Technical Manual for Data transfer system for UAV-deployed sensor packages

Smoaking Hot

4/19/2024

Data transfer system for UAV-deployed sensor packages

Team:

Smoaking Hot

Members:

Jared Peters, Steve Howard, and Jacob Tapp (Team Lead)

Date:

April 19th, 2024

Statement:

This is submitted in partial fulfillment of the requirements for ELCT 404 Spring 2024 at the University of South Carolina. It shall be understood that submission of this document by one member of the team means that each other member of the team has individually approved of its content.

Table of Contents

Introduction:

The project consists of two devices, the sensor package and the waystation. The sensor package is a standalone device that runs on a battery and is small enough to be drone deployed. It gathers accelerometer data, stores it into a SD card in csv format, and transmits the data to the waystation. These functions are performed by the C++ code executed on the Teensy 4.0 microcontroller board that uses libraries specifically made for the modules chosen. The Teensy interfaces with the SD card and the nRF24L01 via SPI bus. Refer to operating instructions for information on operating the sensor package. The sensor package can store 8 Gb of data with the 8 Gb SD card used on it currently and can collect at a sampling rate as high as 1 kHz.

The waystation is a device that can be put into a box and mounted on a pole. It receives the accelerometer data from the sensor package and performs computations on the data. These computations consist of FFT, mean, and standard deviation. The Raspberry Pi receives data via serial connection through a Teensy 4.0 connected to an nRF24L01. The data now on the Raspberry Pi is processed using a python program to output the required calculations. Refer to operating instructions for information on operating the waystation. The waystation receives data at a rate of 456.4 points per second and the time to compute the FFT is 0.00167 seconds per value.

Operating Instructions:

Materials Needed:

- Raspberry Pi Model 4B
- HDMI Cable
- HDMI to USB-C Adapter
- 3 MicroUSB cable
- Power Block
- Keyboard
- Mouse
- Wall Plugin
- 2 Teensy 4.0
- MPU 6050
- 2 nRF24L01
- SD card module
- 32 Gb SD card
- SD card of any value greater than 4Gb

Waystation:

To begin operating this device you would need to connect the Raspberry Pi Model 4B to an HDMI cable that connects to a screen that supports HDMI using a HDMI to USB-C adapter as shown below.

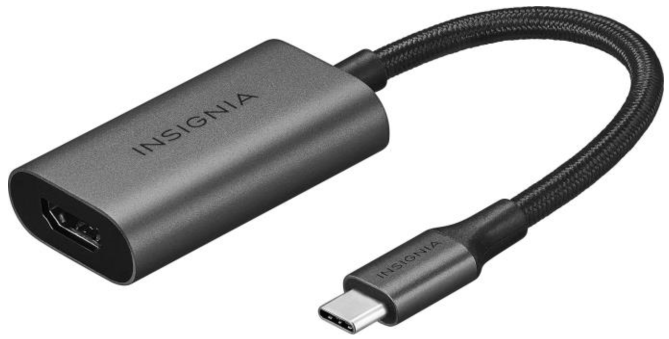


Figure 1: HDMI to USB-C adapter

Next you would connect a keyboard and mouse to the Raspberry Pi Model 4B as shown. Then you connect the Teensy 4.0 to the Raspberry Pi Model 4B via USB as shown. Once all these are connected, you can power the Raspberry Pi Model 4B using a power block and a micro-USB.

