A Novel Approach to Home Automation:
Application Development with Google Glass

Rupesh Chinta
rupesh.ch97@gmail.com

Roland Fong
rfblue2@gmail.com

Isabel Murdock
murshu3@gmail.com

William Kang
williamkang531@gmail.com

Quinn Williamson
quinn14thierry@gmail.com

ABSTRACT

Recently developed wearable devices have opened up new capabilities in human interactions with the world. Home automation is an area of interest for the application of wearable devices. Currently, home automation is performed with home control panels or mobile devices. This paper, on the other hand, presents an implementation of home automation using the Google Glass. With a few simple voice commands or finger swipes, the user can monitor the home and control home appliances, thus bringing a new level of interaction with the home. The Glass communicates with an online server that relays commands to a microcontroller to execute certain tasks. The microcontroller is, in turn, able to communicate sensor information to the Glass through the same server. This framework serves as a prototype that demonstrates the potential for wearable technology to simplify the user's in home experience. In the future, it can be expanded to include more features and more devices for the user to interact with in the home.

1 INTRODUCTION

The rapid advancement of modern devices has led to an outgrowth of new ways for people to interact with the world. Among these advancements is wearable technology, which combines the computing power of a smartphone with convenience and accessibility. Wearable technology consists of computing devices that can be worn, like a belt or a backpack, and have a form of input, such as a touchpad[1]. The rise of wearable technology opens up a new wealth of possibilities for potential applications, including sensing the user's immediate environment[2], navigating[1], and assisting medical professionals[3]. One application of wearable devices is in home automation. A primary goal of home automation is to optimize the efficiency and comfort of home residents. However, current efforts to fulfill these desires face limitations ranging from restricted portability, such as that provided by wall panels, to a lack of a nearly hands free experience when using smartphones and tablets[4].

Wearable devices solve these problems with constant accessibility and awareness of their context, or the user's surrounding area[1]. Because it is constantly accessible, the interfaces of most wearable devices are designed to accommodate constant usage[5]. Google Glass, which focuses on simplicity and efficiency, best fulfills these necessities. It allows users to easily complete daily tasks through advanced voice and motion recognition. In addition, the Google Glass is comfortable, because it can be worn like ordinary glasses, providing an unobtrusive, yet novel experience.

This research seeks to design an innovative approach to home automation through Glass application development. The key features of our application, "Start home auto", incorporate the Android Software Development Kit (SDK), Arduino, and the Google App Engine.

2 BACKGROUND

2.1 Why Google Glass?

Google Glass is a wearable device with an optical head mounted display and a touchpad on the right temple of the glasses. The user can interact with Google Glass both by using voice commands and by tapping/swiping the touchpad. Glass displays information in the top-right of the user's field of view, in a location that is easily accessible without obstructing the user's sight. Additionally, Glass can connect to the Internet through Wi-Fi and Bluetooth, and can use this to access online devices and servers.

2.2 Components

The development of this application relies on several programs, tools, and hardware. The Android Development Toolkit (ADT) allows development of android mobile applications on the Google Glass. The Google App Engine, a "Platform as a Service" (PaaS), allows the android application to access an online server. Finally, the Arduino microcontroller, an open source electronics platform, controls most of the hardware.

2.2.1 Android Development Toolkit

Google Glass runs on the latest Android's operating system, 4.4.2, also known as "Kit Kat." This Android Development Kit (SDK) includes a debugger, libraries, a handset emulator, sample code, tutorials, and documentation. The Java programming language is used to write Android applications. The Extensible Markup Language (XML), is also used for design in the ADT. XML is a common programming language designed to carry data. In the ADT, XML is used to declare static structures like that in the Android Manifest. The Manifest holds the application name, version, and icon (not applicable with this application) as well as permissions from the operating system, used to request additional features of a device's hardware or access personal information and services. The officially supported Integrated

Development Environment (IDE) is Eclipse.

In order to code specifically for the Google Glass, the Glass Development Kit (GDK), a revision to the ADT, is used. The GDK handles specific Google Glass functionality, such as voice commands. Additionally the entire project is complied with the GDK and targets the Google Glass instead of the Android phone operating system.

