ECEN 4013 Project 2: Localization Device Sprint 2 Report

Jimmy Vich
Abdullah Alenezi
Greg Lew
Keenan Holsapple

April 17 2023

Contents

1 Design Concept

1.1 Introduction

This is the second project for ECEN 4013: Design of Engineering Systems. This class and project are designed to immerse us in real-life project environments. We utilize the skills in this class to understand fundamental project planning and practice teamwork.

1.2 Project Goal

Our goal for this project is to create a device that can locate itself, transmit, and store this data. Utilizing different devices that collaborate with each other-typically following a master/slave communication scheme, the microcontroller will give us an output of its relative position. This requires coding a microcontroller and physically planning a board that can achieve all of the desired results properly.

2 Constraint Considerations

The main constraint that we have is time, with other smaller constraints such as our knowledge, our coding ability, and our access to manufacturing equipment. The largest constraint that we have is time because lack of time prevents us from studying our parts more, making a better design, adds to the chance that we may not receive parts, and with everything outside of this project that we have, it makes it difficult to schedule meetings and project deadlines. Our smaller constraints also have an effect on how well our project will end. We will have to gain knowledge of how the individual parts work. Next, we will have to learn how to program the system. Finally, the last constraint is our access to manufacturing equipment, and that will simply be corrected by spending time in our labs waiting for the necessary equipment to become available.

3 Parts List

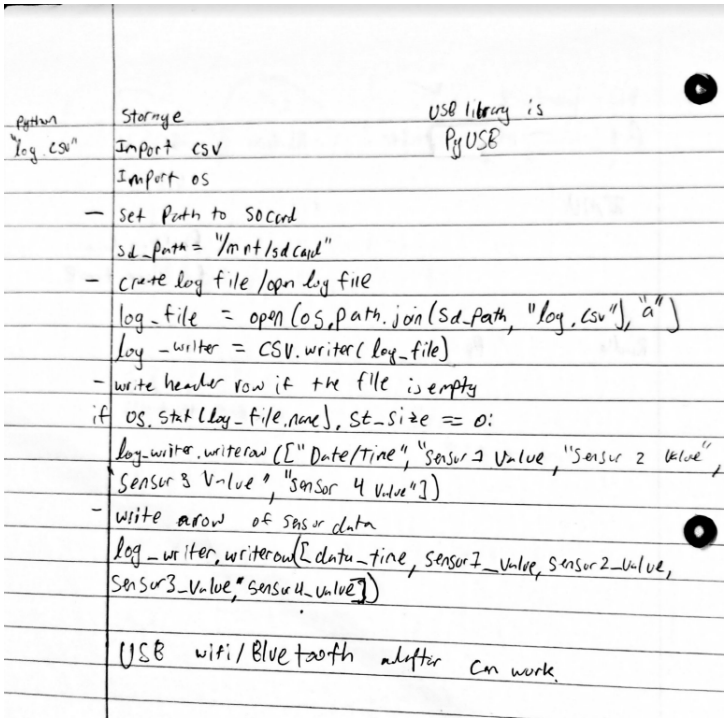
Here, we list the current status of our parts and how we plan to utilize them. To find further details, outputs from unit testing and communication protocols for chips are listed further in the report.

Note: Some of the parts that we ordered do not arrive until later this week. However, we do have parts from the storage room.

- GPS: We debated a long list of GPSs however we currently have a SKY65933-11 on order that should be arriving this week.
- Microcontroller: We debated the Rasphberry Pi Zero; however, we currently have a Raspberry Pi Pico that we are going to begin programming

soon. We will be using MicroPython, the open source library that allows coding in Python on board.

- Radio: We debated a long list of radios; however, we plan to use a HC-05 Bluetooth module. With the module we can open a serial port on a PC or Andriod device to read data being output.
- IMU: We will be utilizing a BNO055 and narrowing it's outputs to the only ones required:
  1. Magnetometer
  2. Angular Velocity
  3. Acceleration
- SD-Card-USB: ADA254 is a microSD card extention that can read CSV files to a microSD. Code below creates the library and creation of the file to put the data on.