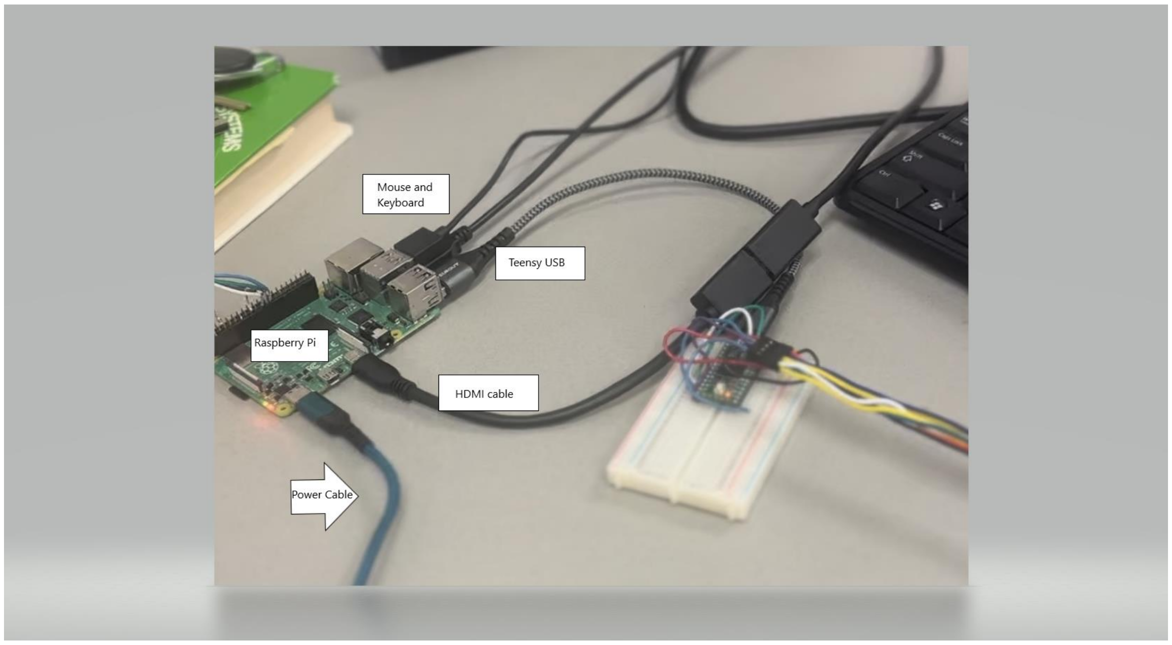


Figure 2: Full plugged in Waystation

Once the Raspberry Pi Model 4B has been powered you will see the screen shown below. Navigate your mouse to the top left of the screen and click the Raspberry logo. Then click programming and Thonny IDE.

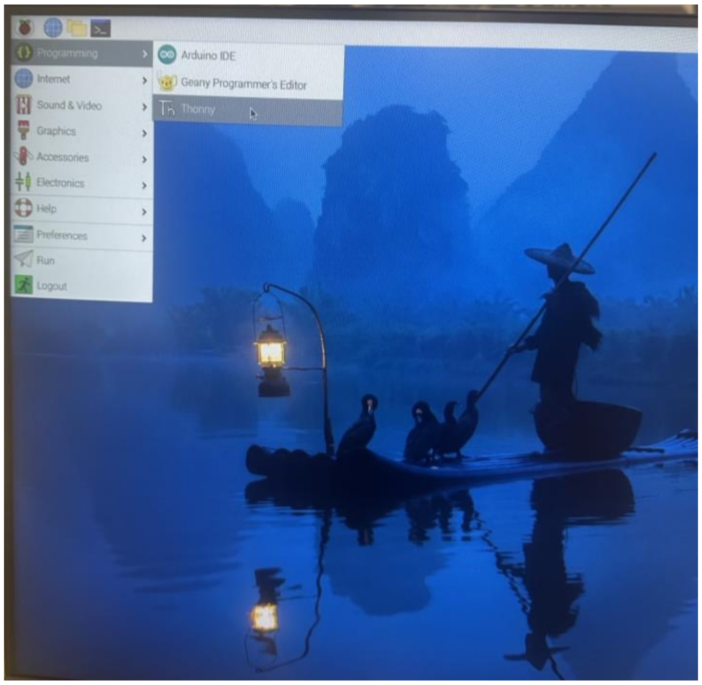
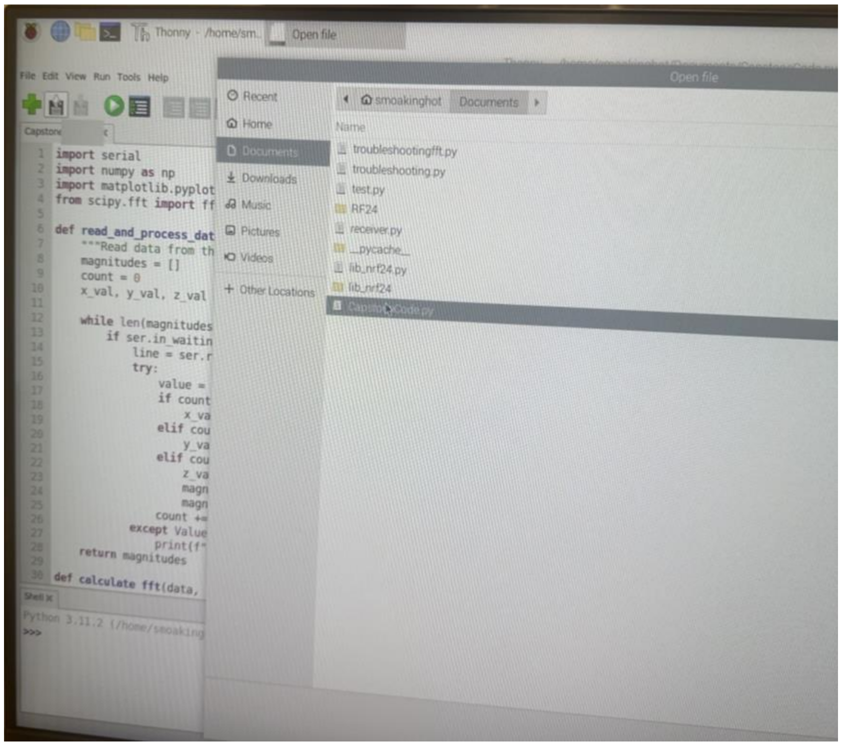


