Sparky: Gesture-Based Home Automation Application

Gary Graham, Jia Ming (Anson) Liang, and Jyotheeswar Arvind Manickavasagar

***Abstract*—Traditional home automation applications mainly use buttons on the mobile phone touch display to toggle the state of home devices. These applications usually require users to navigate through clusters of buttons to control the specific device. Most users would rather flip the switches on their home wall to toggle the specific device instead of going into their applications to find the correct button. We built an Android application to utilize motion gestures to control the home devices. This will eliminate the process of users trying to find the correct button on the screen. With our application, users can toggle a specific device by simply performing a unique gesture with their mobile phone. Our application connects to a Spark Core, which we use for our home automation system. Spark Core is an Arduino-compatible Wi-Fi enabled device that can be used for developing internet-connected hardware. This device enables our application to let users control their home devices anywhere with internet access.**

***Index Terms*—Android, Arduino, Gesture, Home Automation, Spark Core.**

I. Introduction

With the development of our application, we wanted to address what we believe to be an up and coming problem in the world of Internet of Things (IoT). In the home automation space, there are many manufacturers. Each of these manufacturers typically develop applications to support their proprietary devices. However, a quick search on the mobile applications stores show that the UI and usability of these applications leave much to be desired[1]. Cluttered interfaces with no regard to user experience abound. With this in mind, our goal was to create a home automation application that is not only more aesthetically pleasing, but also more functional.

Traditional home automation applications have stuck to the tried and tested method of utilizing buttons inside an application, typically on a mobile device, to toggle the state of devices at home. However, we have many more ways to interact with our mobile devices than simple taps. The overarching goal for home automation is to make the experience seamless for users. If it is not faster or more convenient to use your mobile phone to toggle a device, then why bother? If users want to turn off a light, and it takes them two seconds to take out their phone, 5 seconds to scroll over to the application and open it, 1 second for it to load, and 1 second for users to tap on the correct button, then they have taken 9-10 seconds to do something they could have done in the same time by simply standing up and going to flip the switch manually. We believe that using gesture-based triggers combined with widgets, users can achieve a more satisfactory experience, while reducing the time it takes to actually perform the task at hand.

For the purpose of demonstration, we will only use the three-axis accelerometer data available in our Android phones to perform gesture recognition. The algorithm we use is a simple discrete time series data matching algorithm called Dynamic Time Warping (DTW). It should be noted that gesture recognition is non-trivial and high degrees of accuracy will require using advanced machine learning techniques.

This paper gives an overview of our application, explaining how we setup and connect to the Spark Core[2] home automation system. This paper also explains how we implement our gesture-based interface to control a Spark Core. Finally, we show results of user experience tests, and reveal directions for future work.

II. Overview of the application

The goal of this application is to improve the current state of home automation user interaction. To perform this, we go about it in three ways. Firstly, we improve the communications between mobile and the physical devices at home. Secondly, we implement gesture recognition as a device-toggling mechanism. Third, we build our prototype using publicly available devices, which will allow users to extend our work.

The prototype of the device involves the following:

1) Spark Core
2) Relay Board
3) 2x Electrical Outlet
4) Android Device

Users can plug any of their devices into either outlet. Then, within the android application, they can toggle the power to each of the outlets individually, and become informed via graphical representation if the status of one of the outlets has changed.

For user interaction, we have decided on a multi-faceted approach. The novelty of the application comes from our gesture recognition system which allows the mobile device to register gestures even while the phone is asleep. This in turn toggles power to the devices. Beyond this, we have also implemented widgets, which allows users to define their own interface on their home screens. For example, you can split half your home screen between two widgets for easy access to the functionality, even when not inside the app. Furthermore, we allow users to toggle devices in the traditional sense, using buttons inside the application itself.

To begin with, we discuss the communications method used. To build our prototype, we used a Spark Core. The device exposes a REpresentational State Transfer(REST) API to anybody who has the credentials, and operates over a Wi-Fi network. The advantage of this is that it is extremely

