ECSE 682: Assignment 2 - Embedded Development

Oct 19th, 2020

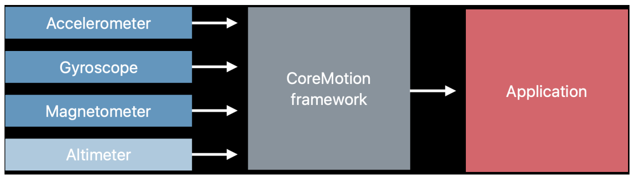
Mian Hamza - 260655263
Jingxu Hu - 260606017

Professor: Zeljko Zilic

1 Introduction

Apple has made various frameworks, and built many sensors into their devices that have a wide variety of applications. One such framework is the Core Motion framework that deals with 'motion- and environment-related data from the onboard hardware of iOS devices'. The Core Motion framework contains libraries that deal with: Process accelerometer, gyroscope, pedometer, and environment-related events. The Core Motion framework has many embedded system applications in health and fitness.[3,9]

Figure 1: Basic Core Motion Framework (Also has other functions like Pedometer)



[9]

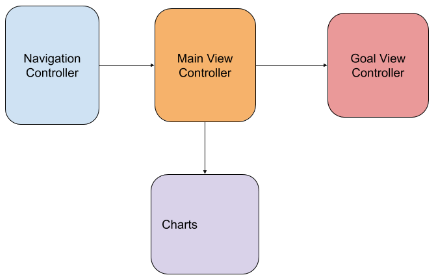
goal of this project was to develop an application that takes Pedometer data from the device and uses it to record the number of steps taken by the user, as well as setting up a goal tracker for the user, in order to ensure that the user can track their fitness progress. The application was developed on an iOS platform by using the Core Motion framework, and especially two libraries:

1. CMPedometer [1,5,6]
   - Fetches live system-generated walking data.
   - Can retrieve step counts, distance traveled and the number of floors ascended or descended.
2. CMMotionActivityManager [4,5]
   - Manages access to motion data that is stored by the iOS device.
   - Can tell whether the user is walking, running, in a vehicle, or stationary. i.e. The state of movement they are in.

2 Architecture

The application developed consists of three View Controllers:

Figure 2: Flowchart of App's View Controller architecture



1. Main VC
    - This is a command center where we can navigate to the other 2 VCs.
    - Displays all of the Pedometer data of the users, including step count and distance travelled.
    - Also displays the goal that the user inputted.
2. Goal VC
    - An input for the user to put their step count goals into.
    - This allows the user to track their fitness.
3. Chart
    - This View Controller displays a graph of the collected Pedometer data.
    - The step data is displayed in a line chart
    - Charts is a library that is installed using CocoaPods, it is an external library that is created and maintained by outside developers.[7]

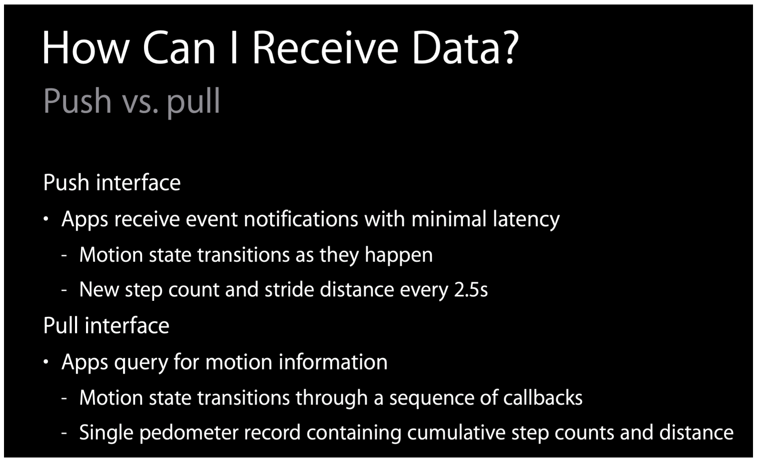
All of the View Controllers are embedded into a Navigation Controller. This allows for a connection between the View Controllers and allows us to easily navigate between Views. In order to continue fetching the Pedometer data, the CMPedometer and Charts functionality are concurrently programmed to run on a DispatchQueue. The functionality of a DispatchQueue is highlighted below:

- DispatchQueue:
    1. FIFO queues to where the application can submit tasks in the form of block objects
    2. Can execute tasks either serially or concurrently.
    3. Work submitted executes on a pool of threads managed by the system.
    4. Can function Synchronously or Asynchronously

The applications DispatchQueue is run in the main thread Asynchronously. This means that the thread will serially execute on the main thread but due to the task being asynchronous the tasks will be sent to a background thread to execute and pass data back to the main thread. In our application the CMPedometer and Charts tasks will run on a background thread. This means the step count, distance travelled, and Chart data points will be managed on a background thread and this will return the data back to the main thread. The main thread will pass all of the data back to the View Controllers for further processing and displaying the results. This ensures that the user can continue using the app and interacting with the User Interface while the system processes are running in the background.[2]

3 Core Motion Framework

The below figures give some insight as to how live pedometer information is gathered and what performance metrics it uses.



How Can I Receive Data?

Push vs. pull

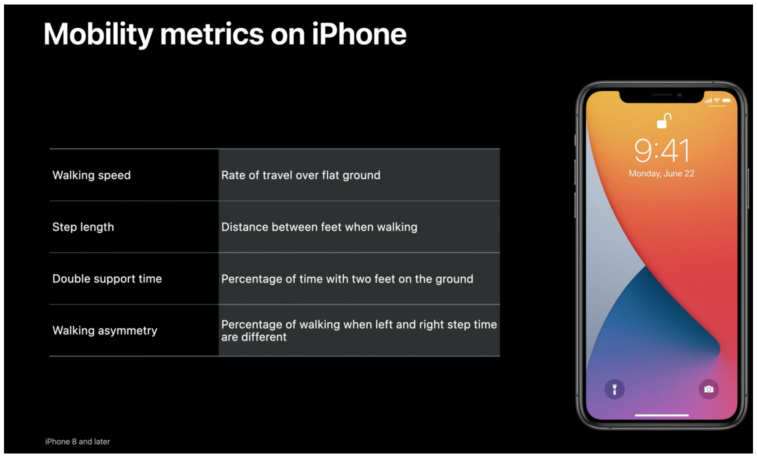
Push interface

- Apps receive event notifications with minimal latency
  - Motion state transitions as they happen
  - New step count and stride distance every 2.5s

Pull interface

- Apps query for motion information
  - Motion state transitions through a sequence of callbacks
  - Single pedometer record containing cumulative step counts and distance

Figure 3: CMPedometer push/pull interface
[6]



Mobility metrics on iPhone

Walking speed	Rate of travel over flat ground
Step length	Distance between feet when walking
Double support time	Percentage of time with two feet on the ground
Walking asymmetry	Percentage of walking when left and right step time are different

iPhone 8 and later

9:41
Monday, June 22

Figure 4: CMPedometer mobility metrics for people with Disabilities
[8]

The **Core Motion Framework** allows the developer to retrieve user motion data for processing and display purposes. The obtained data comes in 2 different forms, namely raw sensor data or software processed data. It is the developer's discretion as to which kind of motion data to include for the intended application. For this application, the processed data was preferred as client side data processing may consume a lot of hardware resources and may not represent the most accurate motion data. This API returns an assortment of methods of which the developer may call in order to query cached data from the IOS sensing hardware. In essence, this framework is the software representation of the motion processing accelerator MPSoC cores available.[3,5,9]

Some of the hardware this framework abstracts and encapsulates include Gyroscope, Pedometer, accelerometer, Magnetometer, and the Altimeter. In this application, the focus of discussion shall be centered around the **CMPedometer** and the **Core Motion Activity Manager classes**. [3,4]