Figure 3: Raspberry Pi Home screen

From here you will load the program labeled Capstone Code as shown below.

It should be in the following directory: smoakinghot/Documents/Capstone Code

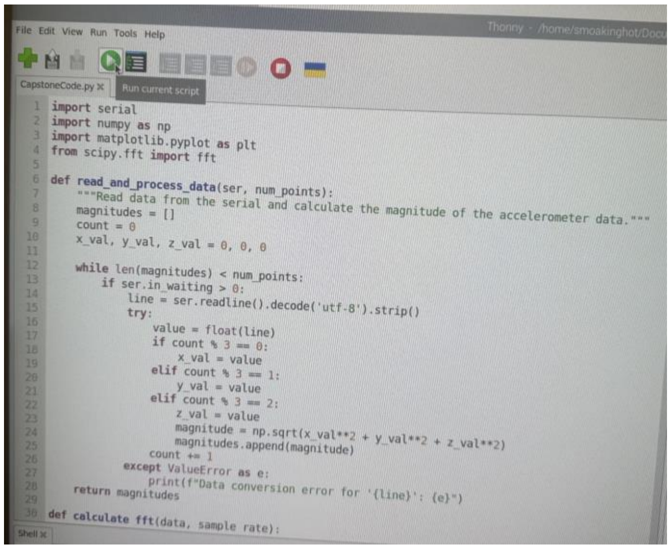


```
import serial
import numpy as np
import matplotlib.pyplot
from scipy.fft import ff

def read_and_process_dat
    """Read data from th
    magnitudes = []
    count = 0
    x_val, y_val, z_val

    while len(magnitudes
        if ser.in_waitin
            line = ser.r
            try:
                value =
                if count
                    x_va
                elif cou
                    y_va
                elif cou
                    z_va
                    magn
                    magn
                count +=
            except Value
                print(f"
    return magnitudes
def calculate fft(data,
```

```
Python 3.11.2 (/home/smoaking
>>>
```

Figure 4: Program Labeled Capstone Code

From here simply click Run and it will wait for sensor package data.