2.2.2 Google App Engine

This PaaS allows developers to build and run applications through Google. A PaaS is a category of cloud computing that allows for the building of applications and of services over the internet. Uses of the App Engine are for applications that have web service components, applications that keep a profile or sync across devices, and applications that need authentication. AppEngine Endpoints API parses parameters as JSON (JavaScript Object Notation), a lightweight data interchange format, to translate data between servers. In the case of the application, Google App Engine was used to create servers. The servers allow Arduino to communicate with the application via the Internet and vice-versa.

The Google App Engine is especially tailored to this application because it is designed specifically for interfacing between servers and android or Google products. The App Engine API also allows flexibility for controlling the input and output to and from the server. Its annotation-based code gives it ease of use from a developer’s perspective.

2.2.3 Arduino

The Arduino is a single-board microcontroller that senses the environment by receiving inputs from many sensors. It can also output data and be programmed, allowing it to control devices. While there are many different Arduino models, the Arduino Yún is the microcontroller of choice because it is optimized for wireless connections, which are crucial to the application. The Yún also contains a USB-A port, micro-SD card slot, 20 digital input/output pins (of which 7 can be used as PWM outputs and 12 as analog inputs), a 16 MHz crystal oscillator, a micro USB connection, an ICSP header, and 3 reset buttons. Arduino Yún is programmed with a C/C++ based language using the Arduino IDE.

3 DEVELOPING A GOOGLE GLASS APPLICATION

3.1 Overview

Start Home Auto was created to demonstrate the potential of wearable devices in home automation. The application ultimately allows the user to remotely control home utilities. The project consists of three components: The Google Glass application, an online server hosted on App Engine, and hardware controlled by an Arduino microcontroller.

3.2 Glass Application

The Google Glass runs on the Android operating system, and is programmed using the Android Development Toolkit (ADT) in the Eclipse development environment. Using the voice command "Start Home Auto," the user can open the application and view the home screen, a menu that allows the user to access the toggle menu, program menu, or live stream video. These options can be used to wirelessly control hardware and monitor the home through live streaming and motion detection.

3.3 App Engine Server

Google App Engine Server allows the Google Glass application and Arduino to interact. The communication flows in both directions. The Glass application can send a command, such as "lights on," to the server. The command is then relayed to the Arduino, which performs the actual command. When the Arduino detects motion, it will prompt the server, and the server provides a timestamp for when the motion was detected. The Glass application polls the server periodically and when it finds that motion is detected, it will get the timestamp and notify the user.

3.4 Arduino Microcontroller and Hardware

The Arduino microcontroller toggles a switch, which is directly connected to the hardware, namely the lights and the air conditioner. With the output received from the Google App Engine, the Arduino turns the devices on or off. In addition, the Arduino is connected to an infrared sensor, which acts as a motion detector. When the values of the infrared sensor fluctuate, the Arduino notifies the server and then pauses before again checking the sensor.

