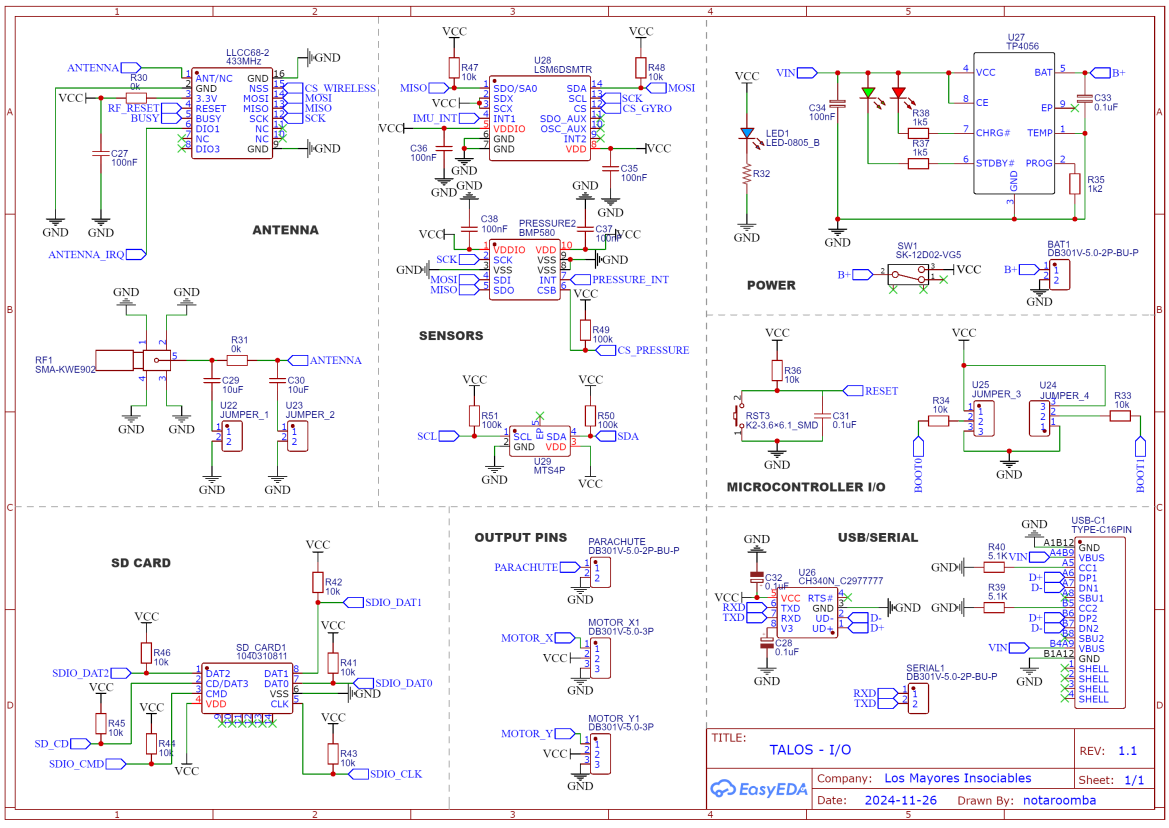


Figure 7: Website with no data

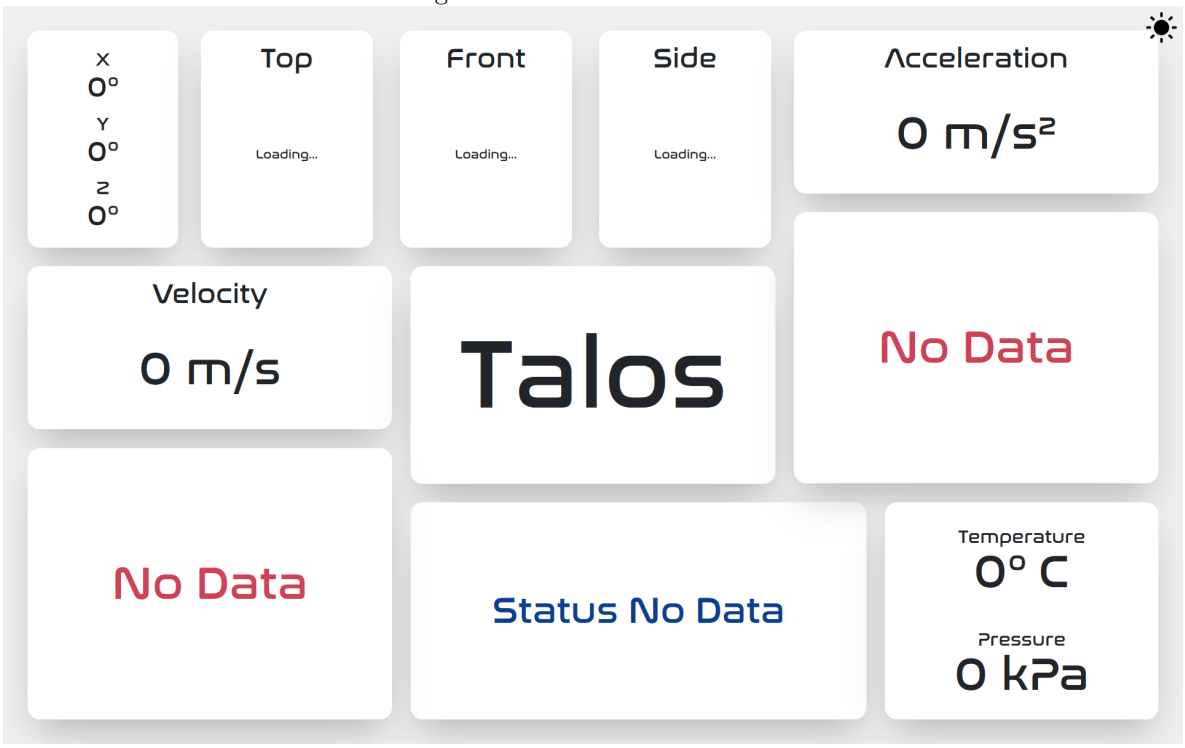


Figure 8: Website with data