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, two systems are present that are essential to any flight: the avionics and telemetry systems. The avionics system controls the flight path, monitors the rocket's orientation, and collects 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 designing and implementing avionics and telemetry systems for model rockets. We document the learning process to design and build Talos, from the initial planning to implementation and testing. We discuss the challenges faced, the lessons learned, and the project's future directions. We hope this paper will serve as a valuable 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 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 and monitoring and allow data collection to undergo optimizations later. As dedicated students passionate about 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 learning to design and build Talos, from the initial planning phase to the final implementation and testing.

2 Background Research

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

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

The Air Data System measures the rocket's altitude and airspeed. It should compute these values using a combination of pressure and temperature sensors.

The Inertial Sensor System measures the rocket's orientation and acceleration. It uses sensors that can measure rotation (gyroscopes) and acceleration (accelerometers). In conjunction with the Air Data System, the INS can compute the rocket's velocity vector information.

We decided to use the LoRa (Long Range) protocol for the Telemetry System. LoRa is a long-range, low-power wireless communication protocol ideal for transmitting telemetry data from the rocket to the ground station. The LoRa protocol, based on spread spectrum modulation techniques, increases

reliability by "spreading" the signal over more frequencies and can transmit data over long distances (up to 10 km) with low power consumption. These factors make LoRa ideal for model rockets, where power consumption and range are critical factors. It is also very immune to the Doppler effect[3], which is important for rockets moving at high speeds.

3 Methodology

We divided the Talos project into planning, design, and implementation phases. Each phase involved a series of tasks and challenges that must be solved.

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. 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 designed our custom PCB for the Talos project. Heavily impacted by the rocket's space constraints and the need for a powerful microcontroller with custom sensors, typical breakout boards (Arduino or ESP32) were insufficient for our use. 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) and the availability of development tools and libraries. Particular CubeSat projects have also been using the STM32 Family[8], which affected our decision in selecting the microcontroller.

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 can measure pressure and temperature, which we use to calculate the rocket's altitude and airspeed. It is also more accurate than the more commonly used BMP280.[4]

3.1.4 Communication Module

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

Figure 1: Table of STM32 Microcontrollers[7]