4 RESULTS AND DISCUSSION

Figure 1. Project Schematic

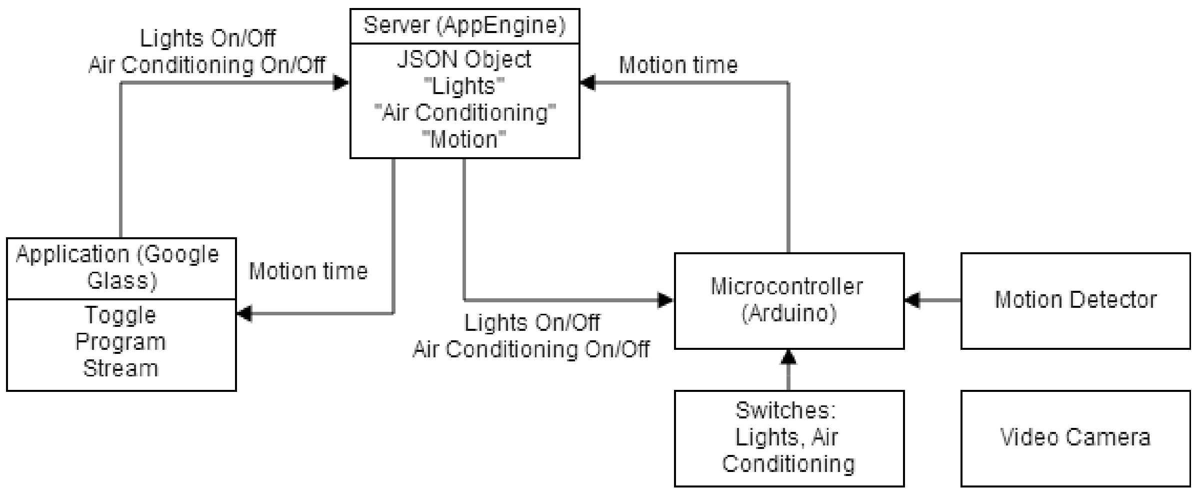


Table 1: Program Classes

Classes	**Description**
MainActivity	Home Page. Allows access to Toggle, Task, and Streaming Activity.
LiveStreamingActivity	Live stream feed from camera connected to Arduino.
MotionNotificationActivity	Not on UI thread. Notifies user about motion detected by device on Arduino.
TaskActivity	Allows addition of programmable tasks (actions that will occur at a certain time)
ManageActivity	Allows the user to delete existing tasks
ToggleActivity	Allows access light switch and AC switch
ToggleGetService	Starts a service that runs in the background which checks whether motion is detected
States	Holds the states of each toggle-able activity as well as the motion timestamp
ToggleGetTask	Application can receive information from Arduino through the server.
ToggleJsonParser.java	Parses string edited in Toggle Activity so server can understand individual commands
TogglePutTask.java	Sends commands to server through JSON

