Gesture Based Information Transfer System and Analysis

Andrew Nguyen
andnguyen1994@gmail.com

Franklin Shih
franklin.shih@gmail.com

Son Pham
sonpham2@uw.edu

ABSTRACT

SiGlove aspires to recreate the way in which we approach connecting with the various devices of our day to day lives. We have established a new input interface to interact with all the Bluetooth enabled systems that people use. By using a glove based design, we have developed an extremely intuitive approach to controlling each device since each gesture is registered to a behavior of the system. By bending your hand or turning the orientation of the glove, five distinct motions are made measurable and thereby can provide easy controls. SiGlove has developed the future of interactive systems. An easy to use control scheme that can adapt to any situation and bridge the gap between natural instincts and complex technologies far more easily than the interfaces we presently use.

Author Keywords

Wearable; wireless device; Android Bluetooth; Arduino sensors; Accelerometer

ACM Classification Keywords

H.5.m. Information interfaces and presentation: Miscellaneous

INTRODUCTION

SiGlove was started with the goal of simplifying the interface development process. Most modern interfaces are overly complicated and not intuitive to use. Even worse, there are dozens of different interfaces used for daily life. We have our smartphones, keyboards, mice, video game controllers, and much more. It is truly a shame since modern technological innovation has pushed us beyond into the possibilities of making a universal, simplistic interfaces a real possibility. That is exactly what SiGlove has brought to the table. Rather than using a traditional focused model for development, SiGlove has aspired to develop a system that can connect to any device and provide intuitive controls.

In this project, we introduce a glove based interaction device that allows remote connection to a bevy of hardware systems through a general Bluetooth connection. By sending in simple sensor data, SiGlove is able to be developed for quickly and easily. The applications can control music, keyboard functionality, and even gaming through a series of five independent gestures—bending the hand, up tilt, down tilt, right tilt, and left tilt. These gestures simplify the process of controlling complicated devices to just a series of motions. As a result, SiGlove has a universal controlling ability with application development at its core.

Figure 1:SiGlove device

Furthermore, by using a glove based setup, we have created a simple and formfitting device that can be used every day while providing advanced features that will enhance everyday life through universal digital control.

COMPETITORS

SiGlove is a better product than those already available on the market because of its affordability, sleek design, and convenience. Most of the most advanced glove based input systems on the market are either very expensive or very bulky. The bulk comes from the significant number of sensors required for their products. However, by limiting the number of gestures that can be detected to just five, the product is both cheaper and easier to use. Furthermore, unlike products like Kitty or the Music Glove, SiGlove is form fitting. The glove is convenient and easy to put on. The 3d printed box of electronics is small and unlikely to burden the wearer. As a result, the product is better for everyday use.

THEORY OF OPERATION

This device uses data collected from various sensors attached to a glove to manage controls on a separate device. The two sensors used in this project are a triple axis accelerometer and a flex/bend sensor.

The accelerometer detects acceleration in different directions. In this case the accelerometer is used to detect acceleration caused by gravity which will determine which direction the accelerometer is facing. For example when the device is tilted downwards, the acceleration in that downwards direction will be greater thus the accelerometer will output a greater value in that direction. The information about the orientation of the device is vital to the desired operation of the SiGlove unit. With the accelerometer in the center of the glove, the device can read tilts of the users hand. The observed values are collected for the four tilt directions used and from that the threshold value to set each

tilt is appropriated. Each direction is mapped to a different command on the connected device.

The flex/bend sensor is a variable resistor who's value depends on the amount of bend in the sensor. This sensor is comprised of conductive materials that are spaced out on the sensor. When the sensor is bent the conductive materials move closer and further apart which changes the resistance of the sensor. This sensor is placed on the knuckle of the glove where it is bent as the user closes his/her fist. This provides a binary value which is used as a button press in the connected device.

All of the sensor data is transferred via Bluetooth connection. Bluetooth is a wireless serial data transmission protocol. Building and application on the android and pc that manages a Bluetooth connection is an involved process that requires additional threads so that the connection does not hog all of the CPU.