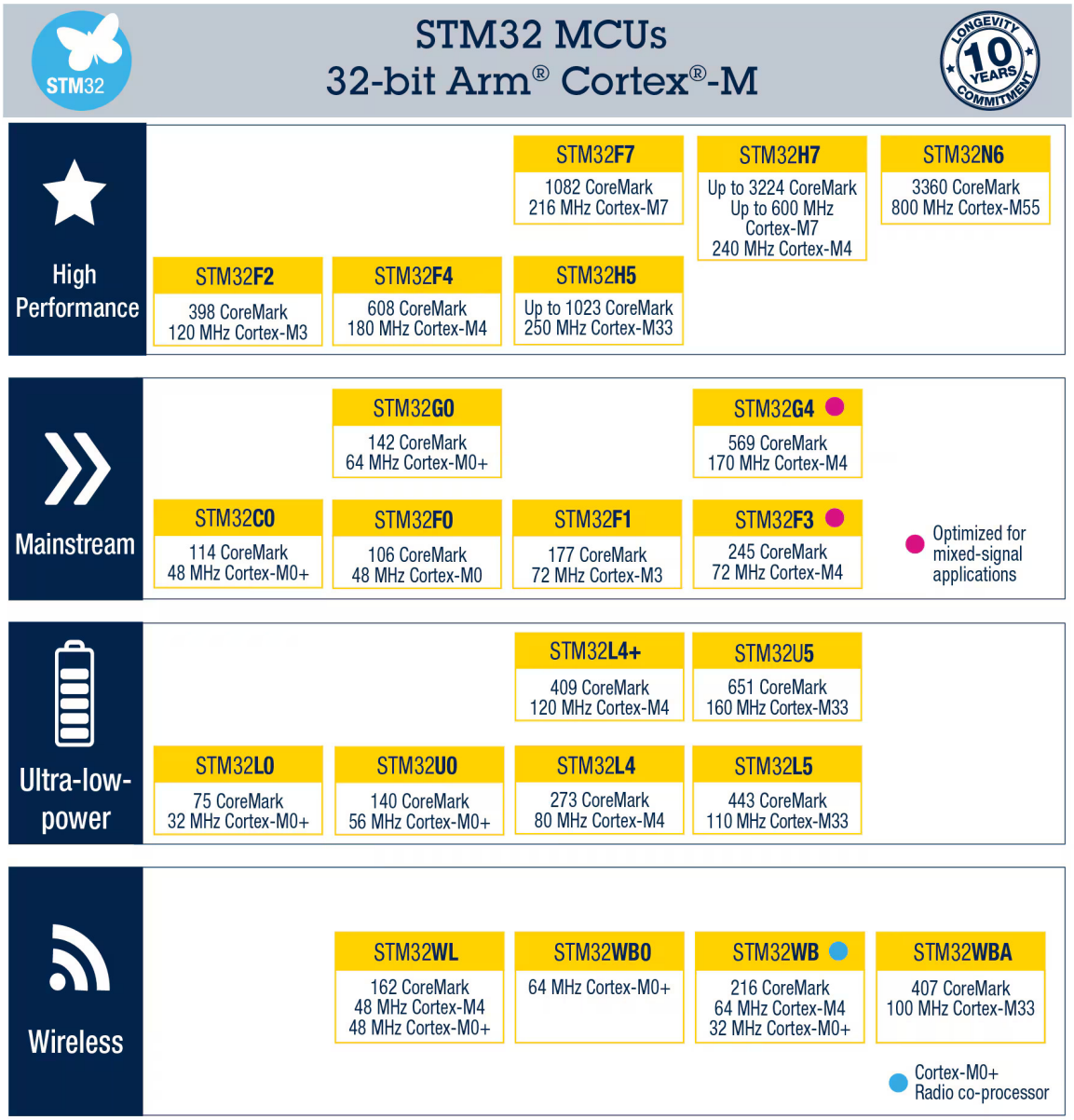


Figure 2: LSM6DM IMU Sensor[5]

