ME 410 - Week 1

Christopher Luey

(a) Text Description (400 words max)

This week, we interfaced a BMI088 IMU with a Raspberry Pi Zero 2W via I2C to measure a quadcopter's orientation. We initialized the I2C communication by assigning accelerometer and gyroscope addresses (0x19 and 0x69 respectively) and verified device connectivity. In `setup_imu()`, we configured the accelerometer to ±3g range and the gyroscope to ±1000°/s. Next, we implemented `read_imu()` to obtain raw 16-bit values from the IMU, correctly applying two's complement conversion and unit scaling: accelerometer values converted to g's, gyroscope to °/s.

To derive pitch and roll, we used `atan2()` on the scaled accelerometer readings and converted the result to degrees. For calibration, we averaged 1000 samples in `calibrate_imu()` to compute gyro bias and static pitch/roll offsets, which we then subtracted during runtime to reduce drift. We continuously printed calibrated values—3-axis gyroscope, pitch, and roll—on a single line for clarity. This output verified both functional sensor communication and basic orientation estimation.

(b) Task Assessment

1. What went well:
We completed the assignment on time and achieved successful I2C communication with both accelerometer and gyroscope. Accelerometer and gyroscope behaved as expected and orientations were correct. Calibration and real-time data printing worked as expected.

2. What did not go well, why:
We encountered bugs from incorrect integer truncation during two's complement conversion. Fixing this required careful debugging and validation of sign handling.

3. What will you change for next class:
I hope to be more thorough in checking edge cases like integer conversion and overflow early in development to prevent time-consuming bugs. I also hope to improve the engineering workflow so we are not copying and pasting.

(c) Team Member Effort Report

Jason – 50%:

- Verified I2C address assignment and sensor config registers

    - Read datasheet to gather address for I2C config
- Computed pitch and roll and formatted print output

Christopher – 50%:

- Implemented `read_imu()` including sensor read logic and two's complement conversion
- Developed and debugged `setup_imu()` and `calibrate_imu()`

Both members contributed and were effective at completing the task. We collaborated on a single computer.