File Edit View Run Tools Help
Thonny - /home/smoakinghot/Docu
CapstoneCode.py
Run current script
Shell

```
import serial
import numpy as np
import matplotlib.pyplot as plt
from scipy.fft import fft

def read_and_process_data(ser, num_points):
    """Read data from the serial and calculate the magnitude of the accelerometer data."""
    magnitudes = []
    count = 0
    x_val, y_val, z_val = 0, 0, 0

    while len(magnitudes) < num_points:
        if ser.in_waiting > 0:
            line = ser.readline().decode('utf-8').strip()
            try:
                value = float(line)
                if count % 3 == 0:
                    x_val = value
                elif count % 3 == 1:
                    y_val = value
                elif count % 3 == 2:
                    z_val = value
                    magnitude = np.sqrt(x_val**2 + y_val**2 + z_val**2)
                    magnitudes.append(magnitude)
                count += 1
            except ValueError as e:
                print(f"Data conversion error for '{line}': {e}")
    return magnitudes
def calculate fft(data, sample rate):
```

Figure 5: Run Button

Sensor Package:

The sensor packages are ready for deployment so all you would need to do is connect it to a power source using micro-USB as shown below and it will begin collecting and sending data. To restart the sensor package with its current working iteration, you have to unplug the micro usb and plug it back in. Do not press the button on the Teensy to restart the board. This has caused issues with the sensor package causing it to not function properly. If this does happen, then you must reupload the code onto it.

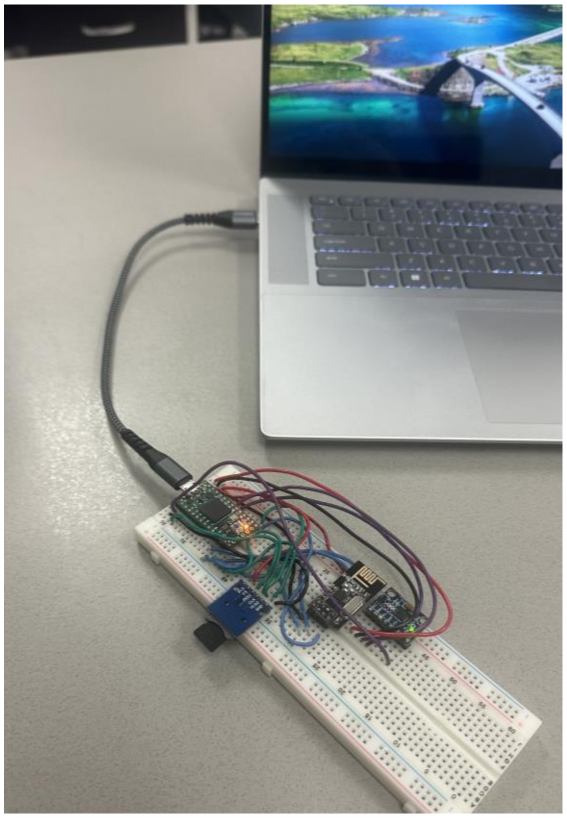


Figure 6: Sensor Packaged Powered

Technical Description

Team Philosophy:

Our team's vision is to provide smoking hot products that are innovative and reliable. This is achieved by our philosophy that centers around ease and scalability. To accomplish this we use affordable and reusable materials in our products that can be easily repeated and flexible to change and troubleshooting. To increase our products' lifetime, we implement risk mitigation tactics to prepare for hazardous conditions where our waystation and sensor packages will be located.

Components and Circuits:

The sensor package contains an SD card module, nRF24L01, MPU6050, and a Teensy 4.0 while the waystation contains a Raspberry Pi 4B, Teensy 4.0, and an nRF24L01.

The SD card module interfaces with an SD card to write and read data from it. It is connected to a controller or other device via Serial Peripheral Input bus or SPI bus for short. For our usage, the Module is wired to the Teensy 4.0 microcontroller.

The nRF24L01 which is used on both the sensor package and the waystation is a wireless module that uses radio to communicate. Like the SD card, it also connects via SPI bus to the Teensy 4.0 on both the sensor package and the waystation. There are two configurations of the board. One configuration, the nRF24L01, is smaller with limited range and contains an internal antenna. The second configuration, the nRF24L01+, is a larger board with longer range capability and comes with an external antenna. The plus version of the board requires a more stable supply of power to operate.

The MPU6050 is a multipurpose sensor that collects accelerometer data, gyroscope data, and internal temperature. The accelerometer data, the focus of our project, is gathered in gravity magnitudes. The device connects to the analog pins of the Teensy 4.0.

