Andrew Hoyle
Dylan Dewhurst
Tim Leikam
5/4/14

Assignment #5: Detecting a Fall

Determining a Fall Algorithm

We brainstormed and tested out many different ideas as to how to detect a fall. A couple of our original ideas were to only focus on downward acceleration, or trying to account for a fall by looking at downward acceleration and use the gyroscope to detect orientation. Ultimately, after reading about the subject in numerous research documents, we decided on gauging whether or not a fall occurred based off the normalization of the 3 acceleration axis` and the IMU's final orientation.

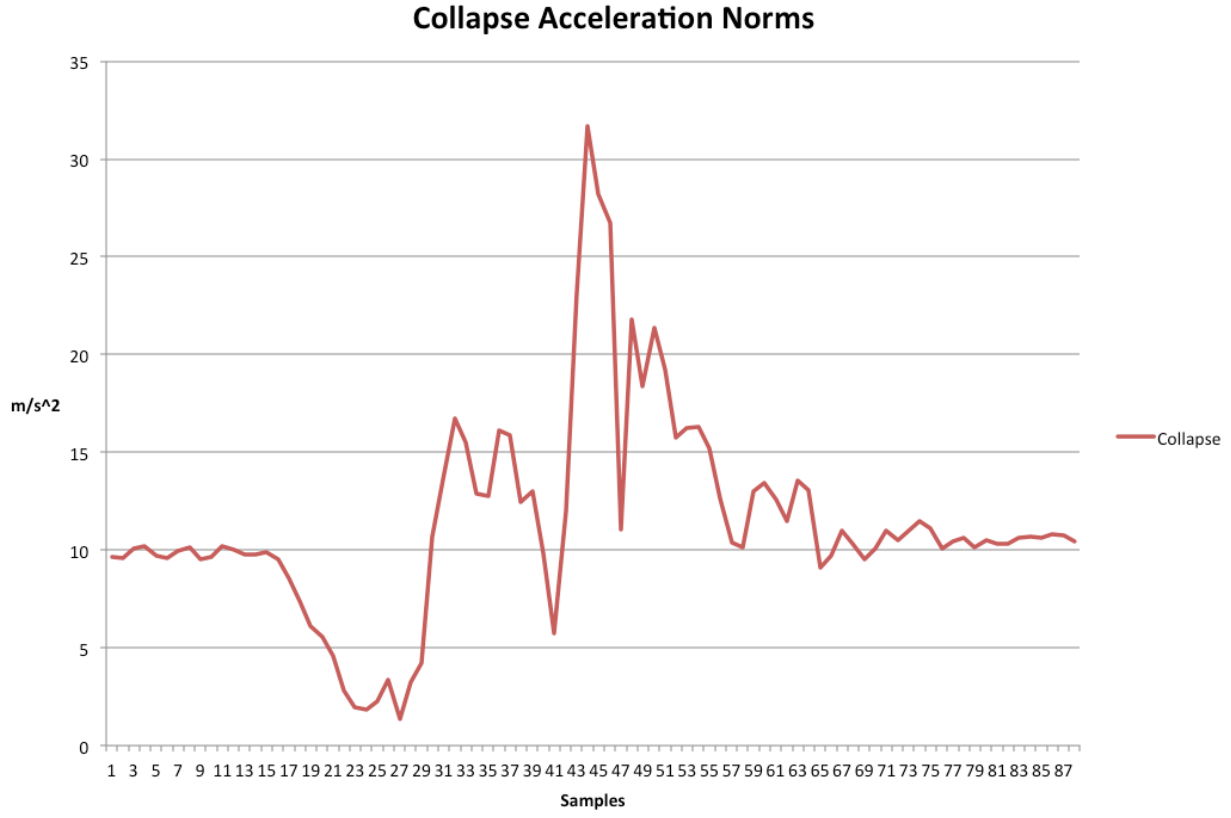
Why Normalization

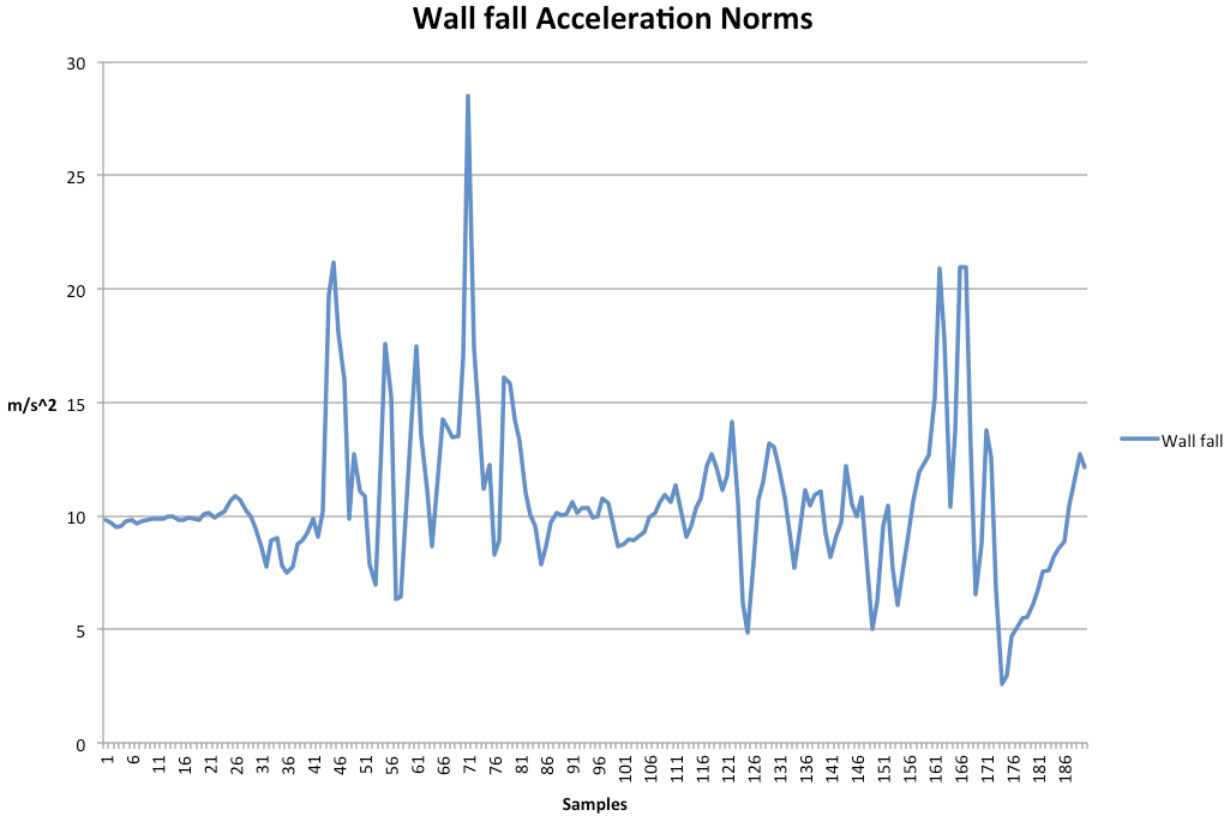
Deciding on the normalization of acceleration was influenced mostly by reading research papers. A common component among them was that they all used this normalization method to help detect falls. Implementing the formula listed below and testing it, we were able to see that there are distinct differences in the values that are registered based on walking, going up/down stairs, sitting down, tripping, and collapsing.