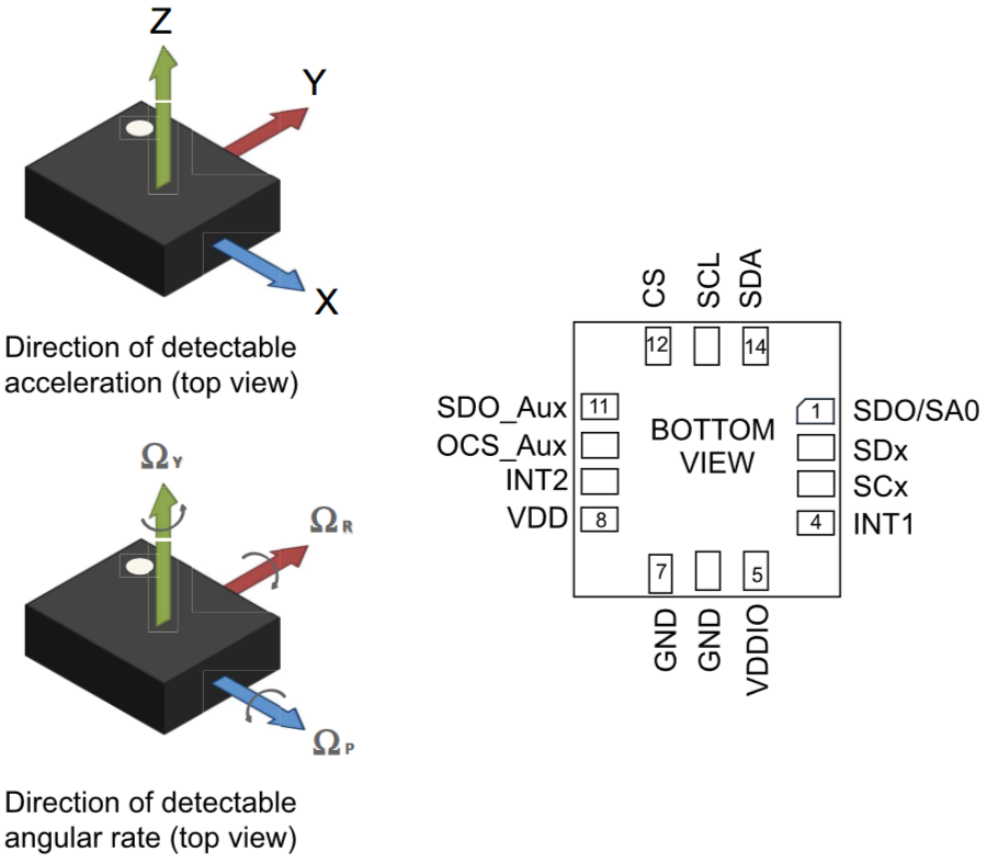
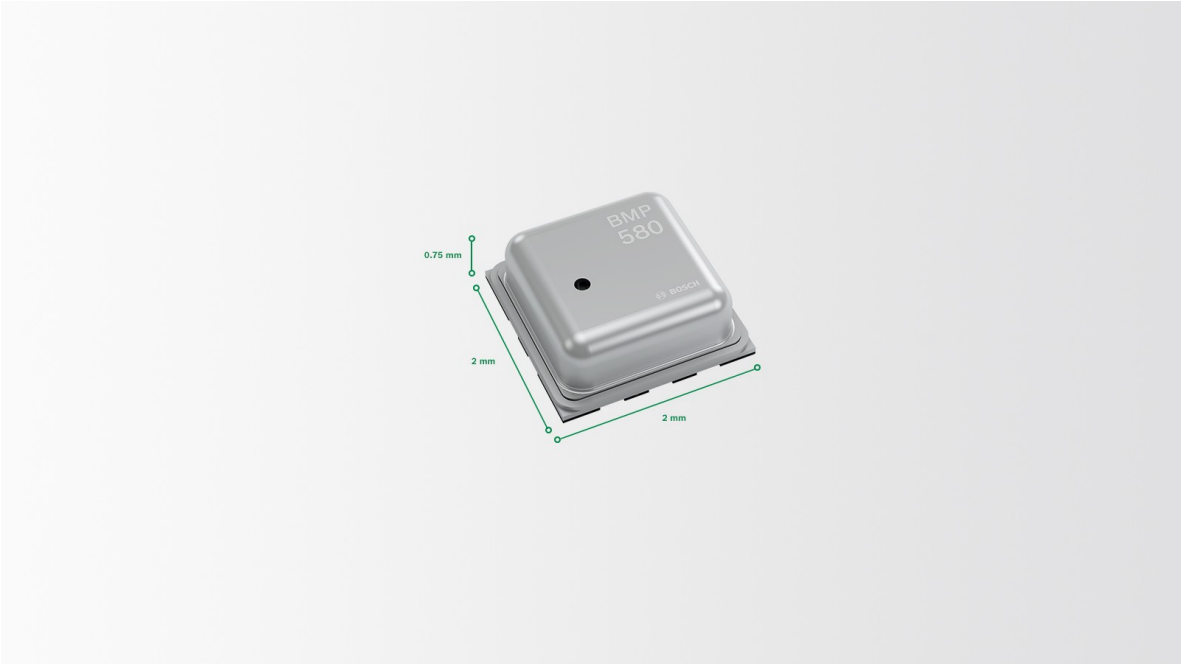


Figure 3: BMP580 Pressure Sensor[1]



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. Creating the schematics was a massive challenge as we did not know how or where to connect each sensor. 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, and 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 receive the date through the MOSI and MISO lines (SDO and SDA). The clock synchronizes the data and ensures the bus arrives when the passengers are ready to board.