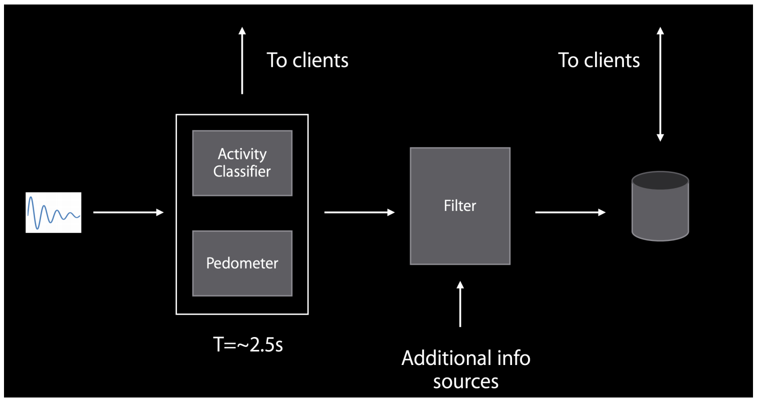
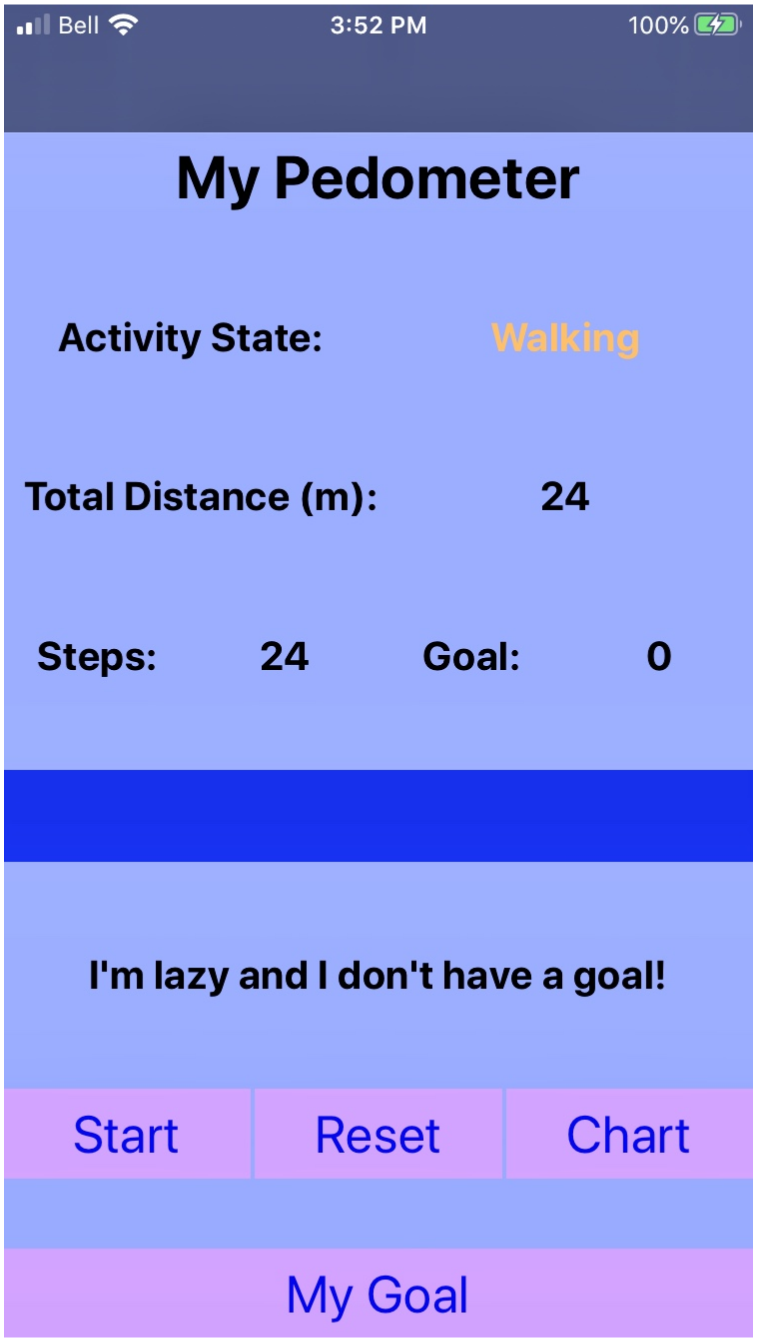


