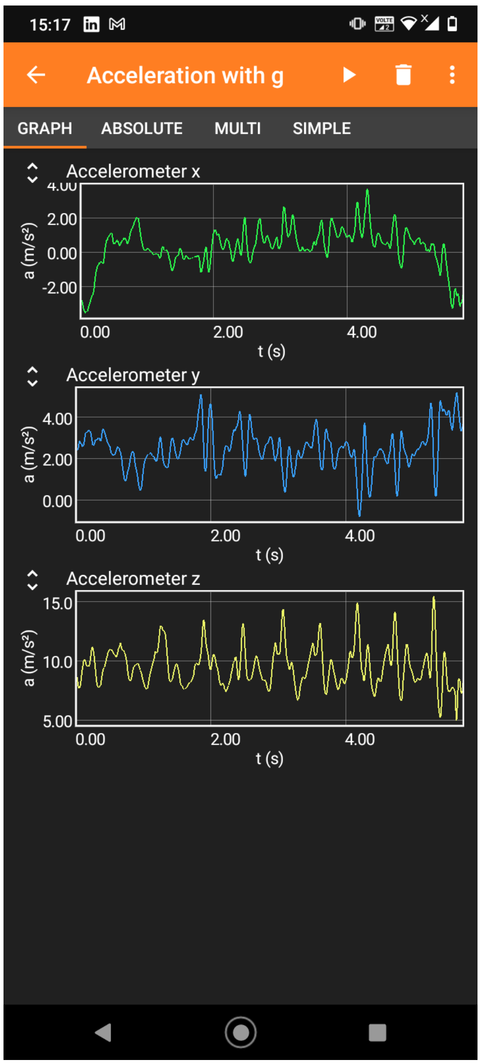
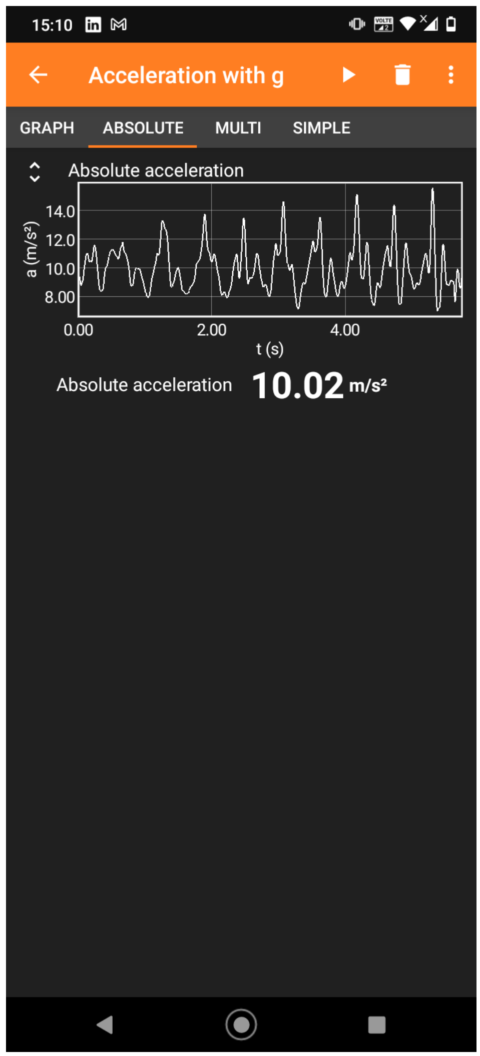
<u>Assignment 2: Build a Pedometer</u>

Akash Nair (asn7bf)
Kalyansundharam Kathiresan (kkl2rw)

Q1.Choose the sensor

Ans: to explore the data we used Phyphox to visualize the data. We noticed that the accelerometer always gave a jerk when the foot landed on the ground. We could use this to calculate the number of steps taken. We wanted the sensor to work no matter what the orientation of the device was, so we chose to use the magnitude of the total acceleration of the device to count the number of steps taken. The time between 2 footsteps left and then right for example with a brisk walk is on average 400 milliseconds, so we think that a sampling rate of 1/0.4 = 2.5 Hz. The accelerometer samples at a rate far higher than that rate.