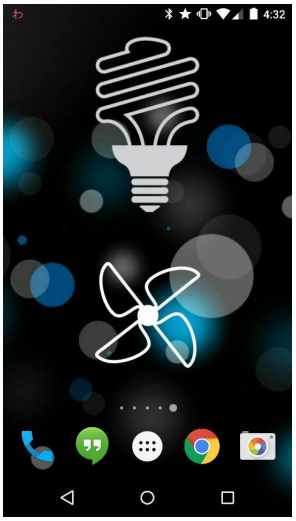

Fig. 1. Widget screenshot of Sparky

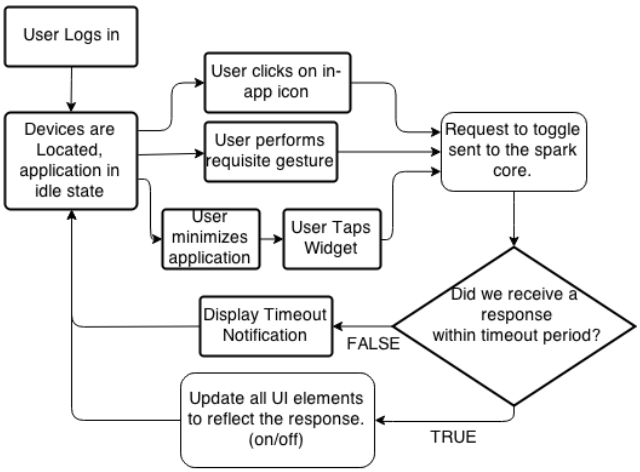

Fig. 2. Application workflow diagram

light-weight, and requires no specific proprietary technology. The Spark Core, being an Arduino-based microcontroller, also allows us to quickly and remotely modify the code running on it.

In this section we give an overview of our Android application prototype, Sparky. We omit low-level implementation details in favour of providing a high-level overview of the interactions. In order to allow the mobile phone to communicate with the Spark Core, we have written a communication library. It is upon this library which the backbone of the application rests. Whenever an intent to toggle is registered, whether by gesture recognition or otherwise, a call is made to the library, which handles request and response to the physical devices. Since we have many UI elements indicating the status of the device (inside the app, multiple widgets), we use the observer pattern to handle notifying all listeners as to the status of the device. For the in-application and widget icons (see Figure 1, we use minimalistic designs that represent the devices they are toggling. This removes clutter from and provides an intuitive idea of how to use the application. Figure 2 shows the overall basic usage of the application.

There are equivalent workflows for other tasks, for example changing the current gesture, but we omit them here due to brevity requirements.

III. Gesture Recognition

Our application uses the Dynamic Time Warping (DTW)[3] algorithm to perform gesture recognition. DTW was originally developed for speech recognition, but we apply it here to gesture recognition. It works by measuring the similarity between two sets of discrete time series data. Instead of speech, we opt to compare accelerometer data in order to recognize gestures. DTW is relatively simple to implement and it works for simple gestures. Below is the pseudocode algorithm[4] for DTW:

```
double DTW(double 01[1...T1], double O2[1...T2])
  declare double DTW[0...T1, 0....T2]
  declare integer i, j

  for i:=1 to T1
    DTW[i,0] := positive infinity
  for j:=1 to T2
    DTW[0,j] := positive infinity
  DTW[0,0] := 0

  for i:=1 to T1
    for j:= 1 to T2
      DTW[i,j] := distance(O1[i], O2[j]) +
        minimum(DTW[i-1,j]), DTW[i,j-1], DTW[i-1,j-1])

  return DTW[T1,T2]
```

Since the data reading is in three dimensions, we treat each time interval reading as a three-dimensional point. The distance function we use in the pseudocode simply takes the norm between the two points.

Before gesture recognition, we train a gesture by obtaining and storing 32 discrete points of accelerometer data. This accelerometer data combined with a gesture label form a single gesture object. By default, our application contains two predefined gestures: Bump Front and Bump Left. We choose these two gestures as default because it is very simple for another person to repeat the movement and it is unique enough that when someone puts their phone in their pocket, they will not accidentally trigger the gesture.

Bump Front in Figure 3 is defined by holding the device straight, tilting it forward by 90 degrees and back to the original position within two seconds. Bump Left is defined similarly, tilting the phone to the left by 90 degrees and back to the original position within two seconds. Users also have the option of creating their own custom gestures via the Start Training button found in Figure 3.