  SDO: Serial Data Out

  SDA: Serial Data In

  SCL: Serial Clock

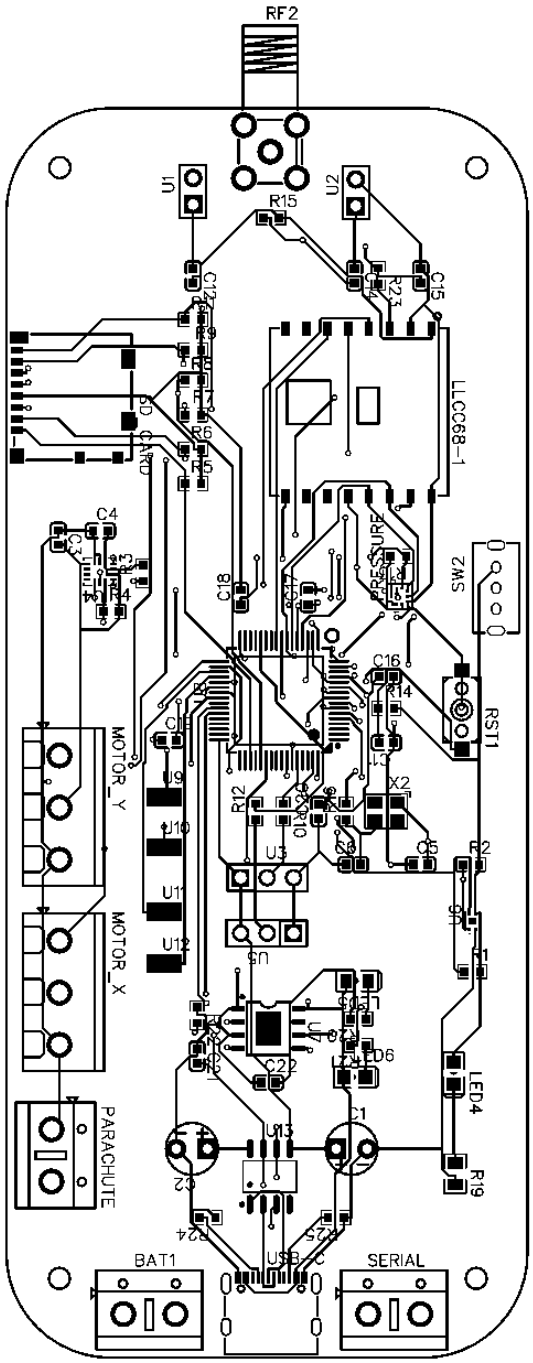
  CS: Chip Select
- Inter-Integrated Circuit (I2C): a way to communicate between devices along a single data line. Using the same analogy as above with the bus, the driver does not know the directions of the stops and has to ask the passengers where to go. The SDA line sends and receives data, while the SCL line synchronizes the data.

  SDA: Serial Data

  SCL: Serial Clock
- Interrupts: a way to notify the microcontroller that an event has occurred. These interrupts are 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. UART is like a walkie-talkie, where one person talks and the other listens. The TX line sends data, while the RX line receives data.
- Pull-up and Pull-down resistors: resistors that set the default state of a pin. Pull-up resistors set the default state of a pin to high, while pull-down resistors set the default state of a pin to low.
- Pull-up and Pull-down capacitors: capacitors that filter out noise in a circuit. Pull-up capacitors filter out noise in a high state, while pull-down capacitors filter out noise in a low state.
- Decoupling capacitors: capacitors that filter out noise in a circuit. These capacitors filter out noise in the power supply.
- Cross-talk: a phenomenon where signals from one circuit interfere with those from another. This cross-talk can cause errors in the data transmission.

Finally, after reading all the datasheets for each component, we created the schematic for the Talos project. Figure 5 shows the microcontroller schematic, while the I/O schematic is at figure 6.