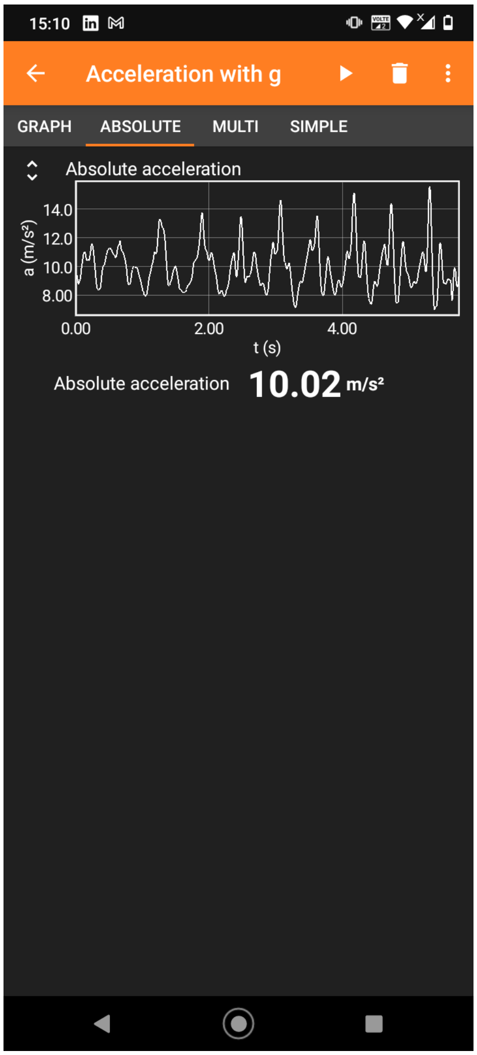
Q2.Choose the location to place the sensor?

Ans: We chose to keep the sensor at chest level, and use that to develop the App. The sensor even works if placed in the pocket but with lower accuracy.

Looking at the Phyphox data and given that we have some experience developing mobile apps in Android we decided to develop a mobile app to implement the pedometer.

Q3.Observe the signal

Ans: we see that the accelerometer reading has a peak every time the foot hits the ground as the whole device shakes subtly with each step. We can use the number of peaks as an indicator of every time the foot strikes the ground and can be used to count each step. We also saw that the input was a bit noisy. For preprocessing we chose to use a low pass filter to filter out the noisy input due to vibration.