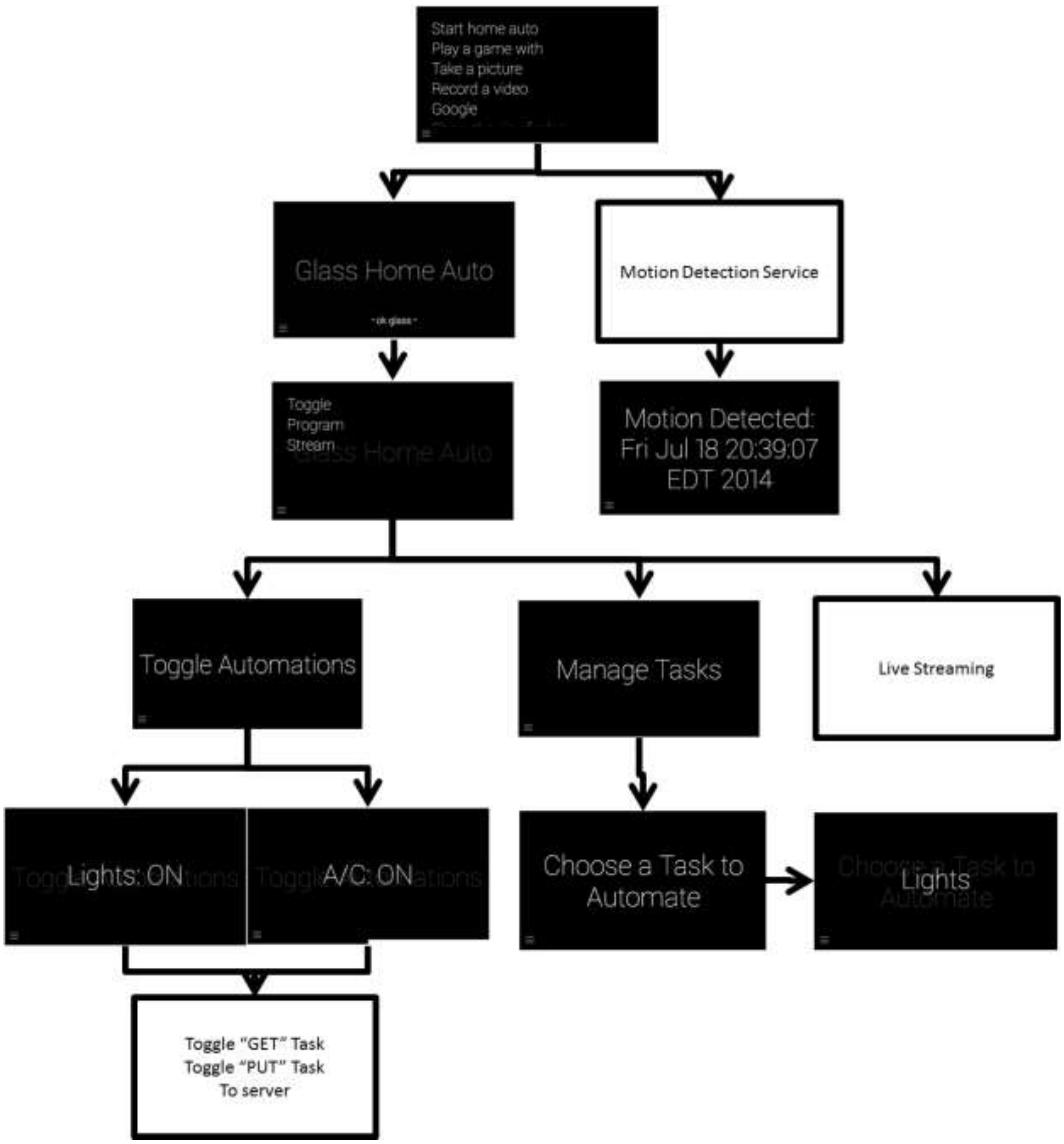
4.1 Application Activities and Classes

The finished application has a simple user interface that includes few activities. The activities included are a Toggle Activity, an Add Tasks Activity, a Manage Tasks Activity, and a Live Streaming Activity. In the actual code for the application, there are a total of eleven classes. The Notification Activity informs the user about detected motion. Otherwise, there is no separate screen for this notification. The remaining classes communicate with the Google App Engine Server. Table 1 demonstrates the various uses for each of the classes in the application.

4.2 Application Expandability

The current prototype of *Start Home Auto* and associated hardware includes capabilities to toggle lights and air conditioning and to notify the user of motion. The user can also create a task program that will automatically execute the toggling of lights or the air conditioning at a preconfigured time. When the user navigates to the Toggle Menu, he or she can tap the Google Glass to bring up the Toggle Menu, which will give the current status of the lights and air conditioning and allow the user to toggle between states by tapping. Additionally, while the Google Glass is on, whenever the Arduino detects motion, Google Glass will alert the user and give the time and date it was detected. The user can then take appropriate action.

Figure 2. Glass User Interface Diagram



The current framework of the sharing of information from Google Glass to the Arduino and vice versa is a framework that can be expanded to include a wide variety of tasks and capabilities, which greatly simplifies the user experience. For example, the Google Glass can be programmed to tell the Arduino to open the garage. The Google Glass can then track the user's location and the user can configure the application to open the garage or the lights when the user is within a certain radius of his or her home. The Arduino, or any microcontroller, can receive data from sensors around the home. With appropriate hardware and utilizing the convenience of Google Glass's voice commands, the user can ask Google Glass to do even more. For instance, a coffee maker can be configured to make coffee every time the user asked Google Glass, "Glass, make me coffee."

4.3 Using "Start Home Auto"

When the user turns on the Google Glass, he or she can view the applications using the voice command "ok glass." Upon reaching the menu, the user can say, "start home auto" to run the application. This command takes the user to the main activity. The main activity is a list including three options: *Toggle*, *Program* and *Stream*. Using voice commands again or taps on the touchpad allows the user to choose an option.

If *Toggle* is chosen, then the user has two screens, which he modifies to send information to the server. The first is the light toggle, with the option to change the status of the "Lights" to "ON" or "OFF" with the tap of the touchpad. The second option on Toggle is the AC toggle, which functions the same way.

If *Program* is chosen, then the user again has two options: *Add Tasks* or *Delete Tasks*. Tasks include programming the application to automatically toggle (either the lights or the AC) at certain times. In the future, such an activity can also be programmed to toggle certain tasks when the user is a certain radius away from the home.

If *Stream* is chosen at the main activity, then live stream from the security camera feed is displayed. The application checks for motion detection every minute from the server. When motion is detected, Glass notifies the user with the time detected. Figure 2 summarizes the Glass User Interface (UI).

4.4 Challenges Faced

Several factors made designing and implementing home automation using Google Glass difficult. Recent updates to Android's operating system introduced bugs in the latest version of the Eclipse IDE, which meant that it would have to be reverted to an older version. The Google Glass, being a relatively recent development, did not have all the tools available for ease of development. For example, the lack of an emulator made testing Glass code difficult. Additionally, the Arduino Yún, designed specifically for Wi-Fi connections, experienced problems accessing the internet and the secure server.