Figure 4: Talos PCB Layout



3.2.1 PCB Layout

After designing the schematics, we moved on to the PCB layout. The layout involved placing the components on the board and routing the traces to connect them. We had to consider signal integrity, power distribution, and thermal management factors. We also had to ensure that the board was compact enough to fit inside the rocket. The design 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 placement minimizes the length of the traces and reduces 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 the parachute and vector thrusting system. We also added a battery connector to power the board with a 3.3V battery. We exposed the serial lines to program the board if 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 received the board, we encountered the following errors: firstly, the microcontroller would not turn on, and secondly, we accidentally double-switched the serial lines. In the USB/Serial section in the schematic in figure 6, there is a chip with the label CH340N. This chip is responsible for the USB to Serial conversion. Part of that serial conversion is swapping the lines RXD and TXD because the USB output is the microcontroller's input. After reading the datasheet more carefully, we discovered that we crossed the lines on the CH340N chip as on the microcontroller, which resulted in the faulty USB-C port. We also missed two decoupling capacitors that were essential for a correct boot-up of the chip. The board booted up correctly after soldering the capacitors and crossing the RXD/TXD pins.

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, program the microcontroller, and interface with the sensors in the C programming language. We achieved this through open-source drivers that we ported to the STM32 MCU. We discovered another error after configuring all the drivers for the various sensors. In the BMP580, we forgot two more decoupling capacitors, which impeded the sensor from starting up. We are currently working on resolving this problem, as the soldering technique required to fix the issue is exact.

3.3.2 Testing

To test 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 web sockets to a web server written in Rust. We chose Rust because of its speed and low memory usage, which are essential in any real-time application. We then used the WebSockets to send the data to the web page. We wrote the web page 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 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 quickly. We learned how to design a PCB, program a microcontroller, and interface with sensors. We also learned to use tools and libraries like EasyEDA, STM32CubeIDE, Rust, C, and WebSockets. As it was our first time working on a project of this magnitude, we encountered many challenges. We had to learn how to read datasheets, troubleshoot errors, and work as a team. We are proud of our accomplishments and are excited to continue working on the project. Talos has the potential to inspire others to explore the exciting world of aerospace engineering and rocketry.

5 Future Work

We plan to continue working on the Talos project and improve the avionics and telemetry systems. We plan to add more sensors and components, such as a GPS module, a magnetometer, and a camera. We also plan on re-doing the PCB layout and placing the IMU sensor in the center of the board. We will also upgrade the sensors as new versions come out. Anyone who also wants to work on this project needs a basic understanding of electronics, programming, and engineering design principles. Do not forget to double-check the design 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