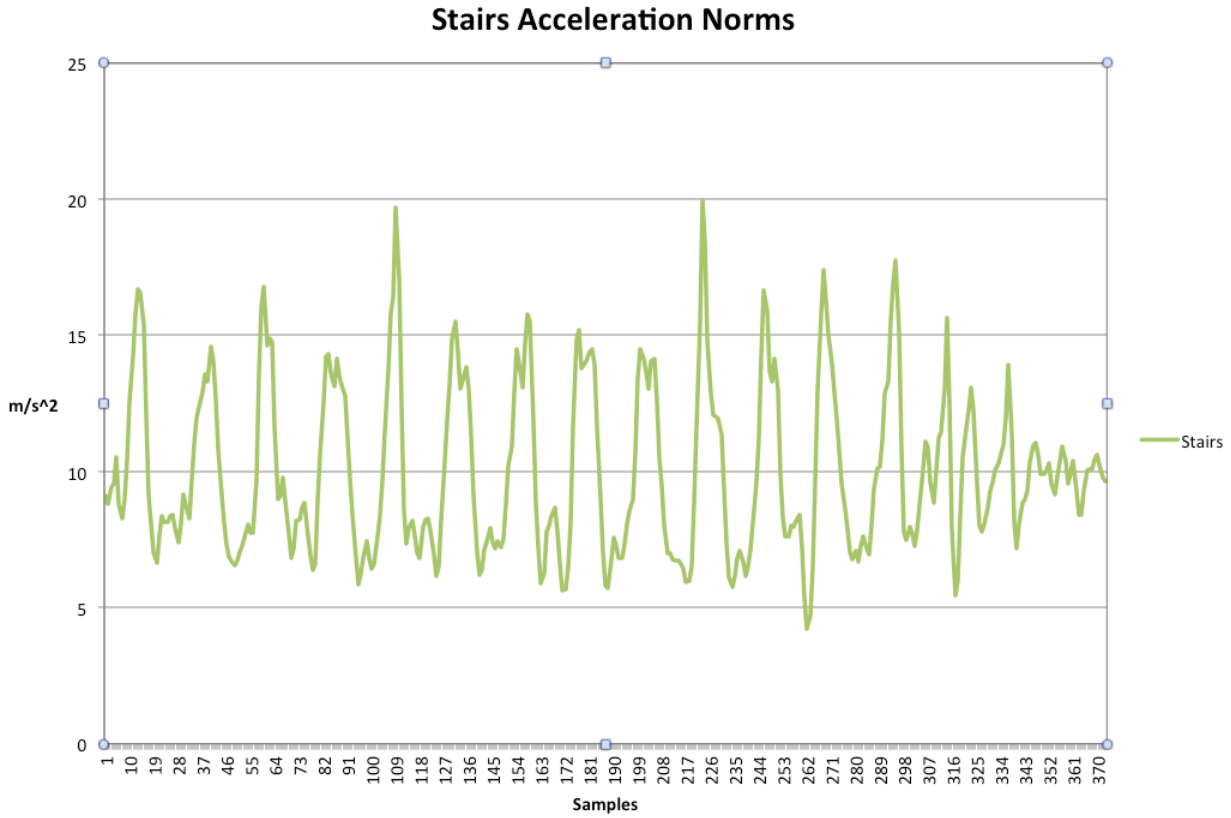
$$Normalization = \sqrt{A_x^2 + A_y^2 + A_z^2}$$

This formula showed (and is shown in our Excel charts) that the mins and maxs that are reached for walking, taking the stairs, and sitting down are much less severe than those that occur when actually falling. After testing, graphing, and seeing these results, we knew that using this data would be vital for us detecting a fall.

In order to use this data, we looked at the common peaks and valleys of a fall and determined a few thresholds that would be reached if someone was potentially falling. If any of our thresholds are reached, we begin a timer that constantly updates what the minimum and maximum values are. During this timer's activation, we look at the difference between our min and max. If this difference is greater than a certain value we determined, chances are then that the person wearing the IMU fell.