Figure 5: CMPedometer data pipeline
[6]

The **CMPedometer** class represents the filtered steps and stride distance information fed directly from the associated cached raw hardware data points. These metrics display the usefulness, precision and accuracy of the Pedometer sensor within iOS Hardware since this data can be utilized to help rehabilitate people with walking related disabilities. [8] Please refer to Figure 5 for visualization of the data pipeline.[6] Within this class, there exists methods to query live pedometer data, to fetch historical pedometer data, and to determine the availability of the pedometer sensing hardware. To this end, the methods that were utilized within this application included **isStepCountingAvailable()**, **isDistanceAvailable()**, **startUpdates(from start: Date, withHandler handler: @escaping CMPedometerHandler)**, and **stopUpdates()**. The first two methods were used to determine whether if the application had pedometer hardware sensing support. The last two methods were used to feed and halt live pedometer data into the application asynchronously in the application's main thread.[1]

The **Core Motion Acitivity Manager** class represents live user activity data associated with the live motion data gathered from within the **CMPedometer** class. Within this class, the application used **isActivityAvailable()** method to determine if the client iOS contains user activity hardware sensing support. If the hardware was available, it then uses **startActivityUpdates(to queue: OperationQueue, withHandler handler: @escaping CMMotionActivityHandler)** to asynchronously deliver user activity information to the application using the application's operation queue. The type of activities include, 'Walking', 'Running', 'Cycling', Stationary', 'Automotive', and much more.

4 Appendix



My Pedometer

Activity State: Walking

Total Distance (m): 24

Steps: 24 Goal: 0

I'm lazy and I don't have a goal!