Once the user selects a gesture for a particular device, the application will constantly poll the accelerometer data. This data is compared with the stored gestures created using the DTW algorithm. We return the distance for every stored gesture. This distance metric indicates how similar the two

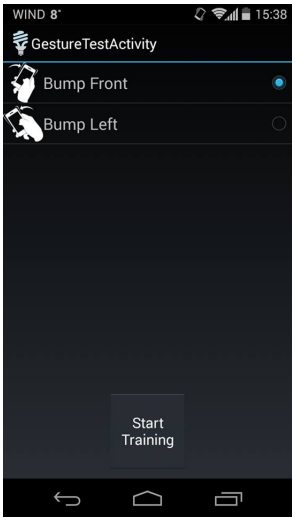

Fig. 3. Default gestures selection page for Sparky

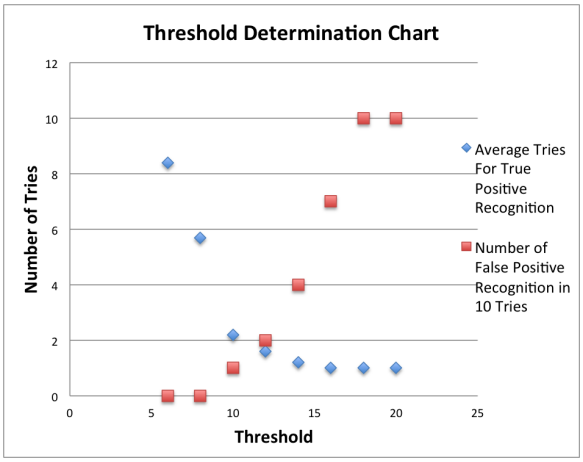

Fig. 4. Chart used to determining distance threshold

gestures are, with a smaller distance indicating they are very similar, while a large distance indicates that they are dissimilar. We take the gesture with the smallest distance as the recognized gesture.

DTW allows some variance in gesture motion and speed when a person tries to repeat the gesture they have trained. The variance is controlled by a threshold variable in our program. We find that using a small variance allows us to get a high percentage of true positive results, but we have found that it sometimes takes multiple tries for users to re-create the same gesture they trained. On the other hand, large variance allows users to easily get their gesture recognized but also has a higher chance of generating false positives.

For our application, we have set the distance threshold by experimentation. We try to set the threshold so that we minimize both the number of tries to get true positive recognition and the number of accidental false positive recognitions. In Figure 4, the diamond scatter plot is the average number of tries to get a gesture properly recognized. We have a person train their own gesture and try to reproduce the same gesture at different thresholds multiple times. We then take the average of this set to the threshold. The square scatter plot is having a person doing 10 random gestures and see how many gestures will get recognized (these are false positive results). We can see in Figure 4 that the ideal threshold value is approximately 12.

IV. User Experience Tests

The application was tested by 17 users aged between 21-27, who consented for us to use the feedback we obtained from them. The participants of the testing consisted of mainly graduate students with a strong technical background. However, we also included several non-technical users as well.

A. Methodology

The participants tested the application individually and no more than two participants were in the testing room at a particular point in time. The participants were initially briefed about the overall idea of the application. The team demonstrated how to use the application first and explained how to perform the default gestures. Then the participants were given access to the mobile device on which the application is running. The participants tested the entire functionality of the application including changing the assigned gestures and creating their own custom gestures. There was no set time limit for the participants to use the application. Upon their stating that they have completed testing the application, they were directed to fill in an online feedback form. The questions on the feedback form were structured to get the users' rating on factors such as the intuitiveness of the application, their comfort using the default gestures, and their willingness/interest to adopt such a technology to control home appliances.

B. User Feedback

Users submitted their feedback and comments online. They also provided their consent on the form before submitting it.

Some comments from the users are: "Takes a bit of trial and error to figure out the default gestures. They seem pretty sensitive.", "The default gestures are good, but, take some time to adjust to the user's personal way of doing these gestures. So, even before using the default gestures, learning these from the user might make this a bit easier.","Why does every single thing have to be internet connected nowadays. Don't track any usage data!" etc.

C. Inference

The application experience rating on a scale of 10 was given a mean of 8.58, and median of 9 by the users. Figure 5 shows the percentage of users who were comfortable using gestures instead of the traditional touch interface to control the application. Although users rated the default gesture experience at a mean of 7.88 and median of 8, about 70% of the users needed multiple attempts to get accustomed to the way the default gestures have been defined. This is attested to by the rating we received for the ease of figuring out the default gesture, with a mean of 7.5 and median of 8. Since the data obtained when a gesture is performed is reliant on orientation