Collapse Acceleration Norms
m/s^2
Samples
Collapse


Wall fall Acceleration Norms
m/s^2
Samples
Wall fall



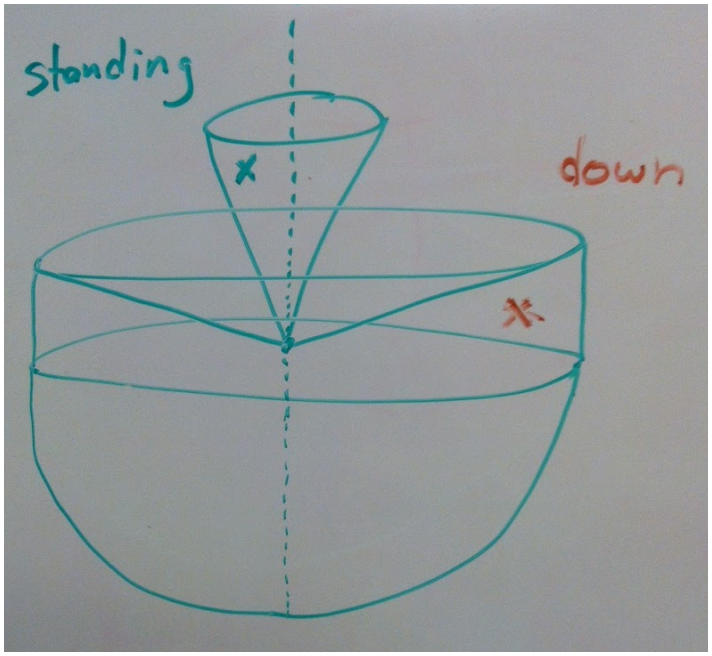
The first threshold to start a timer is activated when the acceleration norm is outside of the stair walking amplitude. This was found to be about anything outside of [4, 20] m/s^2. The amplitude of a fall was calibrated to be about 27 m/s^s.

The calibrated output from the MPU was converted to m/s^s by taking putting one of the acceleration axes parallel to gravity and observing the maximum value. The known gravity value in m/s^2 was then divided by this voltage value to get a conversion. (This was 0.00060535f.)

Why Orientation

After seeing and interpreting the data we gathered from tests with normalization, we determined that this alone cannot detect whether a fall occurred. In order to really see if a person fell, we felt that knowing the person's final orientation would be vital in determining if they fell.

In order to determine their orientation, Andrew came up with a unique solution for determining if a person is "standing" or if they are "down", which is visually shown below.