Figure 6: Pedometer after load up

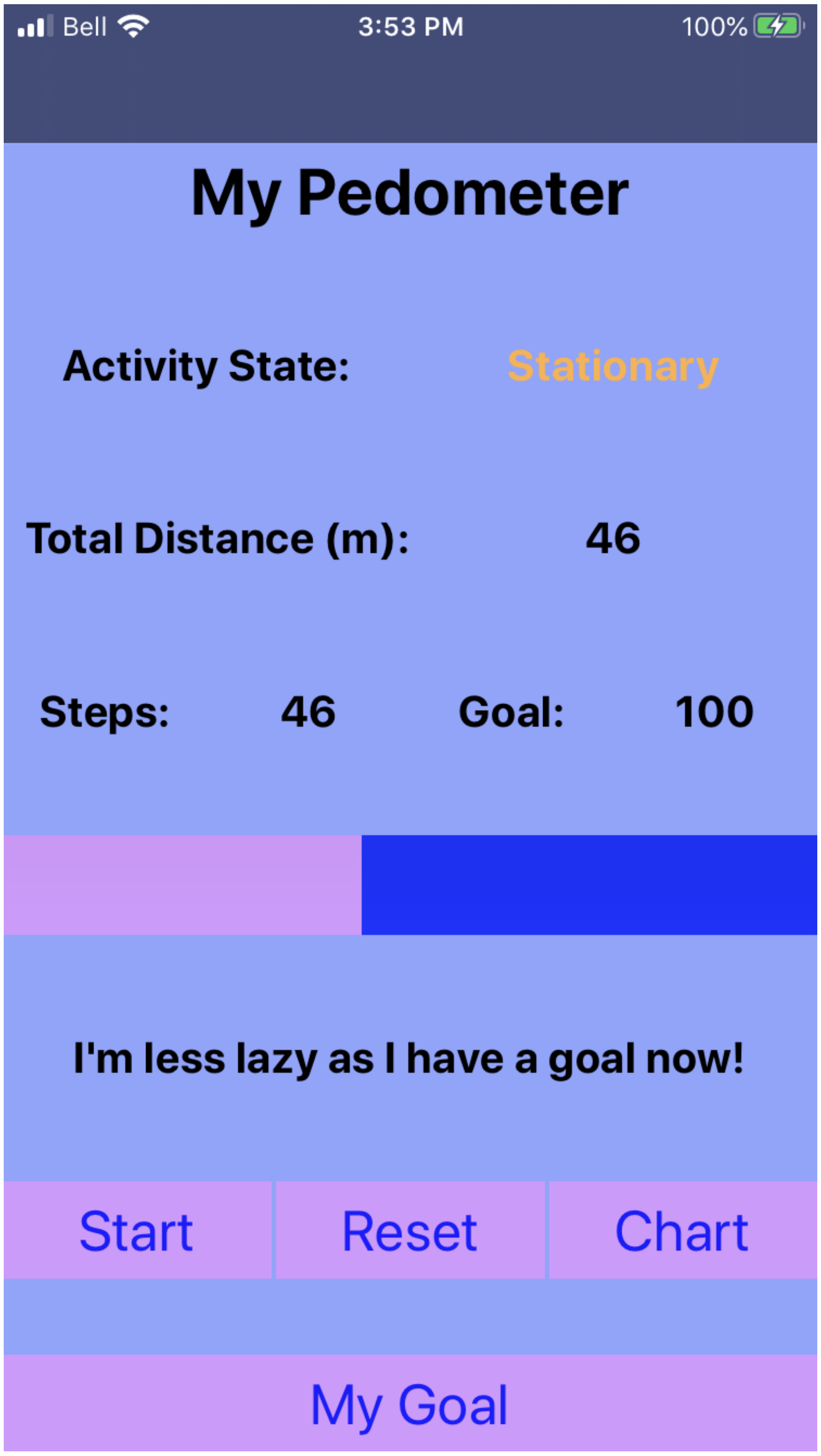


Figure 7: Pedometer during data collection

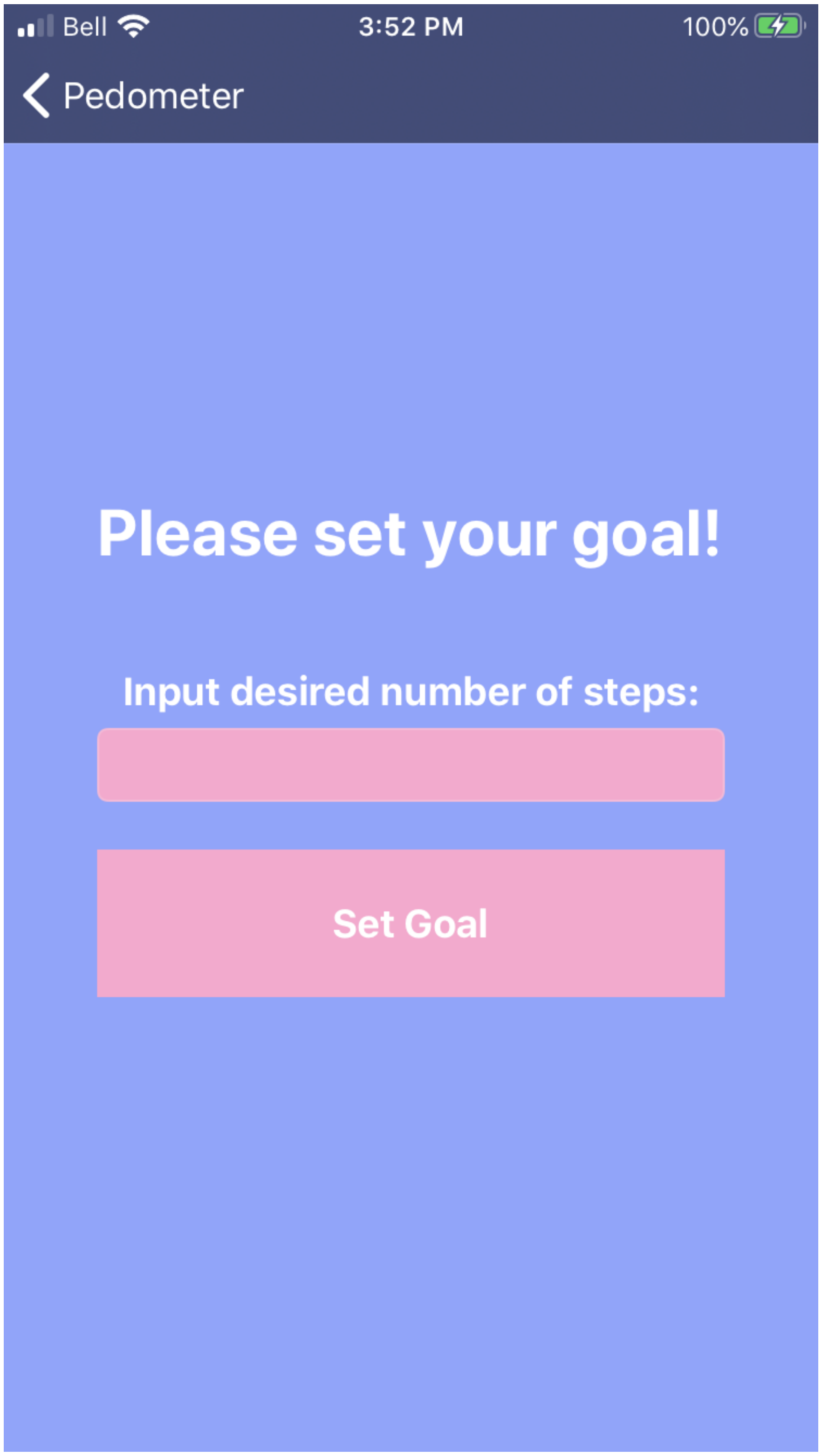


Figure 8: Pedometer goal input view

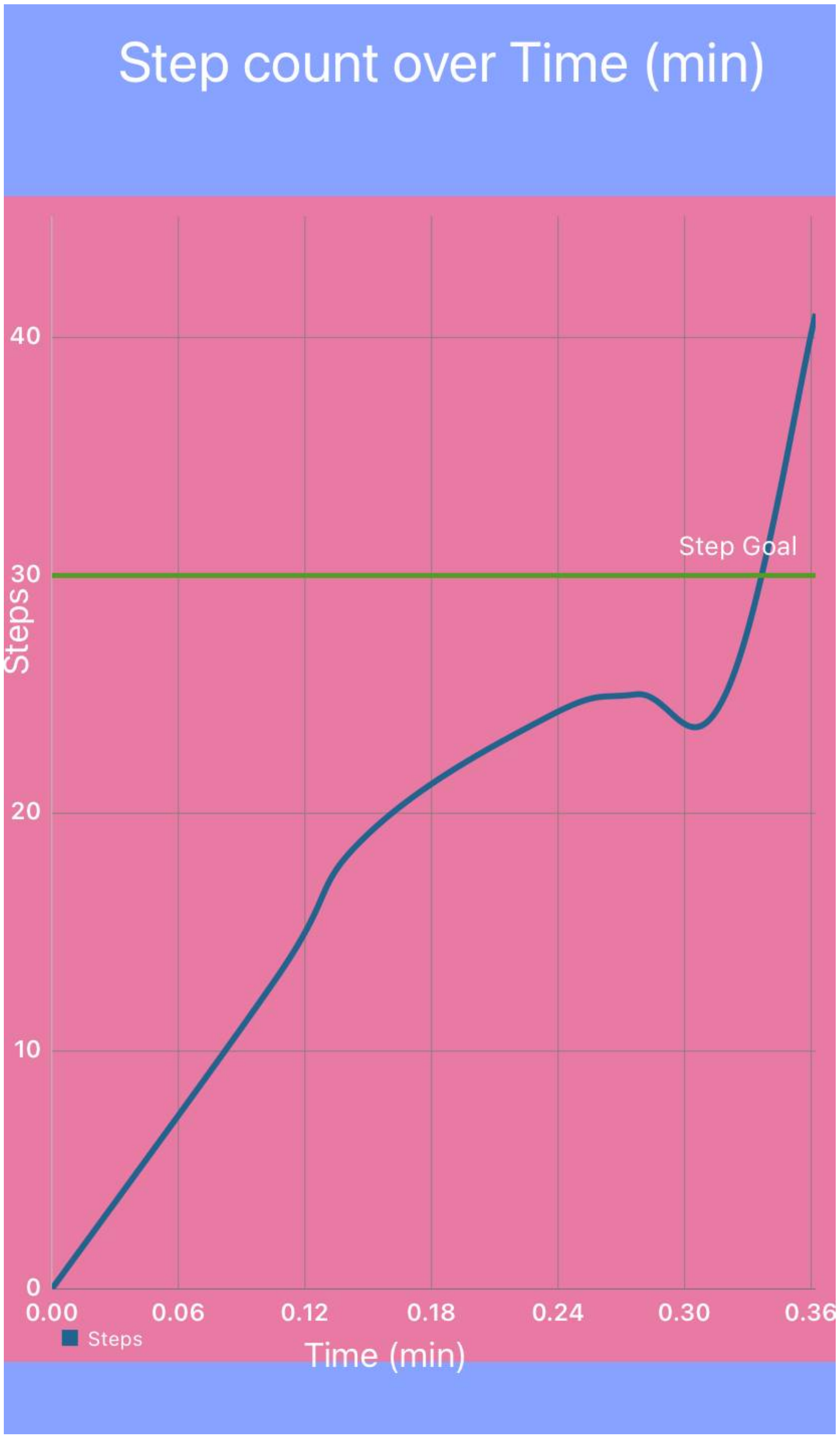


Figure 9: Chart view

5 References

References

[1] Developer.apple.com. 2020. CMPedometer. [online] Available at: https://developer.apple.com/documentation/coremotion
[Accessed 15 October 2020].

[2] Developer.apple.com. 2020. DispatchQueue. [online] Available at: https://developer.apple.com/documentation/dispatch/d
[Accessed 14 October 2020].

