Assignment 2: Build a Pedometer

CS 6501 Engineering Interactive Technologies

Adil Rahman
University of Virginia
Virginia, USA
ar9fb@virginia.edu

Peter Solomon
University of Virginia
Virginia, USA
ps3gj@virginia.edu

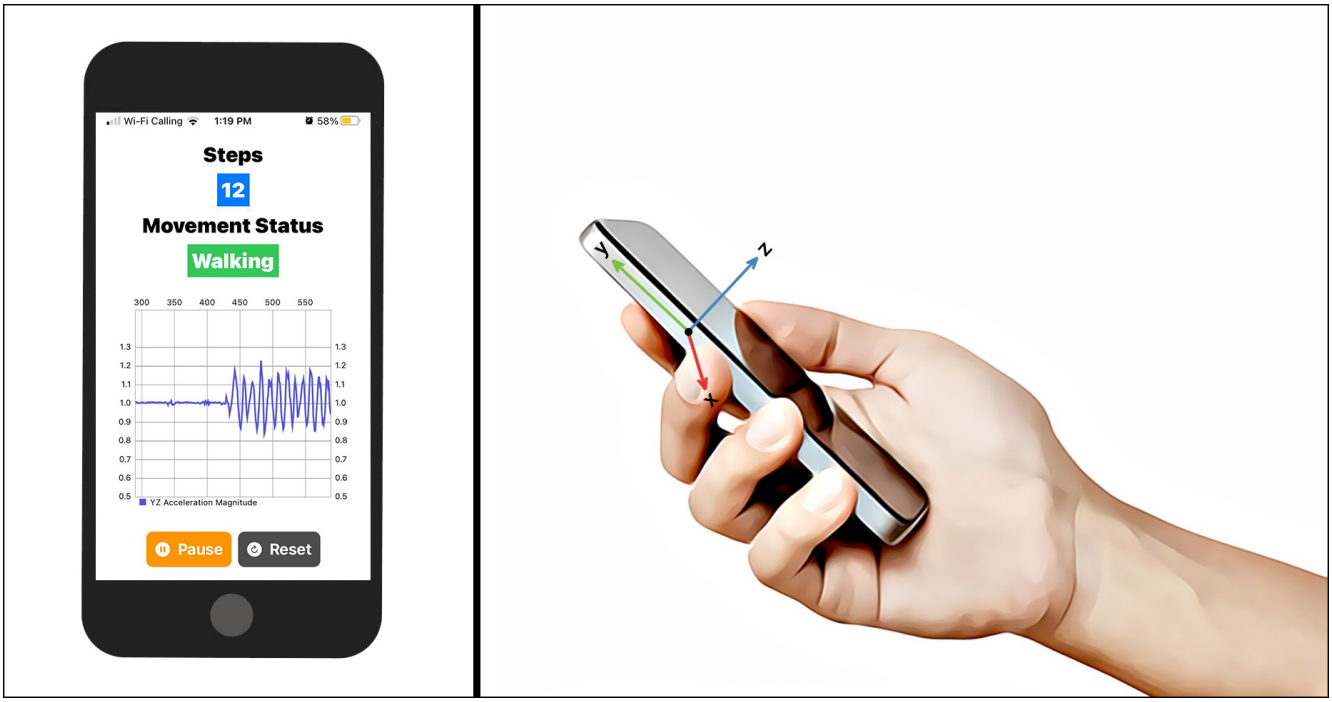

Figure 1: StepUp - The Pedometer Application [Left: User Interface, Right: Holding Posture]

SYSTEM OVERVIEW

We present *StepUp*, an iOS pedometer application which is capable of not only counting steps, but is also able to differentiate between walking, running, jumping and idle stance of the user. The application relies on the accelerometer readings of the device to correctly estimate the step and classify motion. On the user interface, this application displays the total steps walked, type of movement, and the instantaneous acceleration graph. This application also allows for ad hoc pausing and resuming of the pedometer monitoring system and resetting the step counter.

IMPLEMENTATION

We break down our implementation into four sections: *Preferred Prototype Location*, where we talk about the preferred location where our system would be placed for effective functioning and user experience, *Formative Study*, where we talk about the experiments we performed to determine the features necessary for designing our algorithm, *Algorithm*, where we talk about the functioning and workflow of our system, and *Hardware and Software*, where we talk about the hardware, programming language, framework, and libraries used in building our system.

Preferred Prototype Location

Before beginning the implementation, we first decided the preferred location on a user's body where we would like to deploy the pedometer prototype. We intended to create a system which could detect the motion at run-time and immediately give feedback to the user as compared to systems which analyze the data offline. Thus, we decided that the ideal use-location for the deployment of our prototype would be on the user's hand, with the prototype's display facing towards the user to allow the user to receive immediate feedback. Our implemented prototype's holding posture is illustrated in Figure 1 (Right).

