Artur Lukin
Jace Bartley
Bryson Stevens

Milestone 3

When we started working on our Milestone 3 we were curious about the raw data that we were receiving from the accelerometer. The data sheet showed that it is a 12 bit number, but we were only seeing 8 bits. With the help of Bryce we realized that our shifting of the data from the registers was not right and we were losing data. Once the shifting was fixed we were getting numbers that were from 0 to 4096. Later we also realized that there was not data in the negative direction so we looked into it and noticed that the data was stored in a uint16_t instead of an int16_t. Once that was all fixed we finally had real raw data that looked good to us.

Now that the correct data was being read we needed to get it over to the SD card. That proved a challenge, because the SD card has been taking us over 3 weeks to figure out. With every advance forward it presented more problems to think about. One of the major issues is the write speed. We set up a test for the accelerometer to read 2000 times at 100Hz before it stopped reading. Once it reached 2000 the writing thread would finish its current write and stop. When we looked at the data file there was about half as much data written as read. So the writing thread was not being able to keep up with the data so after talking with Thijs he let us know that writing to the SD card in 512 byte blocks will achieve the fastest writing speeds. We also looked online and it confirmed his statement so we figured out a was to go through the data and pack it all into a 512 byte block. The way we stored the data in the text file was allowing 5 bytes for each reading to include the negative sign and two spaces and a new line character meant we had 18 bytes of data per line. We divided 512 by 18 and it gave us 28. It wasn't perfect but based on our needs we would be writing 504 byte block and it was the closest we could get to the 512.

Once that was implemented we read 2000 times at 100 Hz and wrote to the SD card. Then we looked at the data and noticed that there was much more written to the file, but it was around 1750

lines. We were getting closer, but the reading was still faster than the writing. Our next step was to create two buffers where we store the x, y, and z data and once the buffer was full, it would flag that buffer as able to be written and then switch to the other buffer and write data to it. This seemed to have solved our problem, but after a few tests sometimes the data would not finish writing all the way. So then we implemented a third buffer and that seemed to have solved that issue. We wanted to make sure the correct buffers were being written sequentially so instead of reading data we changed the x, y, and z to simply write 1, 2, and 3 respectively. We began testing and noticed that sometimes one of the buffers would be skipped over or a buffer would be written twice in a row. We concluded that there was an issue with the way the buffers were being signaled to write.

Our approach to fixing that problem was to implement a circular buffer that would store which buffer needed to be written and every time it would finish writing one it would check for the next one. At most there would be only 2 entries in the circular buffer, but it solved our issue of the order of the buffers and the data file showed that everything was being written properly. So then we changed back to reading the values and writing the actual data to the SD card. This time rather than using a timer we check the status register on the accelerometer to see if the data is ready. Once we started using that we were getting some garbage data in the first 100 readings. We talked with Thijs and he helped us come up with a solution to see if the accelerometer is awake and the status register shows that data is ready. When all of that was implemented we tested again and there was not any garbage data in our data file.

Our next step was to implement the button so that we would press it when a person would walk over the bridge to confirm it with the data we were getting from the accelerometer. We thought about how we would go about it and realized that it was a simple palSetPadMode for the GPIOA pin 2 and we included it in the read thread and created a new buffer for it to store a 1 when it was pressed and 0 when not. We modified our data that would be written per line to the data file to include the 1 or 0 at the end. We changed our block size to be written to the SD card. When we started testing we noticed that something was not working. After fiddling for a while, we opened the button case and noticed that

one of the solder connections was broken. When we held the connection in place everything seemed to be working. I took the button home with me that night and re-soldered it.

Our next step is to get 24 hours worth of data with the button input and we will be doing that during the lab. Once we have data we will start working on an algorithm to detect a person's movement. With so much work on the SD card this was definitely the hardest Milestone, but we have finally finished it.