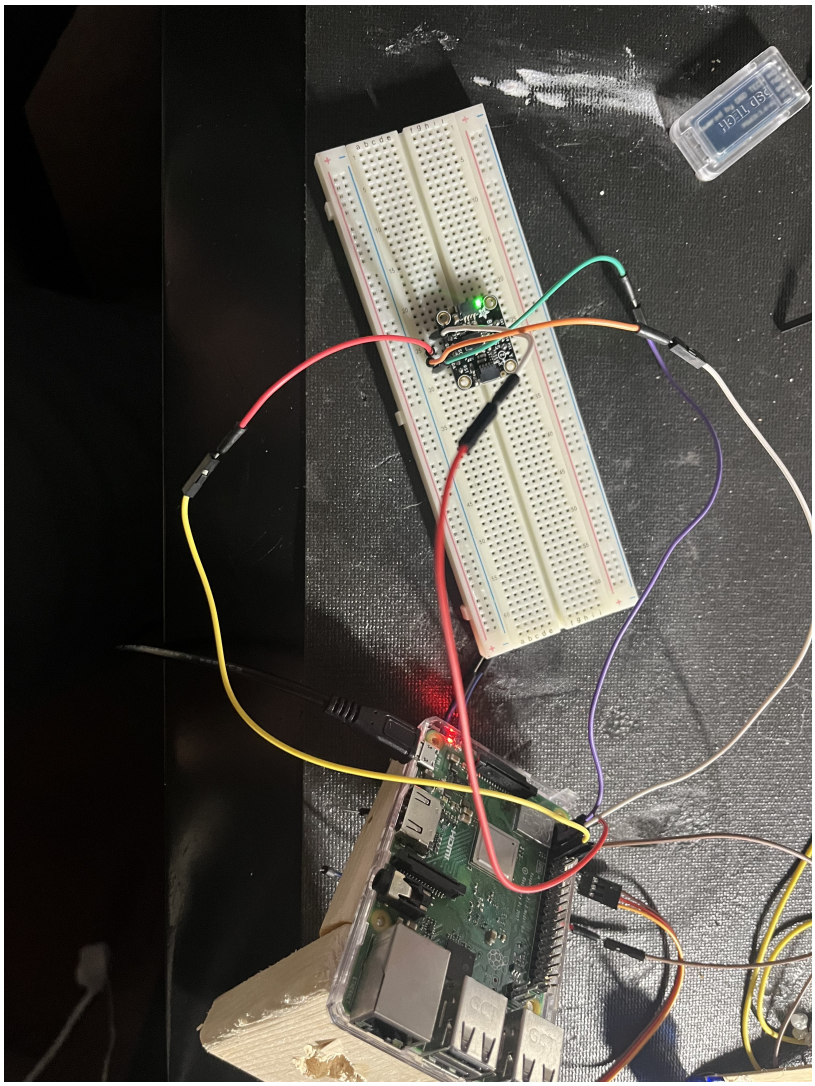
3.1 Part Outputs

Right now, we have unit test and codes that can be implemented into the overall processing of the Rasperry Pi Pico. Here are the current outputs:

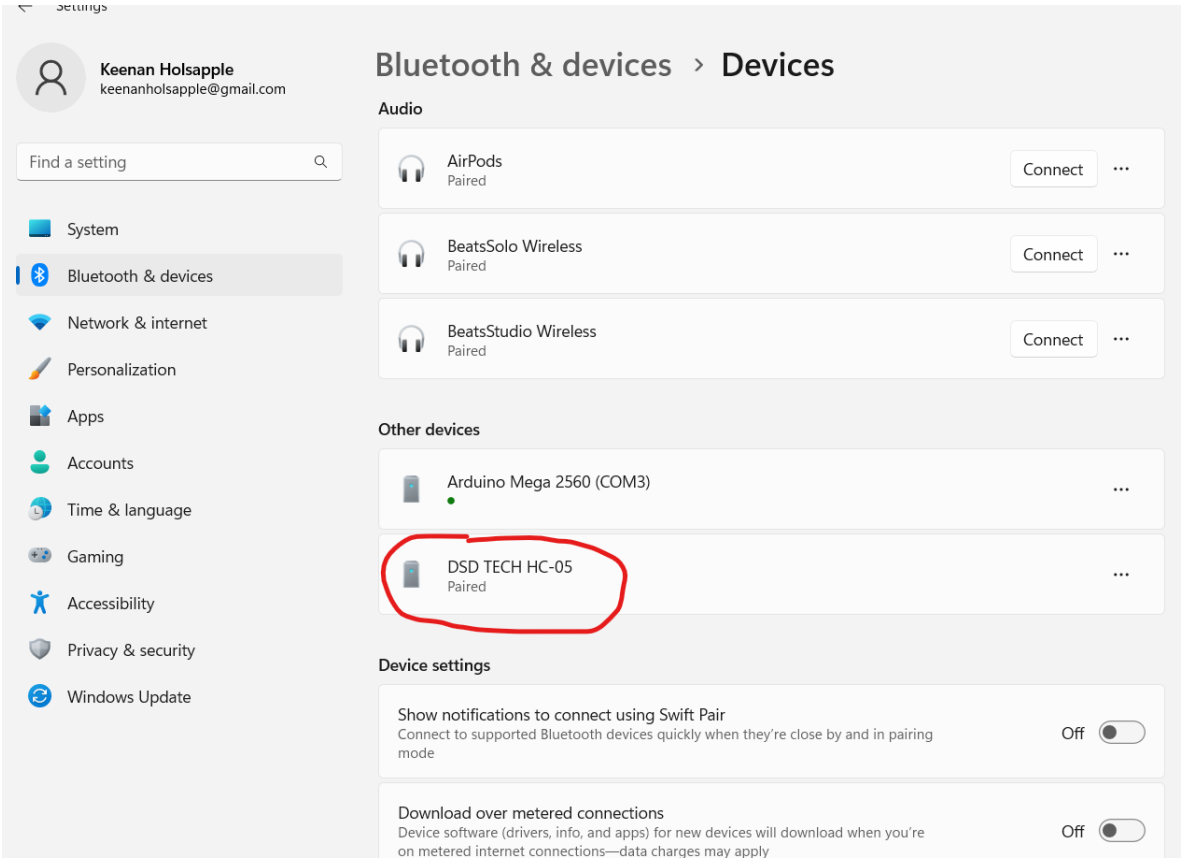
- IMU - Getting all the values available on the current chip we have, we were able to find the following outputs:

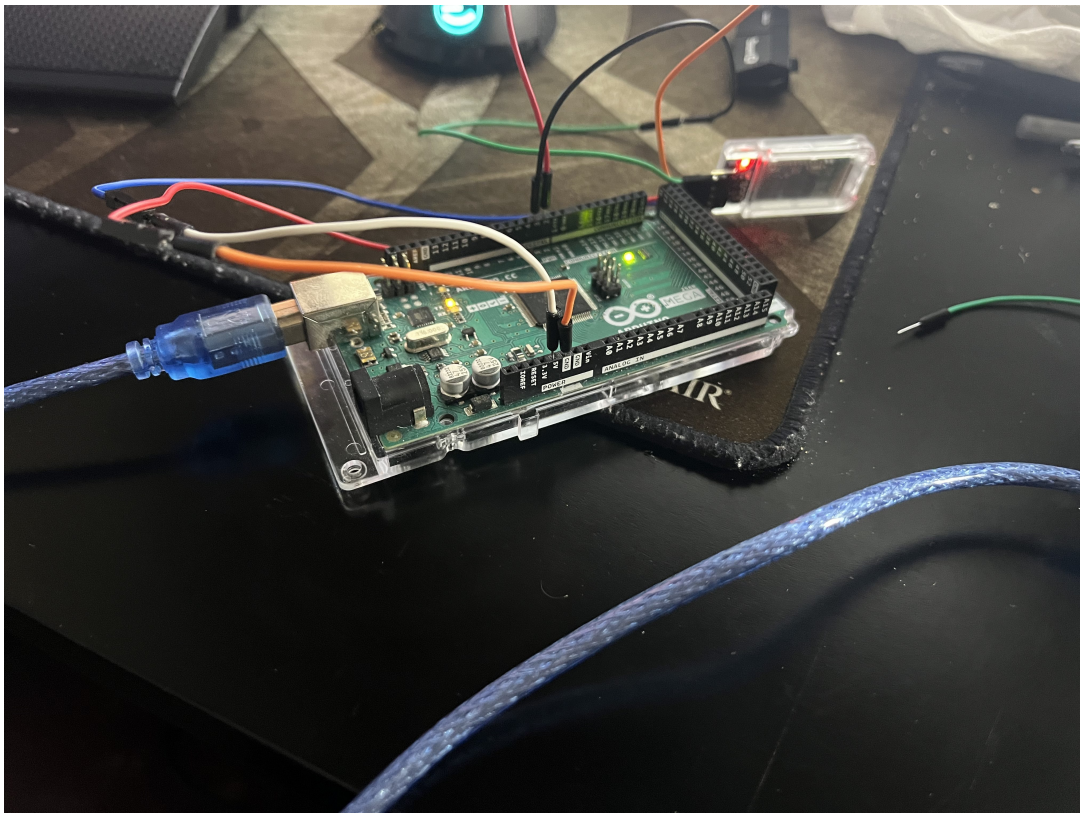
```
Gravity (m/s^2): (0.06, -0.38, 9.790000000000001)

Temperature: -104 degrees C
Accelerometer (m/s^2): (0.8300000000000001, 1.02, 9.17)
Magnetometer (microteslas): (1.0625, -1.0625, 0.0)
Gyroscope (rad/sec): (0.0, 0.0, 0.0)
Euler angle: (0.0, 1.8125, -0.125)
Quaternion: (0.9998779296875, 0.00115966796875, -0.015869140625, -0.00030517578125)
Linear acceleration (m/s^2): (0.5, 0.88, -0.85)
Gravity (m/s^2): (0.32, 0.05, 9.8)

Temperature: -105 degrees C
Accelerometer (m/s^2): (0.49, 1.37, 9.35)
Magnetometer (microteslas): (1.0625, -1.0625, 0.0)
Gyroscope (rad/sec): (0.0, 0.0, 0.0)
Euler angle: (0.0625, 3.0625, 2043.75)
Quaternion: (0.99896240234375, 0.0372314453125, -0.0269775390625, -0.0009765625)
Linear acceleration (m/s^2): (0.05, 0.41000000000000003, -0.39)
Gravity (m/s^2): (0.52, 0.73, 9.76)

Temperature: -105 degrees C
Accelerometer (m/s^2): (-0.01, 1.05, 9.84)
Magnetometer (microteslas): (1.0625, -1.0625, 0.0)
Gyroscope (rad/sec): (0.0, 0.0, 0.0)
Euler angle: (0.1875, 2.5, -7.0625)
Quaternion: (0.99786376953125, 0.06182861328125, -0.02215576171875, -0.00189208984375)
Linear acceleration (m/s^2): (-0.39, -0.21, 0.03)
Gravity (m/s^2): (0.42, 1.2, 9.72)

Temperature: -104 degrees C
Accelerometer (m/s^2): (-0.49, 0.77, 10.11)
Magnetometer (microteslas): (1.0625, -1.0625, 0.0)
Gyroscope (rad/sec): (0.0, 0.0, 0.0)
Euler angle: (0.3125, 0.3125, -6.0)
Quaternion: (0.99859619140625, 0.05279541015625, -0.0029296875, -0.00299072265625)
Linear acceleration (m/s^2): (-0.46, -0.59, 0.36)