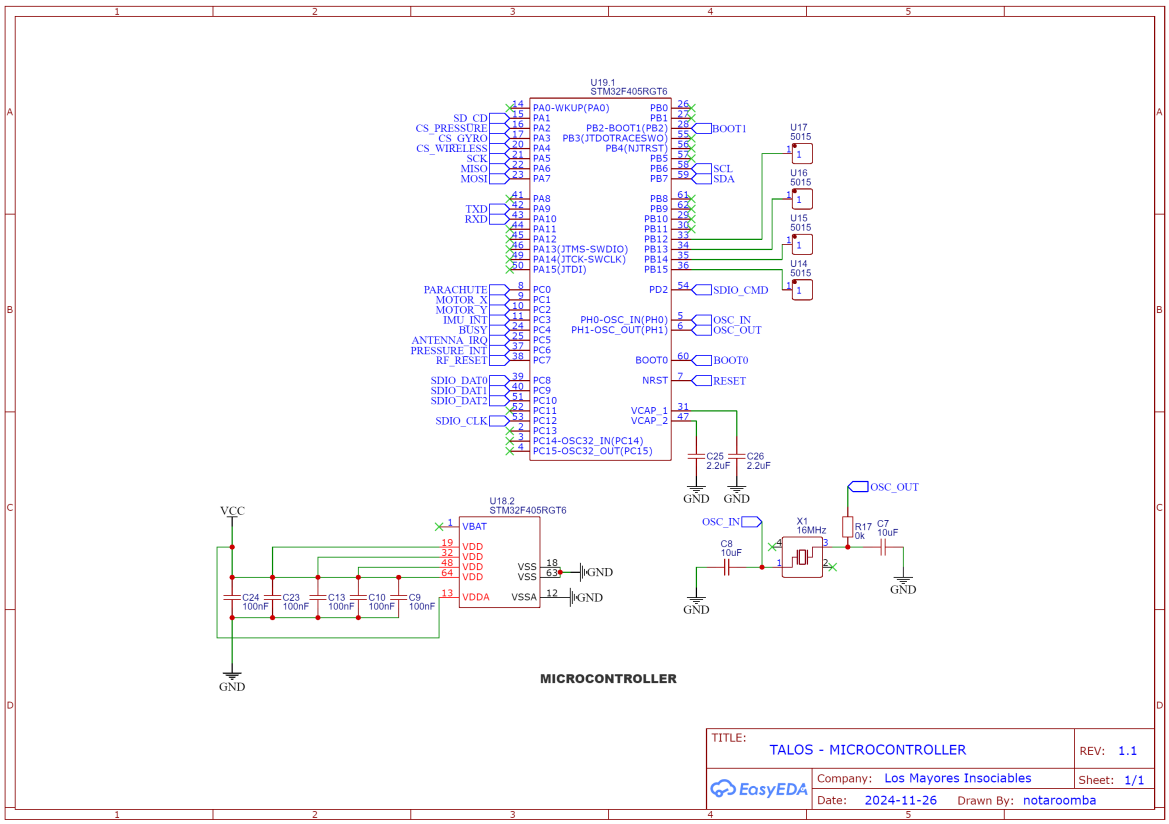


Figure 6: Talos I/O Schematic

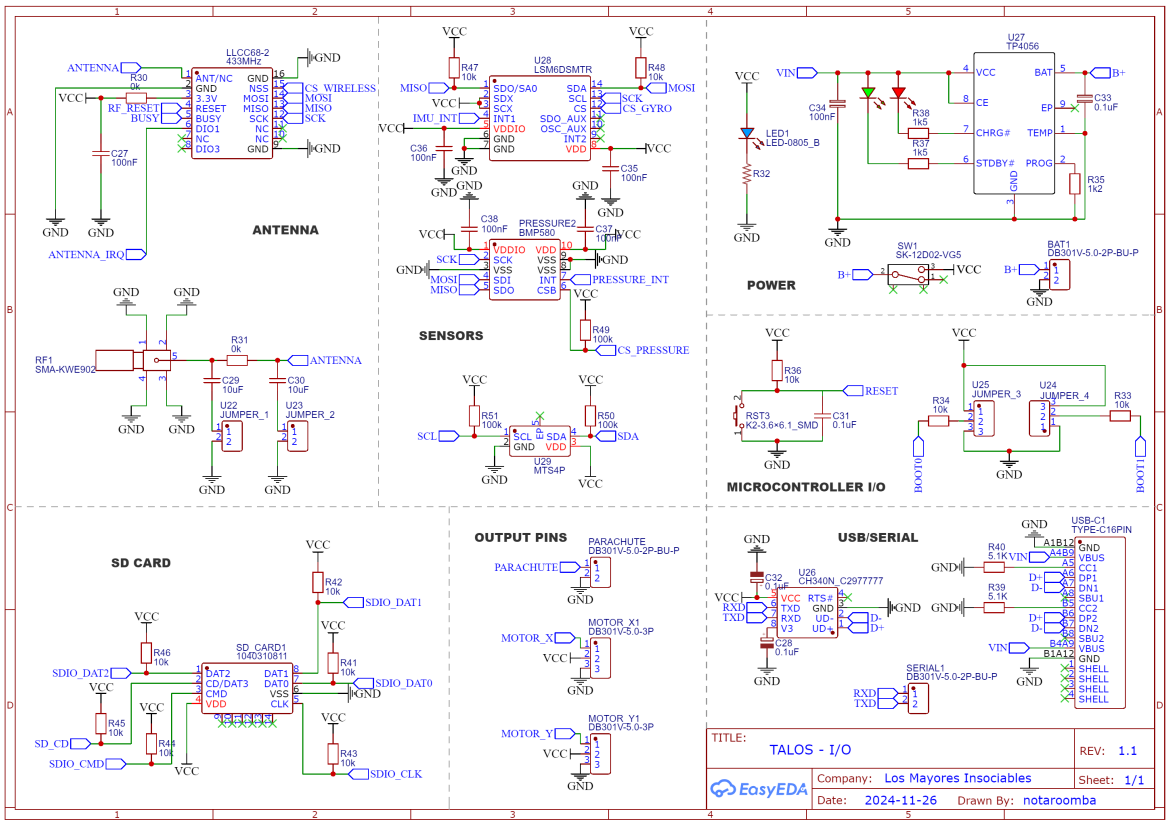