[3] Developer.apple.com. 2020. Core Motion. [online] Available at: https://developer.apple.com/documentation/coremotion.
[Accessed 15 October 2020].

[4] Developer.apple.com. 2020.CMMotionActivityManager. [online] Available at: https://developer.apple.com/documentation/coremotion/cmmotionactivitymanager. [Accessed 12 October 2020].

[5] Rao, Bharath, and Paul Thompson. 'PPT.' Apple, June 2016. Health and Fitness with Core Motion. Available at: https://developer.apple.com/videos/play/wwdc2016/713/. [Accessed 13 October 2020].

[6] Pham, Andy, and Sunny Chow. 'PPT.' Apple, June 2014. Motion Tracking with Core Motion Framework. Available at: https://developer.apple.com/videos/play/wwdc2014/612/. [Accessed 13 October 2020].

[7] Gindi, Daniel Cohen. 'Charts.' Github, 2020, github.com/danielgindi/Charts.

[8] 'PPT.' Apple, June 2020. Beyond counting steps. Available at: https://developer.apple.com/videos/play/wwdc2020/10656/.[Accessed 13 October 2020].

[9] Blackwell, John, and Ahmad Bleik. 'PPT.' Apple, June 2017. Creating Immersive Apps with Core Motion. Available at: https://developer.apple.com/videos/play/wwdc2017/704/. [Accessed 13 October 2020].