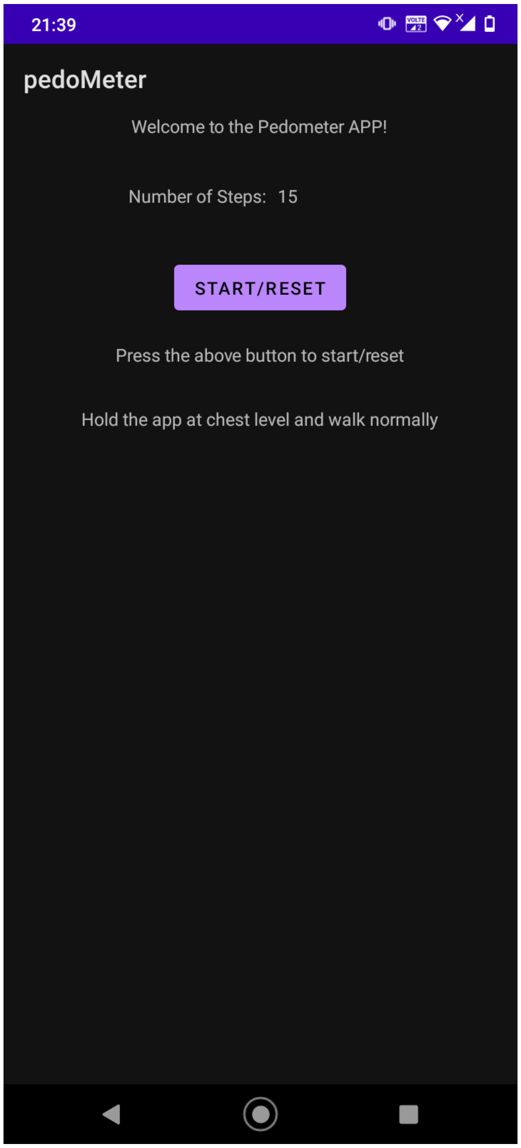
Q4. Choose the feature

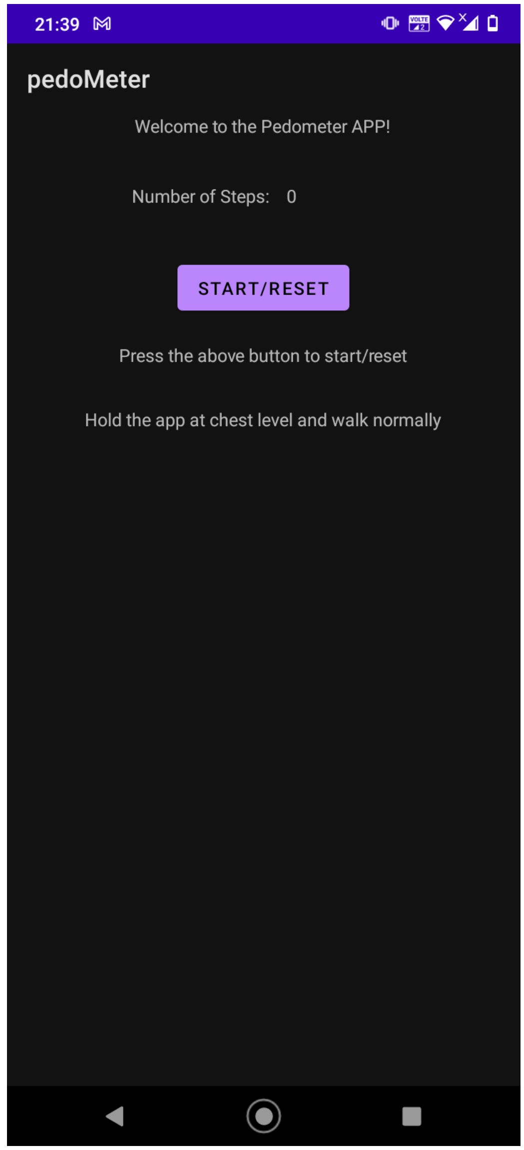
Ans: First, we took the accelerometer data and found the absolute magnitude of the acceleration. We passed this value through a low pass filter to remove the noise. This value will always be around 9.8 m/s^2 due to gravity. We saw if it increases past a threshold. If it does we then calculate the peak as a count. We didn't want other noise data to register as a step so ignored any more peaks for around 400 milliseconds.

Q5. Implement the step counting algorithm

```java
//get the sensor data
        float x = sensorEvent.values[0];
        float y = sensorEvent.values[1];
        float z = sensorEvent.values[2];
        //get the absolute magnitude of the force on the device
        double currentAccelration = Math.sqrt(x*x + y*y + z*z);

        //low pass filter to remove the high frequency noise
        double lowPassAcceleration = lowPassLoop(currentAccelration);
        /*
        * CHECK IF THE ACCELERATION/FORCE ON THE DEVICE IS GREATER THAN THE
THRESHOLD SET
        * IT IS NOT POSSIBLE TO TAKE A STEP IN LESS THAN THE THRESHOLD SET
IGNORING OTHER JERKS DURING THIS PERIOD, THE TIME THRESHOLD CAN BE LOWERED
FOR A RUNNING MODE.
        * */
        if(lowPassAcceleration > GRAVITY + THRESHOLD &&
System.currentTimeMillis()-lastCountUpdate>TIME_BETWEEN_STEPS_MILLISECONDS)
{
            numberOfSteps++;
            lowPassAcceleration = 0;
            lastCountUpdate = System.currentTimeMillis();
        }
        //SET THE VALUE IN THE TEXT BOX
        op.setText(numberOfSteps +"");
```





Q6. Test if it correctly measures steps

Kalyan took 82 steps but the device caught around 78. with an error of (82-78)/82 = 4.878 with an accuracy of 95.122.

Akash walked for 75 steps but the device calculated around 73. With an error of around (2)/75 = 2.66 % with an accuracy of around 97.33%.
Average accuracy: 96.226%.

Conclusions:
We were able to create a step counter that was able to measure the number of steps quite accurately, by looking at the number of 'jerks' seen by the accelerometer.

Work done was 50-50.