Figure 7: Website with no data

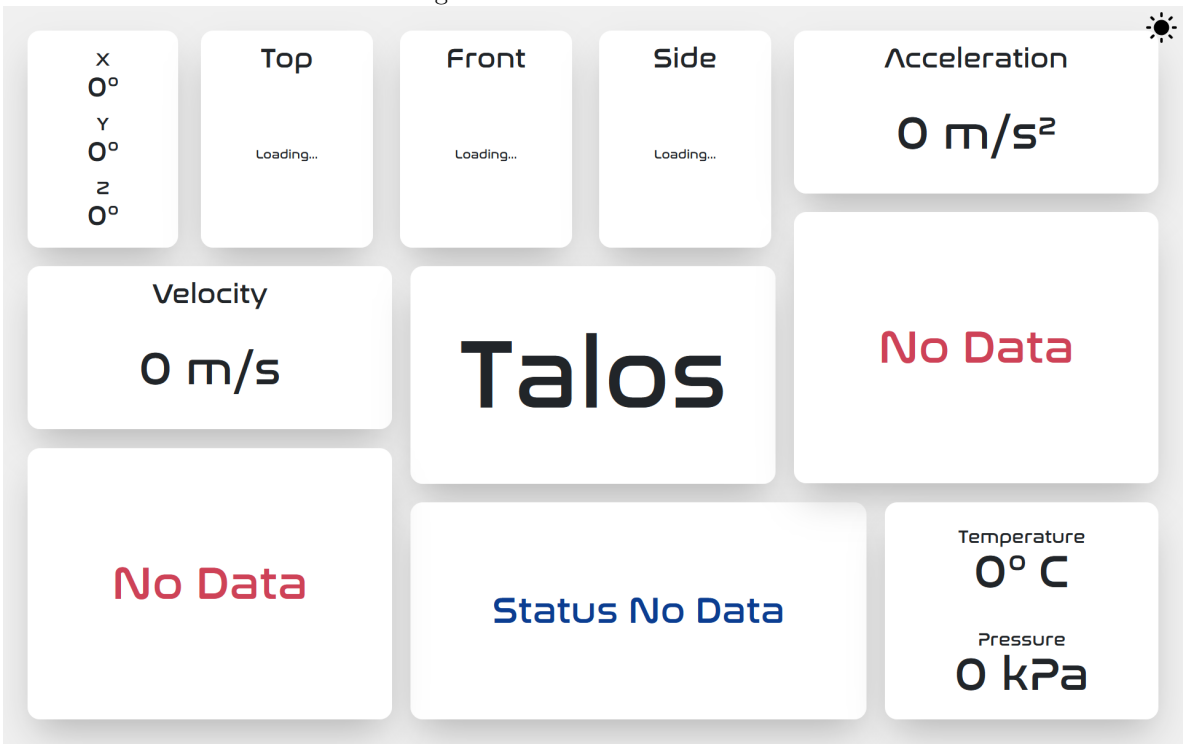


Figure 8: Website with data