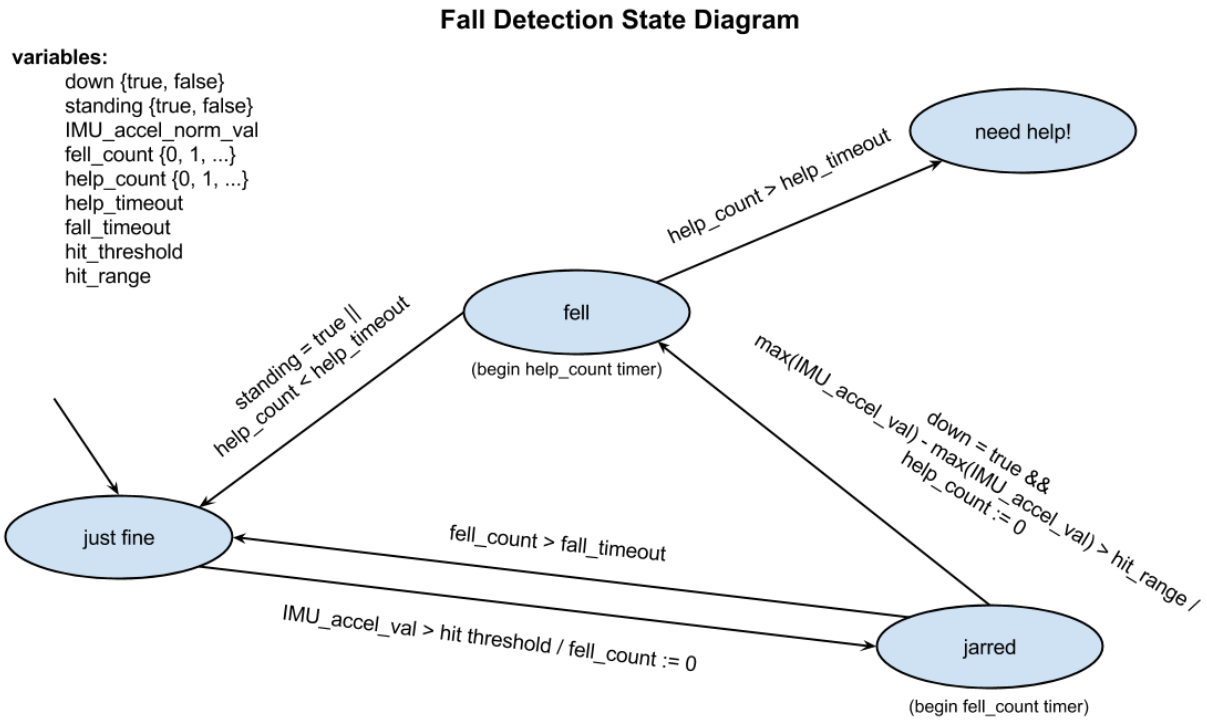


In this model, the inner cone represents the degree range where we figured a person could normally be standing. The flatter bottom cylinder is the range in which we thought a person could be considered as “down” (i.e. bent over, laying down, etc). The lower sphere half also is considered “down.” The blank area between the cones is a range where a person isn’t “standing”, but they’re not necessarily “down” either. If a person is in a “down” position at the end of the timer mentioned previously, then chances are they fell.

Detecting a Fall

If the person wearing the IMU is “down” and if the normalization difference is greater than a certain value after the timer runs out, then we have decided that the person has fallen and we report it. This method also takes into account that a person may have appeared to have fallen, but recovered, thus not giving a false positive.

In addition, we also went a step further and determine whether or not a person fell and is seriously hurt. To determine this, after deciding that a person fell and reporting it, we then wait an additional amount of time to see if the person has reached a standing orientation or if they are still in a down state. If they are still in the down state after our new timer has ended, then they are potentially seriously injured and need assistance.



A state diagram has been created to demonstrate the various timers and variables accounted for in the fall detection system. The output of the *fall_detection_answer.ino* file from the serial connection shows the current state the sensor is in. The current values are printed to the screen every time the MPU gets updated information. Each important state or variable is printed in a column. The first column is the acceleration norm value, the second is the gyroscope booleans “down” and “standing” if either are true. The subsequent columns show the states and timers: “fell timer” and then the “help timer.” If the user has reached the “need help!” state, that is printed on the far right.

(After 10 seconds or so in the help state, the device resets so another fall can be detected. This final state would include calling the person to see if they’re okay, and then sending an ambulance if there is no phone response.)

Assumptions Made

Together, we decided as a group that this would be a device used by the elderly on their chest in a specific orientation. With this assumption in mind, we made a few more assumptions:

- The person would not take the stairs or walk too quickly (i.e. no running, jogging, etc).
- The person would not be able to fall and remain in a “standing” position by the end of it.
- This would be worn next to the sternum, for example clipped to the shirt and inside a shirt pocket.
- The IMU would be oriented with the y-axis facing up as demonstrated in the video.

Problems Encountered and How They Were Handled

- Figuring out what algorithm to use to detect a fall
    - Decided to read research papers and see what methods they used. Ultimately deciding to go with normalization of acceleration and a person's final orientation.
- Finding out a place to meet at
    - Trying to find a central point that worked for all of us at the same time was difficult. We decided to meet at the North Classroom and use an empty room there.
- The UART to USB device was not working properly. An Arduino Nano was used until a new converter could be obtained.
- Modeling a fall with an extended state diagram was a challenge, but it was instrumental in understanding a problem conceptually before coding.

Additional Files

More files in the *fall_code* folder have been included. These show our previous attempts to test and make different parts of the fall detection system. The *imu_sample_rate_measurement.ino* file is our group's attempt to instrument the code to see how much time passed between the MPU.ready() calls and the main loop(). This value averaged to about 22 ms. (This just gave us a ball-park measurement to make sure that the sample rate was sufficient for recording fall data.)

The *fall_detection_angular_vel_averages.ino* file also shows how we were able to extract an angular velocity average. The differences in angles were placed into a circular array and an average was taken at different intervals. It was determined that angular velocity was not needed for our fall detection solution, but it was an interesting value to extract. (The averaging calculations were also quite expensive if the array size was too large.)

Who Did What

We all gathered together and figured out all the details of a falling algorithm to use. We all helped with troubleshooting code, making this report, and making images/charts. Andrew made the demonstration video since he was the only person with a bluetooth computer. Dylan was the test subject for all jogging, walking, and falling.