The Teensy 4.0 is a programmable microcontroller prototyping board that uses Arduino IDE in C++. The device has analog pins, one SPI bus, and digital pins. For our purposes it functions as the brain of the sensor package and the receiver go-between for the waystation.

The Raspberry Pi 4 B is a general-purpose computer aimed at providing anyone with internet access to a reasonably powerful computing device. It is powerful enough to function as the waystation for our purposes. Its high processing power makes it an excellent option for generating FFTs.

In figure 7 below is a circuit schematic for the sensor package that illustrates its connections to each of its components. The schematic was made in KiCAD.

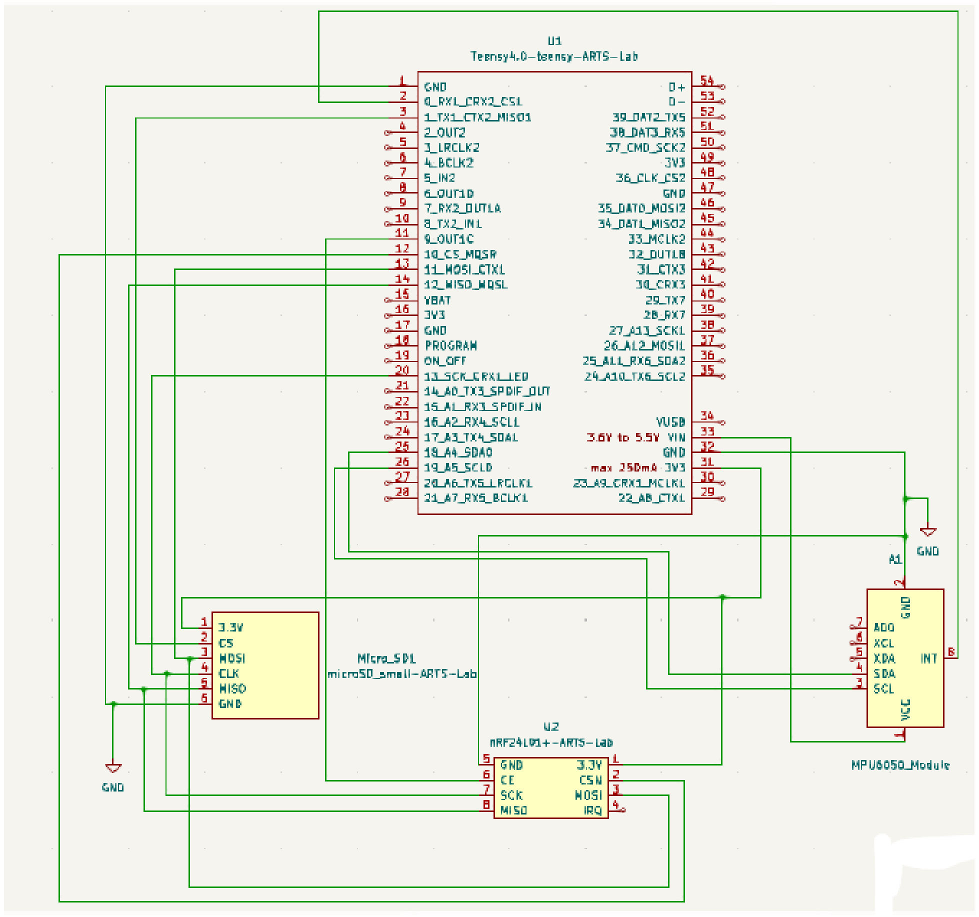


Figure 7. Sensor package circuit schematic.

System Performance:

The Sensor package is a device with several different components each boasting their own specifications that affect the overall performance of the system. It is important to note the limitations of these components as well as the limitations of the software.

As noted in the introduction, The MPU6050 has a limit of 1kHz sampling rate which is noted in its datasheet. The device has no lower limit and can be adjusted for in the code accordingly.

The nRF24L01 is reportedly capable of about 100 meters with the nRF24L01+ being capable of up to 1000 meters. We did not achieve this range in our project; however, we believe that is due to issues providing the device with a strong steady signal. In our tests, we achieved 8 feet between the transmitter and receiver. It is worth noting that this range was tested using the nRF24L01, not the plus as it would not work at all due to power supply issues.