Formative Study

We first started using the *Phyphox*[1] smartphone application to get an idea of the sensor readings as we tested different mobile device

[1]https://phyphox.org

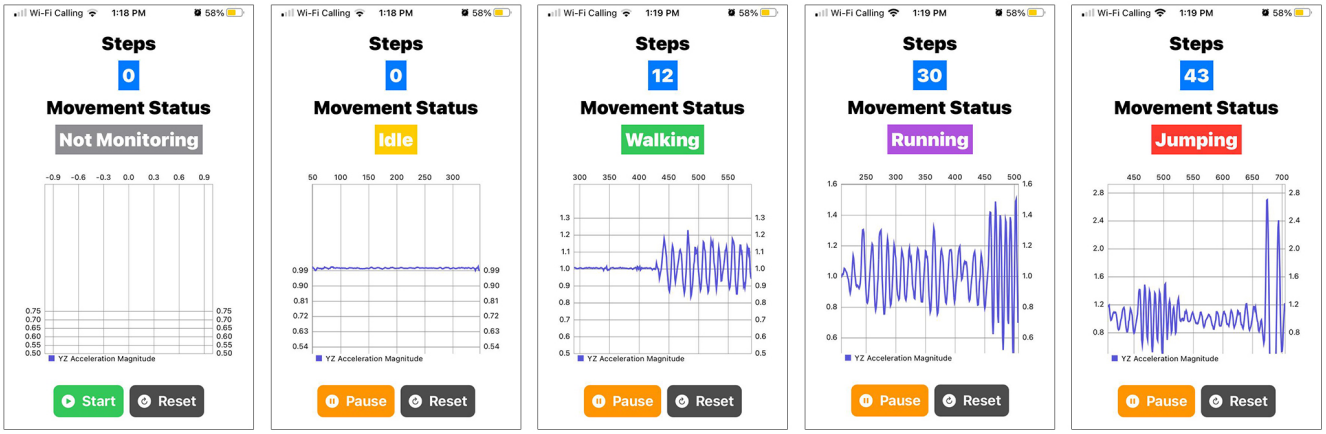

Figure 2: StepUp UI Screenshots [Left to Right: Not Monitoring, Idle, Walking, Running, Jumping]

orientation and observed the signal patterns as we moved. For this experiment, we chose to hold the smartphone on our hand with its screen facing towards our face, as delineated in the previous section, so that we could instantly observe the signals as they were recorded. Based on our observations, we decided to use the accelerometer sensor for our purpose since we could observe clear crests and troughs on the accelerometer signal upon walking. Furthermore, we also noticed that the amplitude of these crests and troughs increased when we shifted from walking to running and jumping, with jumping having the highest amplitude values. We concluded that these crests and troughs were created due to the jerk motion produced when we placed our feet on the ground, with the amplitude of these signals being directly proportional to how hard we placed our feet on the ground. This conclusion was also supported by the fact that when we walk, our body experiences the least amount of jerk when our feet falls on the ground, whereas when we run or jump, we experience a greater jerk impact.

Algorithm

Accelerometer Readings.
Based on our formative study, we used a simple, thresholding-based algorithm for building our pedometer. Since we want to hold our device in our hands with its screen facing towards our face (Figure 1 Right), we realized that the only motion relevant for capturing the movement are the Y and Z axes since the user moves forward (Y-axis) and the user may bob up and down (Z-axis) due to the jerk impact produced upon landing the feet on the ground. The X-axis of the device is only relevant on capturing sideways motion. However, since we are not concerned with side motions, we choose to entirely discard the X-axis, which in-turn, significantly reduces the noise in the input of our algorithm.

Input Preprocessing.
In order to get the aggregate magnitude (*aggAcc*) of the Y-Z component of the accelerometer reading, we performed the following computation:

$$aggAcc = \sqrt{accY^2 + accZ^2}$$

where *accY* and *accZ* are the Y and Z components of the accelerometer readings, respectively. After obtaining *aggAcc*, we computed a 5-sample running average to reduce noise and smoothen the data. This averaged data is hereon referred to as *avgAggAcc*.

Thresholding-Based Algorithm.
Based on our observations from the formative study, we realized that there exists discrete values which could be used as thresholds for determining and differentiating between walking, running, and jumping motion. After further experimentation with *avgAggAcc*, we empirically estimated the following thresholds for determining the nature of motion:

Idle : 1.0
Walking : 1.1
Running : 1.4
Jumping : 2.0

Based on the values above, we designed a thresholding-based algorithm such that as soon as the *avgAggAcc* value crosses any given threshold value, the motion corresponding to that threshold is classified as the user's motion. Another inference that can be made from this algorithm is that every time the *avgAggAcc* value crosses 1.1, the event is counted as 1 step. Moreover, in order to prevent the system from going on recording steps as long as the *avgAggAcc* value remains above the given threshold, we use a flag variable to control the step addition which allows only one step addition after the threshold is crossed, and gets reset only when the *avgAggAcc* value falls below 1, which happens after every step since a step's signal consists of a clear crest and trough. Moreover, the signal generated upon jumping once contains three crests and troughs, the first two being of extremely high amplitude (the user's initial jump and landing), and the final one of moderately high amplitude (the user restoring to original stance after landing). This movement, however, incorrectly accounts to 3 steps. To counter this, every time a jump signal is detected, 2 steps are deducted from the total step count, and thus as a result only 1 step is added as a result of jump. We refer to this as the *jumpPenalty*. Finally, we define a *moveRetentionThreshold* which is responsible for retaining a move

instead of immediately changing it to something else. Again, based on observation, we empirically set $moveRetentionThreshold$ to 40, i.e., two-third of a second (since we are sampling 60 values per second and the move is retained for up to 40 values). This prevents the system from displaying *Idle* at every step since the signal value returns to 0 upon every step and thus allows for a more noise-free experience. The signal generated for each movement type is illustrated in Figure 2.

Hardware and Software

Since testing a pedometer requires a lot of physical movement, prototyping the system on an Arduino board with an IMU module attached to it would be problematic given the immediate lack of portable power source, rigid and graspable form factor, and exposed wires which could potentially result in components getting disconnected. For a more elegant and compact build solution, we decided to use the on-board sensors present on smartphones to create the pedometer system. Doing so would eliminate all the aforementioned caveats of prototyping on an Arduino board. We designed our pedometer application *StepUp* for iOS devices ($\geq$ 14.0) using *Swift* as our programming language. *Xcode* (v12.4) was our choice of IDE, whereas the framework used for building the UI was *SwiftUI*. Our algorithm relies on the device's accelerometer readings (sampled empirically at 60Hz), which we obtained from Apple's *CoreMotion* framework. *Charts*[2] was used for designing the real-time graphs. All tests were done on an iPhone SE (First Generation) and an iPhone 12 Pro. The user interface of our system is illustrated in Figure 1 (Left) and 2.

SYSTEM EXPERIMENTATION

Experimental Setup

To evaluate the accuracy and effectiveness of our system, we performed the following tests:

(1) **Quantitative Evaluation**: To measure the accuracy of the pedometer, we (both the authors of this report) walk 30 steps with the device on our hand, and then measure the estimated number of steps. Based on this data, we apply the following formula to obtain the error rate (%) and total accuracy (%):

$$ErrorRate(\%) = \frac{|TotalSteps - TotalEstimatedSteps|}{TotalSteps} \times 100$$

$$Accuracy(\%) = 100 - ErrorRate$$

(2) **Qualitative Evaluation**: We perform a qualitative evaluation to observe the performance of our system's movement detection. For this experiment, three participants (the authors of this report, and the roommate of one of the authors) perform all the supported movement gestures - *Idle*, *Walking*, *Running*, and *Jumping* to observe whether the system is able to classify those movements correctly or not. While the total number of participants is less, on a larger scale, this experiment could potentially establish the validity of our threshold values.

[2]https://github.com/danielgindi/Charts

Results

The results from our experimentation are as follows:

(1) **Quantitative Evaluation**: The data obtained from the two participants are delineated in Table 1.

Table 1: Quantitative Evaluation Results

Participant	Total Steps	Estimated Steps	Error Rate (%)	Accuracy (%)
P1	30	30	0.00	100.00
P2	32	32	0.00	100.00
Aggregate	**62**	**62**	**0.00**	**100.00**

(2) **Qualitative Evaluation**: All of the three participants movement was correctly determined by the system.

BUILDING THE SYSTEM

Source Code.
The source code of *StepUp* can be downloaded from the *GitHub* repository[3].

Prerequisites.
In order to compile *StepUp* and run it on a local system, the following prerequisites must be met:

- MacOS 10.15.4 or later
- Xcode 12.4
- iPhone (iOS $\geq$ 14.0)

Building Instruction.
Once the local system has all the prerequisites met, follow these instructions to compile the source code:

(1) Clone *StepUp* repository to the local system.
(2) Open `/path/to/stepup/StepUp.xcodeproj` on Xcode.
(3) Ensure your Xcode has signing capabilities.
(4) Plug in a compatible iPhone and press the `Build` button.

If the instructions are followed correctly, *StepUp* should be installed in the connected iPhone.

LIMITATIONS

While our system is able to give a decent recognition accuracy for the number of steps walked and the type of movement among *Walking*, *Running*, and *Jumping*, almost instantaneously at run-time, our system is not free from inaccuracies induced by noise. Since we employ a thresholding-based step and movement detection algorithm, there is no check for countering other conflicting body gestures, i.e., just moving the hand while holding the phone can trigger the system to classify the event as walking, running, or jumping. This issue could potentially be solved by using advanced machine learning algorithms to create a model capable of recognizing movement as compared to other conflicting signals. Moreover, using other sensors like magnetometer, gyroscope, etc., can also prove to be helpful. However, with respect to our project, we limit the scope of our system to recognizing the aforementioned motion when the user *actually* intends to perform those motion.

[3]https://github.com/adildsw/stepUp