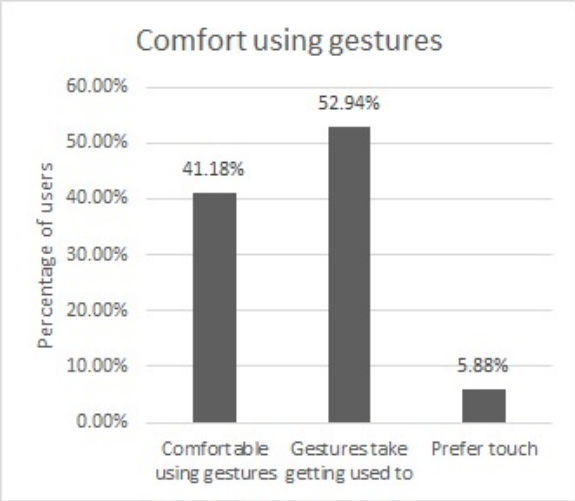

Fig. 5. User feedback on gesture vs touch input

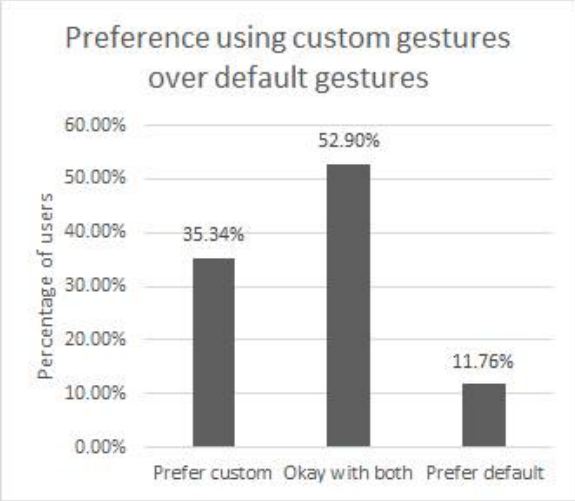

Fig. 6. User feedback on default vs custom gesture preference

and the starting position at the time of doing the gesture, users needed practice before being comfortable. Once they learnt the default gestures, the comfort using them greatly increased. Figure 6 shows user feedback on their preference between using custom gestures and default gestures. The application's intuitiveness received a mean rating of 8.11 with a median of 8. On being comfortable to use the application in a real life setting, users gave a positive 70.5% yes. It is important to note that our numbers may have been accidentally inflated due to the fact that the primary portion of our testers were technically-inclined people aged between 21-27.

V. Future Work

We aim to improve our application in the future by providing an interface for multiple Spark Cores to be connected to our application. The ability to add devices dynamically, and the ability to customize the icons for each device are more features that could be worked on in the future.

Another direction for future work is to improve the accuracy of the gesture recognition by incorporating the three-axis gyroscope. We also want to have an option for users to utilize the magnetometer to control multiple devices with the same gesture but perform the gesture in a particular direction. This would allow a user to point their phone at a particular device, perform a generic gesture, and have that specific device toggled.

DTW is an effective algorithm when the gesture is a simple motion, as it allows some small variation in speed and motion. However, our user feedback results show that the default gestures are minimally comfortable because the algorithm generally performs poorly when the moving rate and motion has a large variation, as is common among different users. Furthermore, with more complex gestures DTW does not work effectively. One alternative is to do gesture recognition via Hidden Markov Models[5]. Using this, we may realize some accuracy gains, because it uses knowledge of probability and statistics to account for larger variation of speed and motion, while still retaining the ability to accurately recognize gestures.

We would also like to test the application among a wider audience to observe how much this technology is applicable to the greater populace.

VI. Conclusion

In this paper and its accompanying mobile application, we have shown that not only is it possible to replace traditional home automation application usage patterns with gestures, but also that it is generally effective. The feedback we have received gives a few important insights that point towards future work. Clearly, the level of accuracy for the gesture recognition, while acceptable, has a long way to go in order to be adopted by any sizeable segment of the home automation market. Given the current state of the smart home and the huge recent advancements in the world of IoT, we feel that gesture-based recognition points the way towards the future of home interactions. We also feel that our application demonstrates that it is very feasible, once accuracy levels become high enough.

References

[1] Web Mountain Technologies. (2014, Dev 5th). Home Automation Controller [Online]. Available: https://play.google.com/store/apps/details?id=com.webmtn.android.app

[2] Spark IO. (2014, Sept 21st). Spark Core Controller [Online]. Available: https://www.spark.io/

[3] H. Sakoe and S. Chiba, *Dynamic programming algorithm optimization for spoken word recognition*, IEEE Trans, on Acoust., Speech, and Signal Process., ASSP 26, 43-49 (1978).

[4] P. Borza , *Motion-based Gesture Recognition with an Accelerometer*, Babeş-Bolyai University, Cluj-Napoca, Romania, July 3, 2008

[5] A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition, Rabiner Lawrence R, 1989, IEEE, pp. 257-286.