The SD card performs well with a standard read/write speed of 12.5 MB/s and a high speed twice that value. Unless you wanted to store millions of data points, the read/write speed would never affect the system in any noticeable way. It is worth noting that the standard number of points we need to collect is 70,000.

The software of the sensor package and the waystation are the most impactful in the system's performance. For the sensor package to gather data, it needs to do so at the users specified rate, meaning that data collection time is very variable. It is worth noting that data transfer cannot begin before data collection is complete, because the time required to transmit the data points in the middle of data collection would change the sampling rate in unpredictable ways. The variability in data collection time due to user requests can mean that data collection could take 1 second with a higher sampling rate and a lower number of samples or it could take an infinite amount of time. For testing purposes, we used 1000 data points and a sample time of 9.5 seconds.

The performance of the software on the waystation is also directly dependent on the number of data points collected by the sensor package. The Raspberry Pi can generate an FFT at a rate of 456.4 points per second. Thus, there is a direct correlation between the number of data points and the time it takes to generate the FFT. This means that with 70,000 data points it will take 153 seconds (about 2 and a half minutes) to generate the FFT for that number of points.

Performance Trade-offs:

One of the main trade-offs in the design of this device is the trade-off between precision and time. This is referring to the correlation between data points and processing time being directly proportional.

Another trade-off was in the problems we devoted our attention to. As stated above, we only achieved an effective wireless communication range of 8 feet, while other users using the same module reported a greater range of about 100 meters (about 328.08 ft). Rather than trying to solve this issue, after bringing it up with our sponsor, we decided to devote our attention to optimizing our software. At the time, neither we nor our sponsor were aware of the nRF24L01 module Power demands and the Teensy 4.0’s limited ability to supply that power.

Reprogram the system:

It is important to note that the number of data points collected and the sampling rate are designated in the code of the sensor package. To change these values, edit the code slightly. To change the number of samples, change the value for the datapoints variable defined early on in the code before the setup. To change the sample rate, change the wait time in the data collection loop. This will determine the wait time between the collection of each point.

It is also possible to change the name of the file the data is stored in. This can be changed by editing the line that starts with “SD.Open”.

Standards/Codes Adhered To:

For the purposes of our project, there were no standards we had to adhere to. This is due to the nature of our sponsors' work, being research and not shipping products to be used practically. We were told that the waystation and sensor package were for proving functionality purposes only.

Block Diagrams:

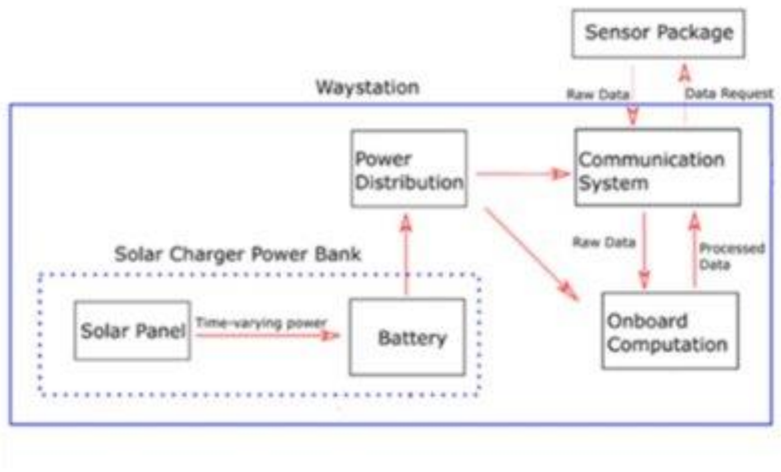


Figure 8. System block diagram for the waystation

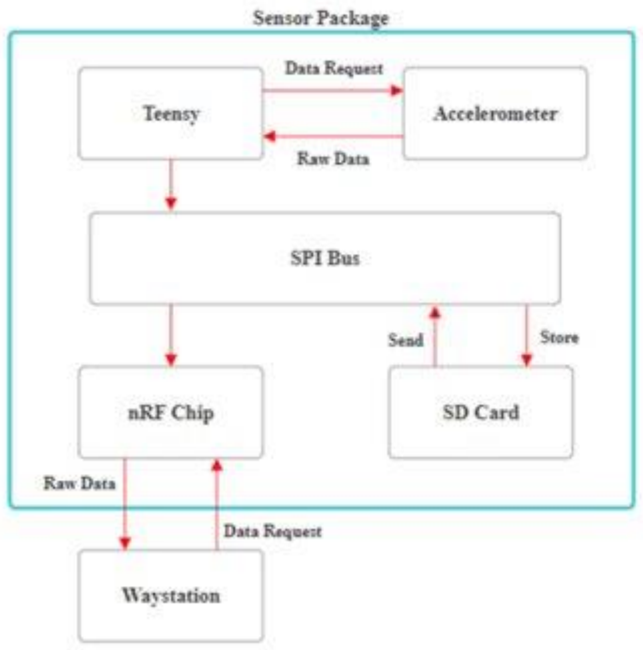


Figure 9. System block diagram for the sensor package

Troubleshooting Tips:

Hardware:

- Use a multimeter to test for continuity (buzz test) on the PCB’s or Breadboards
- Consult the wiring diagrams for the Sensor Package
- Review Data Sheets for specific expectations of a specific module

Software:

- Review the nRF libraries for extra troubleshooting code
    - PrintRadioDetails() is a great tool to see if the radio is connected and initializing
- Don't be afraid to make it print something to the console to see if data is being sent to the right variable or radio
- Restart Application or Hardware
- Use the Reprogram button on Teensy 4.0
- Always save an extra file before editing the last so you know what you changed

Appendix

Patent Search:

A patent search was conducted for our product at United States Patent and Trademark Office (USPTO) Online Directory. The following words were used in the advanced search category.

- Bridge Sensors
- Waystation
- Sensor Packages
- Bridge Data Collection

When reviewing the first ten patents of each search (40 total) none of the patents directly mentioned anything regarding our project or design. Although, we do not necessarily believe that this product passes the criteria for patentability due to its lack of novelty and the fact that it's open source for anyone to use.

Public Impact Statement:

In our ongoing commitment to public safety and traffic infrastructure, our latest innovation in bridge stability testing technology represents a significant advancement in the monitoring and maintenance of bridge structures. This technology employs a combination of advanced sensors and data analytics to assess the structural integrity of bridges in real-time, providing essential information that can prevent accidents and extend the lifespan of vital transportation assets.

Overall, we believe that this product will allow for safer communities, cost savings, and environmental protection from large structure collapses.

Prototype to Production Plan:

The next steps for our project to reach the overall goals of our design would be the following:

1. Turn the Waystation into a Transceiver
2. Remove the Interconnect Teensy in the Waystation
3. Order the new set of PCB's
4. Design the Physical Waystation Cover
5. Increase Range of Sensor Packages

1: Turning the Waystation into a Transceiver

Instead of constantly waiting for the data to be sent to the waystation, ideally it would be user prompted. This would save on power expenses and the overall lifetime of our product.

2: Remove the Interconnect Teensy in Waystation

Currently wireless communications are done through nRF chips that are connected to the teensy in the way station that then uses a serial (micro-USB) connection to the Raspberry Pi to conduct the onboard processing. Ideally this is not efficient because it makes the product cost more and has more troubleshooting possibilities. This can be removed by connecting the GPIO pins directly to the nRF module and changing the program to enable communication to other teensys.

3: Order New PCB's

Our product currently is using a breadboard due to the first set of PCBs not having a power regulating addition to power the nRF24L+ (external antenna) due to it needing more consistent power than the original nRF module. The design is already made and waiting to be ordered.

4: Design Physical Waystation Cover

All the testing done has been done inside due to the lack of range but for practical use our product will need an environmentally-proof cover that is secure for public use.

5: Increase Range of Sensor Packages

When designing this product, we were using the nRF24L01 but ended up purchasing nRF24L01+ for longer ranges. This needs to be implemented and tested.