Another challenge was using Glass's simple interface to program some complicated aspects of the user interface. For instance, selecting a time on a phone is trivial due to the screen size (which is larger than that of Glass) and its touchscreen. Glass, on the other hand, uses a smaller screen, only a touchpad, and a simplistic design philosophy. As such, to select a time when programming tasks, a user must navigate through four menus, first, a menu containing all the hours, then a menu containing all the "tens minutes" (the tens digit of the minutes), then the "ones minute," and finally, whether the time is AM or PM.

4.5 Future Employment

The concept of home automation using wearable devices is easily expandable, and this project demonstrates just a fraction of its capabilities. "Start Home Auto" was created to be built upon a model for future applications. Its capabilities are minimal as a result of time constraints and limited access to home devices. Home automation systems should maximize user customization. "Start Home Auto" in some ways limits the extent to which the user may personalize the application for his specific needs. The application allows the user to toggle only two devices. Applications that will build upon these features should allow the user to more easily add devices and program more features with ease.

Home automation application systems will eventually be more accessible to the public. Because the application depends on its users to have some background in code to understand its works and use it efficiently, it is not entirely practical. Future applications will build upon its functions and current capabilities as a first of Glass home automation applications. The next generation of applications of its kind will make a home automation application that is more accessible and user-friendly to consumers.

5 CONCLUSION

"Start Home Auto" serves as a prototype for future projects involving the integration of the home with wearable devices. The Google Glass worked as an excellent medium for exhibiting the potential of wearable devices because it incorporated simple convenience with powerful computing unmatched by other kinds of wearable devices to date. The simple interface, which includes a menu to toggle activities, program tasks, and stream video, captures three of the most basic aspects of the integration of such technology with the home. The user can control his or her home utilities and monitor the surroundings without so much as moving a finger. Additionally, the motion detection serves as an example of how appliances in the home can communicate back to the wearable device. Flexibility that is inherent in the program allows for an easy expansion of functionality for more practical applications.

ACKNOWLEDGEMENTS

The authors would like to thank Mr. Victor Kaiser-Pendergrast for his guidance throughout the Android development process and his insights on the Google Glass application. The reference resources provided by Mr. Kaiser-Pendergrast also contributed greatly in determining the most accurate and efficient process for setting up a network for this research. The authors would also like to acknowledge their Residential Teaching Assistant Anthony Yang for overseeing and providing support and guidance throughout the duration of the project. The research group is also grateful to Rutgers University, the Governor's School of Engineering and Technology (GSET) program, Director Dean Ilene Rosen, Assistant Director Jean Patrick Antoine, the program sponsors, and all Residential Teaching Assistants for all the support crucial for the success of the group. Finally, the research group would like to thank the sponsors of the GSET program: The State of New Jersey, Morgan Stanley, Lockheed Martin, Silver Line Windows, South Jersey Industries, Inc., The Provident Bank Foundation, and Novo Nordisk.

REFERENCES

[1]Billinghurst, M., and T. Starner. "Wearable Devices: New Ways to Manage Information." *Computer* 32, no. 1 (1999): 57-64. doi:10.1109/2.738305.

[2]Farringdon, J., Moore, Andrew J., Tilbury, N., Church J., and Biemond, Pieter D. "Wearable Sensor Badge and Sensor Jacket for Context Awareness." *Third International Symposium on Wearable Computers*, 1999. Accessed July 18, 2014.

[3]Hung, K., Y. T. Zhang, and B. Tai. "Wearable Medical Devices For Tele-Home Healthcare." *26th Annual International Conference of the IEEE EMBS*, September 1-5, 2004. Accessed July 16, 2014. doi:10.1109/IEMBS.2004. 1404503.

[4]Yamazaki, Tatsuya. "Ubiquitous Home: Real-life Testbed for Home Context Aware Service." *International Journal of Smart Home* Vol. 1, no. No.1 (January 2007). Accessed July 14, 2014.

[5]Lumsden, J., and S. Brewster. "A Paradigm Shift: Alternative Interaction Techniques For Use With Mobile & Wearable Devices." *Proceedings of the 2003 Conference of the Centre for Advanced Studies on Collaborative Research*, 2003, 197-210. Accessed July 16, 2014.