Gravity (m/s^2): (0.04, 1.02, 9.75)
```

```
Temperature: 24 degrees C
Accelerometer (m/s^2): (0.12, -0.39, 9.78)
Magnetometer (microteslas): (1.0625, -1.0625, 0.0)
Gyroscope (rad/sec): (0.0, 0.0, 0.0)
Euler angle: (0.125, 0.8125, 1.5)
Quaternion: (0.9998779296875, -0.01318359375, -0.00732421875, -0.00140380859375)
Linear acceleration (m/s^2): (0.0, -0.12, -0.04)
Gravity (m/s^2): (0.14, -0.25, 9.8)

Temperature: -104 degrees C
Accelerometer (m/s^2): (0.12, -0.39, 9.790000000000001)
Magnetometer (microteslas): (1.0625, -1.0625, 0.0)
Gyroscope (rad/sec): (0.0, 0.0, 0.0)
Euler angle: (0.125, 0.8125, 1.5)
Quaternion: (0.9998779296875, -0.01318359375, -0.00732421875, -0.00140380859375)
Linear acceleration (m/s^2): (-0.01, -0.12, -0.04)
Gravity (m/s^2): (0.14, -0.25, 9.8)

Temperature: 24 degrees C
Accelerometer (m/s^2): (0.12, -0.38, 9.75)
Magnetometer (microteslas): (1.0625, -1.0625, 0.0)
Gyroscope (rad/sec): (0.0, 0.0, 0.0)
Euler angle: (0.125, 0.8125, 1.5)
Quaternion: (0.9998779296875, -0.01318359375, -0.00732421875, -0.00140380859375)
Linear acceleration (m/s^2): (0.0, -0.11, -0.04)
Gravity (m/s^2): (0.14, -0.25, 9.8)

Temperature: -104 degrees C
Accelerometer (m/s^2): (0.12, -0.38, 9.77)
Magnetometer (microteslas): (1.0625, -1.0625, 0.0)
Gyroscope (rad/sec): (0.0, 0.0, 0.0)
Euler angle: (0.125, 0.8125, 1.5)
Quaternion: (0.9998779296875, -0.01318359375, -0.00732421875, -0.00140380859375)
Linear acceleration (m/s^2): (0.0, -0.13, -0.03)
Gravity (m/s^2): (0.14, -0.25, 9.8)
```



