Rushil Master ID: 20476854

Lab Report

Part 1: Lab Testing

For lab 1, the testing consisted of working with one sensor at a time. The first sensor tested was the light sensor. For this, we placed our finger on the light sensor and checked whether the value of the sensor decreased. Accelerometer was followed by light sensor. To test the accelerometer, we moved the phone along each axis, and checked whether the corresponding values for each direction changed. Rotation vector was tested in a similar fashion. To test the magnetic field sensor, we introduced the phone to an area with a great magnetic field, such as near a speaker. Maximum values recorded were tested by checking whether the value changed when the current sensor value exceeded the current maximum value. The functionality of the graph was tested by checking the pattern of the lines. The tests conducted were very effective. It resulted in an app which functioned the way it was supposed to.

The approach to test lab 2 was much different compared to lab 1. To test it, the method of trial and error was used. Every time we changed the values within the state machine, we tested the app for accuracy. We kept changing the values used in the state machine according to the amount of steps counted. If the app counted less steps compared to how many were actually taken, we made the range of values greater and if the app counted more steps, then we decreased the range of values. This was done many times, until finally the steps counted were within the desired accuracy. In addition to trial and error, we also took advantage of using the log cat for testing. We used log to output messages at different points in the code to help us analyze which parts of the code were functioning correctly and which parts needed changes. The tests conducted for lab 2 were not the most effective as it physically required a lot of walking every time the code was changed.

Lab 3 was tested in the exact same fashion as lab 2. The only difference was having to test the compass reading and testing displacement in each direction. To check the compass reading, we simply compared the value shown in relation to the physical world magnetic poles. Displacement for north/south and east/west were tested by walking in a particular direction and checking if the resultant values are correct. To further test it, we walked in a circular path and checked if the displacement shown was zero at the end of the path. The tests done were very effective as it helped to perfect the app functionality and gave us an insight into how to approach lab 4.

In lab 4, the testing was checking whether or not a path free of obstacles was shown for every single combination of origin and destination selected. The second test performed was checking if the user location updated correctly. Initially, the user location was updating according to the true magnetic pole. To fix this, we shifted the axis of the compass by twenty degrees so that when the user walks in the direction of the map north, the user location on the app also moves towards the map north. The last test conducted for lab 4 was to check if the path to the destination changed if the user walked in the wrong direction. This was done by walking ninety degrees from the path shown. To simplify the testing, we also created different buttons which simulated a step in the desired direction. This made the testing very effective because it made it possible to test the app from a single location and it did not matter if that location was heavily influenced by a magnetic field since the steps were simulated manually.

Part 2: JUnit Tests

I do not think it would be beneficial to add JUnit tests as a requirement of the labs simply because they are not necessary for what the purposes of the lab are. The labs can be completed just fine without them. If they were made as a requirement of the labs, it would end up causing more confusion for the students which may result into failures. Also, the labs are already time consuming as they are right now and if JUnit tests were to be added, it would end up consuming even more time. It is possible to make JUnit tests as bonus, so for those who want to go beyond the scope of the labs can earn extra marks towards something which they may have lost marks for.

Part 3: Design Decisions

One major design decision we made in lab 2 was the idea of implementing a state machine and determining how many states to use. We analyzed the pattern shown upon taking a step. Figure 1 shows the pattern and is labeled to describe the pattern:

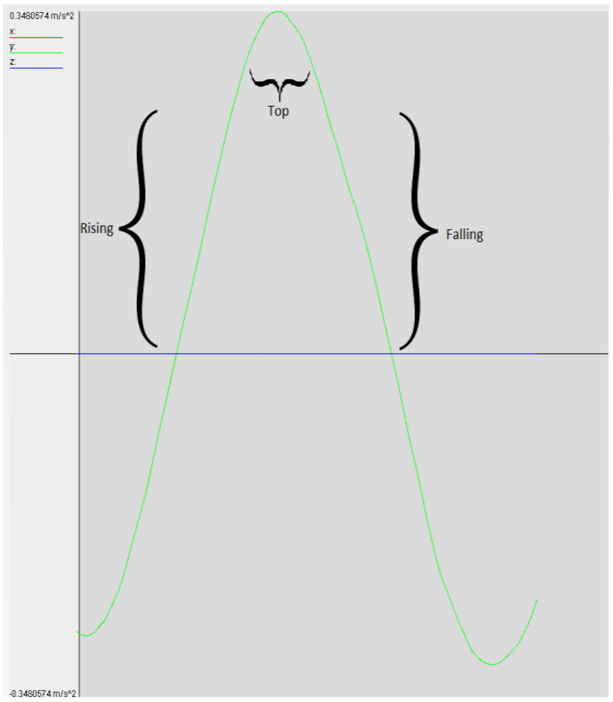

Figure 1

After examining this pattern, we chose to implement the state machine with three states and using the values from the y-axis. The three states were rising, top, and falling. It was an immediate realization that these states must be executed one after another in order to determine what counts as a step. The step counter should increase by one only if rising is followed by top and top is followed by falling. The problem we faced was determining the range of values needed to go from one state to another. For this, we did a lot of trial and error until we got the right values. An alternative we considered was to collect sample data, plot the data in excel and obtain the values by analyzing the critical points in the plotted graph. However, this would have required us to write more code than necessary. It would have forced us to use file input/output, store data within arrays, and printing them out in a CSV file. In the end, we made the right decision of not doing that. The method of trial and error was physically exhaustive but it served a great purpose to make the lab run with the same accuracy as we would have got if we had decided to go with the alternate method. One improvement which could be made to this lab is if we had introduced more than three states. With the current state machine, it is very easy to imitate a step by shaking the phone. With more states, we could have made the app such that imitating a step would be very hard.

Part 4: Lab 4 Improvement

One improvement which could be made to our existing lab 4 is within the algorithm to find a path free of walls. Currently, the code has been hardcoded to find the path within Lab-room-peninsula. We have four fixed points in the map located at (3,9), (7,9), (11,9), and (16,9) as shown in figure 2.

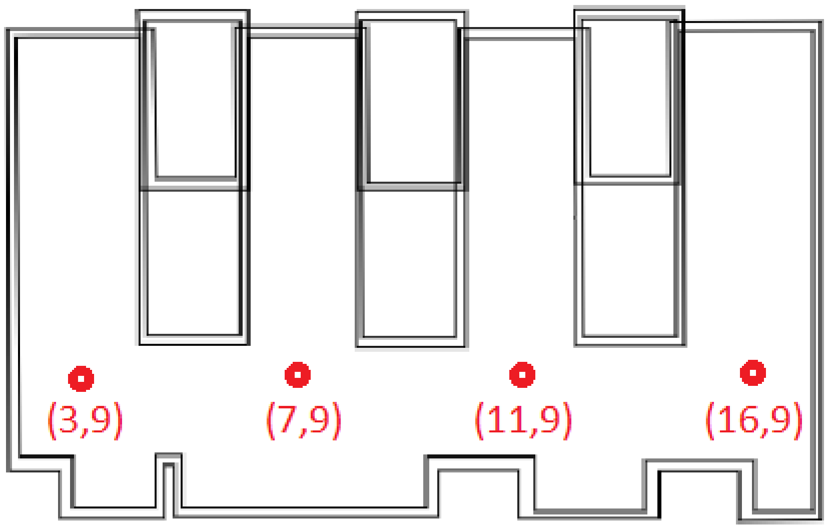

Figure 2

Using these four points, we wrote the code such that it tests for every single combination of origin and destination selected. It is approximately a 180 lines of nested if statements which calculate a path. As an improvement, we could have implemented Dijkstra's algorithm for shortest path finding. This method would have made the code much shorter and made it better since it would find the shortest path without running into obstacles. Another small improvement we wanted to make was to ask the user for their approximate step length in meters before they start walking. Currently, the program assumes that every step the user takes is 0.6 meters. By making the step length a variable, it would have made the program much more accurate.