**Date:** 10/28/2022  
**Total hours:** 15  
**Description of design efforts:**

**Accelerometer**

Much effort was spent this week taking the working example code from Owen and getting the axis values to be read properly, calibrated, and integrated into gyro headings in correct units.

Starting out, the data was too slow, and had the wrong bit format. After fixing the bit shifting etc., and setting a higher data rate, it was clear all 6 axes were being properly read.

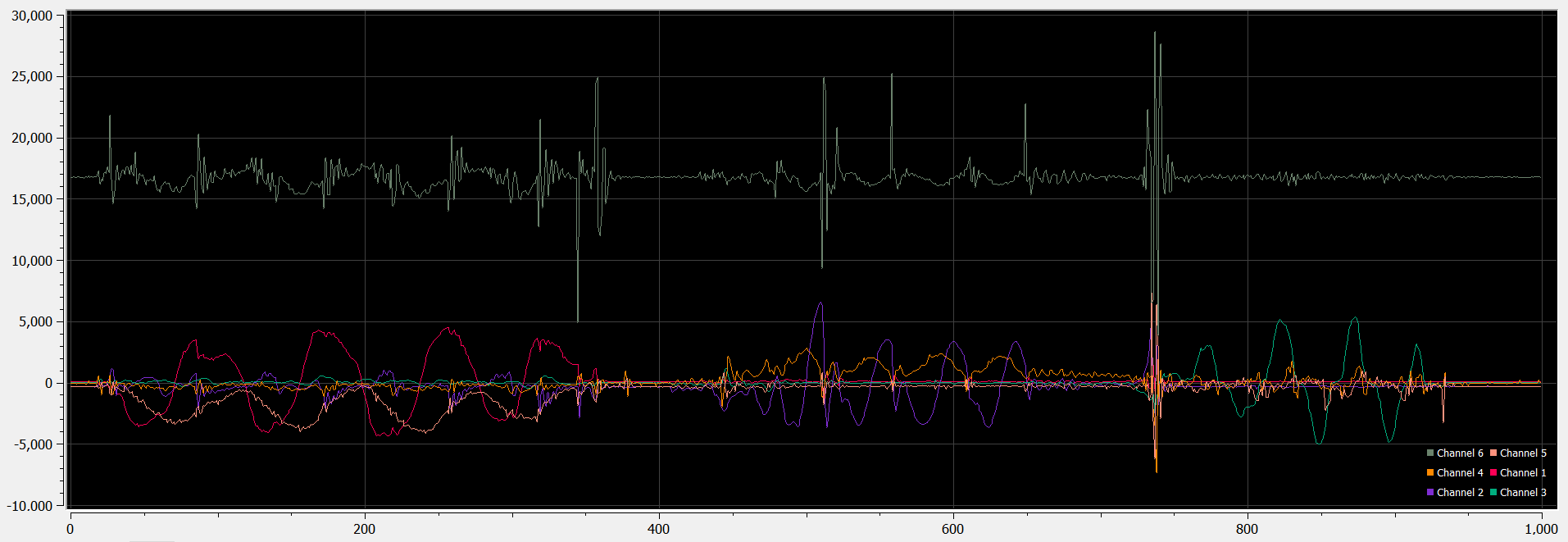


Fig. 10.1. 6 axes being read, accelerometer X,Y,Z, and gyro X, Y, Z.

Once this was clearly working, I created a working timing driver to measure sub-millisecond time deltas as well as those with sub-millisecond fidelity but multi-millisecond span.

To do this, I combined the Systick timer’s countdown register, and a ‘tick’ variable incremented on every systick interrupt. I calculated the Systick roll-over value based on the clock frequency such that the interrupt occurs once every millisecond, giving a simple 1KHz timer.

Text

Description automatically generated

Fig. 10.2. #defines determining Systick configuration.

To calculate inter-interrupt timing, I use the value of the systick down-counting register to determine how many decrements remain before rollover. When compared to the clock frequency (how many decrements per second), a floating point time in units of seconds can be calculated. I still need to determine however if fixed point math would be better suited to this though. The STM32F4 has an FPU to accelerate these calculations, although 32 bit fixed point may be better.