- Bluetooth - The HC-05 is not compatible directly with a higher level RaspberryPi (as we were unit testing with a RaspberryPi 3B+). However, implementing the same code it will utilize with the Pico, we tested it with an Arduino on hand and saw that it is able to connect on PC and mobile devices:


Settings
Keenan Holsapple
keenanholsapple@gmail.com
Find a setting
System
Bluetooth & devices
Network & internet
Personalization
Apps
Accounts
Time & language
Gaming
Accessibility
Privacy & security
Windows Update
Bluetooth & devices › Devices
Audio
AirPods
Paired
Connect
BeatsSolo Wireless
Paired
Connect
BeatsStudio Wireless
Paired
Connect
Other devices
Arduino Mega 2560 (COM3)
DSD TECH HC-05
Paired
Device settings
Show notifications to connect using Swift Pair
Connect to supported Bluetooth devices quickly when they're close by and in pairing mode
Off
Download over metered connections
Device software (drivers, info, and apps) for new devices will download when you're on metered internet connections—data charges may apply
Off



4 Task Division

4.1 Administration Roles

The administration roles are broken down into these four sections, assigned to the team members:

- Scrum Master: Greg Lew
- Scribe: Keenan Holsapple
- Hardware/Code Architect: Abdullah Alenzi
- Point of Contact: Jimmy Vich

4.2 Technical Roles

The technical roles are an assignment to each part of the circuit. This ensures that each person is contributing effort to the physical design of the circuit. These assignments are:

- GPS: Abdullah Alenezi
- Micro-controller: Jimmy Vich
- Radio and Storage: Greg Lew
- IMU: Keenan Holsapple

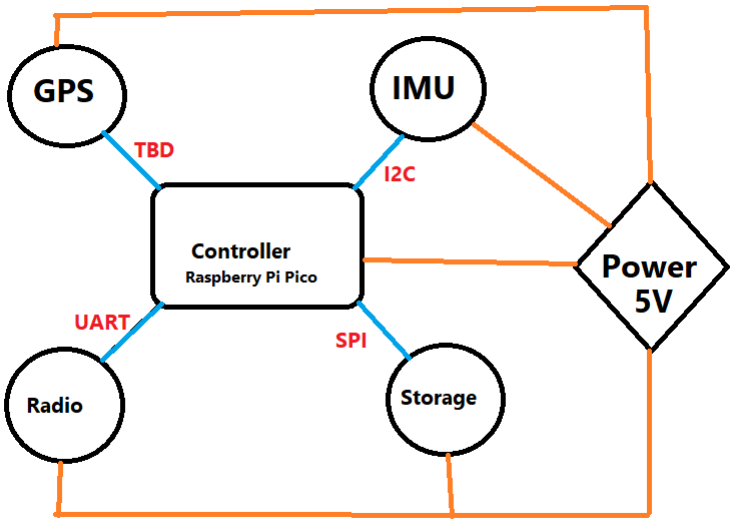
5 Psuedocode

Algorithm 1 Raspberry Pi

```
Start Serial Port
Initialize CSV format on microSD
while True do
    Begin GPSRead():
    Read GPS data
    Convert to iterable list/array
    If list index == 4, return array
    Begin IMURead():
    Read IMU data
    Convert to iterable list/array (2D; [3,2] tuple)
    if list index == 3, return array
    Begin BluetoothWrite(GPSArr, IMUArr):
    Upload array to serial write, return true
    Begin SDWrite(GPSArr, IMUArr):
    Append CSV sheet with GPSArr and IMUArr, return true
    If microSD card not present, end loop
end while
```

6 Block Diagram

In this image, we specify the communication protocols for each respective chip, along with the fact that all of them can be powered at 5V.



7 Next Steps

As we enter the last leg of design for the project, we still have many things to accomplish. However, having the base

- Implement written function for exporting tabular data with input parameters of other parts.
- Write function for outputting input parameters of IMU and GPS over Bluetooth signal on Serial Monitor.
- Unit test GPS on Pico.
- Create full circuit on breadboard, implement current unit testing codes with Pico.
- Build/order PCB. (this needs to be done ASAP, we are ordering one online)
- Design/print chasis for PCB and power supply.
- Test.

8 References

- IMU unit testing: `https://learn.adafruit.com/adafruit-bno055-absolute-orientation-sensor/python-circuitpython`
- Bluetooth unit testing: `https://www.makerguides.com/arduino-and-hc-05-bluetooth-module-comp`

- CircuitPython on RaspberryPi: https://learn.adafruit.com/bno055-absolute-orientation-senso
software