Graphical user interface, text

Description automatically generated with medium confidence

Fig. 10.3. Code to calculated floating point timestamp in seconds.

Once this was created, I could calculate the precise timestep between readings of the gyro data. This is crucial for accurate integration of the gyro angular rate data. Although much remains to be configured in terms of low pass filters, aliasing prevention, etc. I was able to obtain unit-correct heading data.

Text

Description automatically generated with medium confidence

Fig. 10.4. Code sample including LSB scaling and time delta.

(max 245 Degrees per second, signed 16 bit number -> 32768 max value)

Before this was usable however, I had to create calibration code, so that the average value returned by the gyro for angular rate would be zero. Otherwise the integration of the angular rate to get heading would result in immediate drift and a useless measurement. Therefore, the system sums up for each axis a few thousand readings at reasonable time intervals, then divides that sum by the umber of samples to get the average reading. That is then subtracted from all future readings to get a zero mean angular rate reading.

A picture containing text, receipt

Description automatically generated

Fig. 10.5. Calibration code

Owen is working this week on incorporating everything into #defines so that this code is much more configurable as we start tuning these parameters.

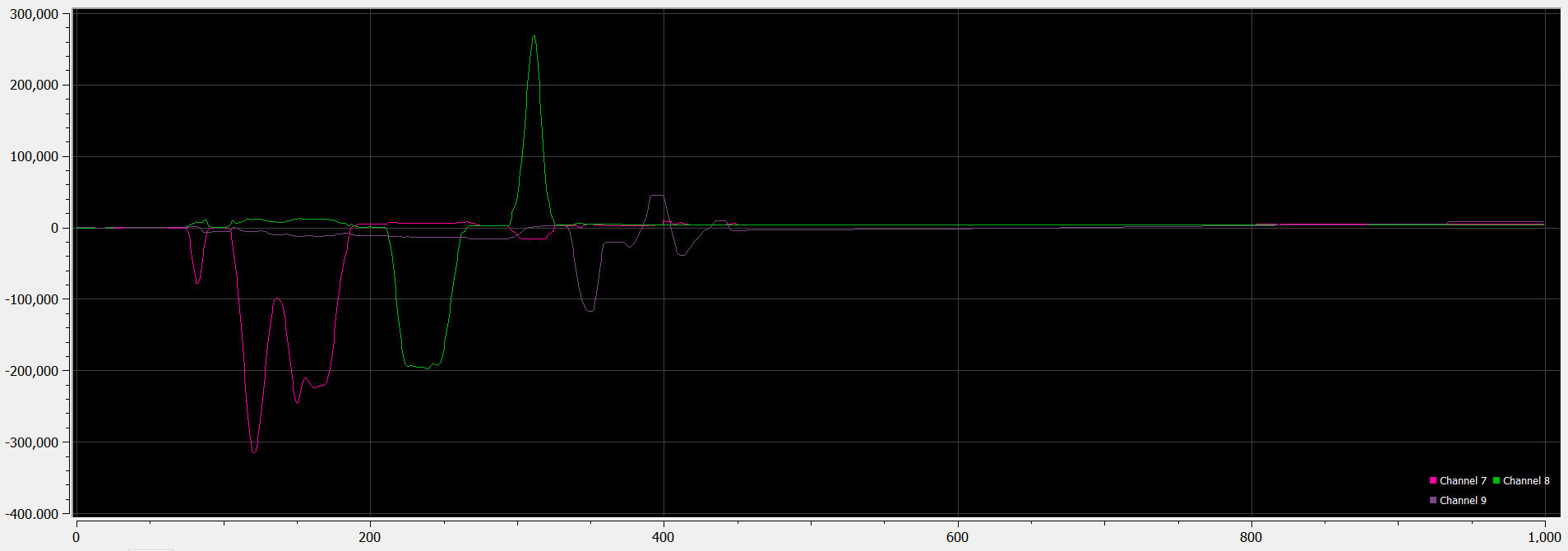


Fig. 10.6. Integrated gyro heading data (before scaling applied)

**Code organization**

Now that interfacing code is nearly complete, I am integrating each example source code project into its own driver file, headers, and separating base hardware drivers from device drivers (i.e. separate IMU driver from SPI driver). I am also ensuring all code is driven by #defines, so that clock frequency changes, baud rates, etc. can be quickly changed to their proper values, and register values based off those parameters can automatically scale.

Graphical user interface

Description automatically generated with medium confidenceGraphical user interface

Description automatically generated with medium confidence

Fig.10.7. Project organization.

**Lidar ToF research and debugging**

Lastly, I’ve been helping those working on the LIDAR to get it working. To improve timing, I have added the systick driver, and to add debugging functionality, I implemented a USB Uart driver to allow printf to debug their incoming data.

This allowed me to determine that although the primary read function was returning a not-so-accurate reading of distance, using an FIR filter on one of the internal variables to the driver provided a much higher fidelity distance measurement.

Work still remains, but it is a huge jump in functionality from before!

A screenshot of a computer

Description automatically generated with medium confidence

Fig. 10.8. LIDAR internal data being printed. Note the purple trace is much higher fidelity than the returned data (green).

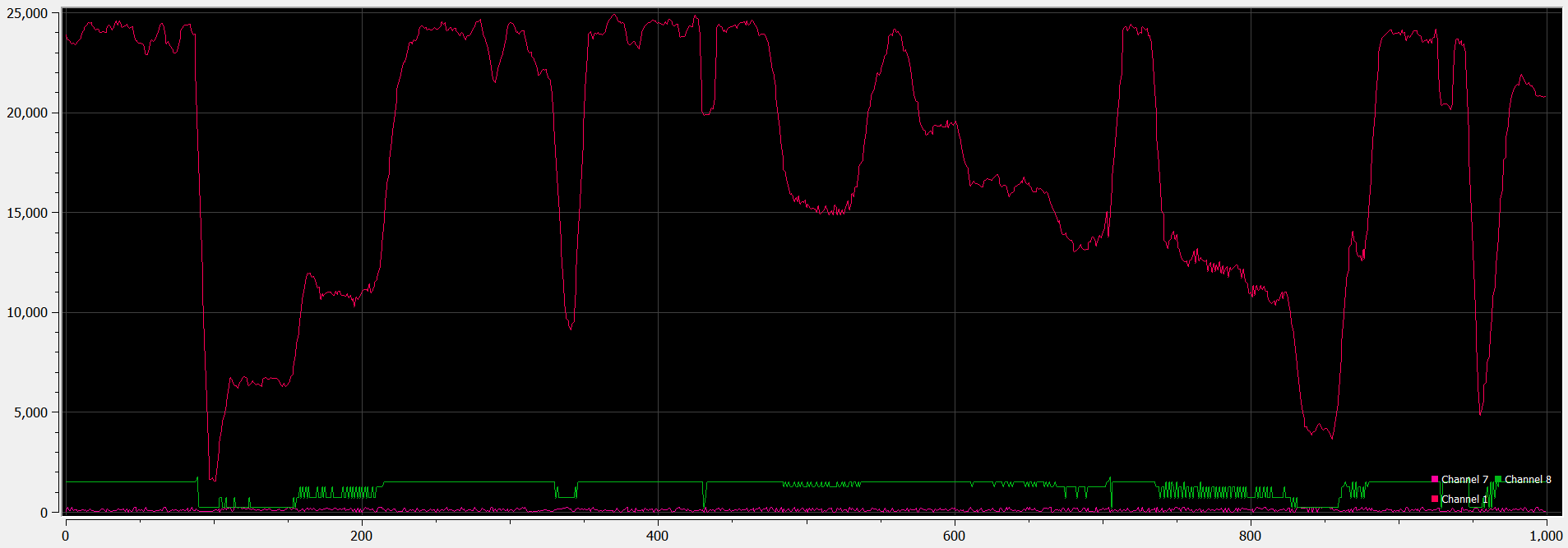


Fig. 10.9. High fidelity data passed through FIR filter. Range measurement is a result of moving my hand back in forth in front of the sensor.