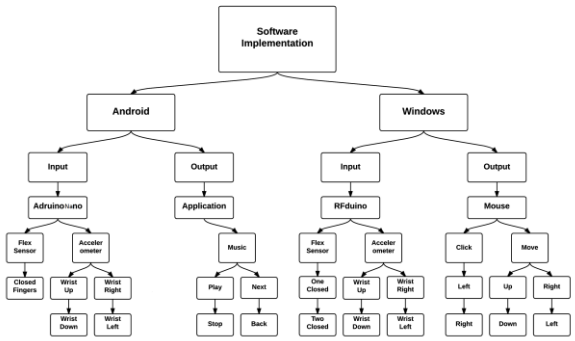

Figure 2: Software Implementation Diagram

SYSTEM IMPLEMENTATION-HARDWARE

In order to get gesture data from the hand, we used the Sparkfun MMA8452Q triple-axis accelerometer and flex sensor. Both of the sensors are connected to an Arduino Nano which uses the ATmega328 chip. The Nano runs on the Arduino platform which allows for plenty of flexibility to work around the sensors with. More specifically the Nano was selected due to its size over its larger counterpart, the Uno. The accelerometer is attached to the center of the back of the glove while the flex sensor is attached to the middle finger on the glove. The flex sensor acts as a variable resistor thus the data is sent directly to the analog to digital port of the Arduino.

The accelerometer sends I2C data there its SCL and SDA ports are connected to the corresponding ports on the Arduino. In order to conserve battery life, the accelerometer is set to its low power, low resolution setting which is acceptable for our design since we are not measuring intricate and minute movements but rather macro scale movements of the hand. For our communications with other devices we are using an hc-06 bluetooth module.

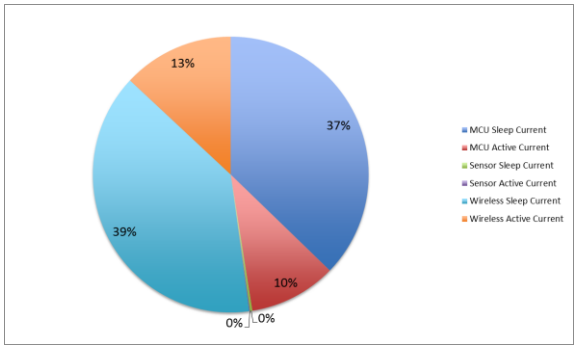

Figure 3: Power Consumption Breakdown

Finally, for our power supply, we calculated the average current consumption of our device to be 1.61mA. This calculation includes the time that the device is off as well as when the device is asleep. In actuality the device spends the majority of the time off as well as asleep thus consuming little to no current. Figure 3 shows the current consumption percentages.

Originally we were using a disposable 9V battery. However this did not meet our desired size constraints as well as needing replacement multiple times a week. We determined that a 1000mAh lithium ion battery is the best fit for this project due to its size, total supply, and cost. The battery supplies 3.7V thus a boost converter is used to step up the voltage to 5V. Another added benefit of the lithium ion battery is that it is rechargeable thus removing the need to dispose of the battery constantly and adding convenience to the user.

Figure 4 shows the schematic diagram of the SiGlove device. The two resistors connected to Analog 0 are the flex sensor and a 10K Ohm resistor which is used in the voltage divider circuit. The 330 Ohm resistors are used for level shifting since the MMA8452Q accelerometer runs at about 3.6V and the Arduino runs at 5V. Connecting the Bluetooth module was simply connecting the VCC of the module to the 5V out of the Arduino, connecting the grounds, and wiring the RX and TX of the Bluetooth device to the TX and RX of the Arduino respectively.

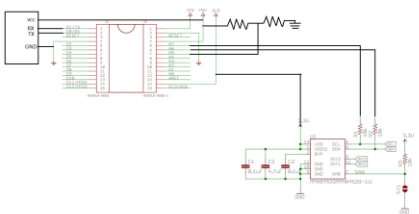

Figure 4:Circuit Schematic

The housing unit was aptly designed to accommodate all of the components that won't be physically on the glove

including the Arduino, Bluetooth module, and battery. The housing unit needed openings for the wires connecting the sensors to the board to come through as well as an opening for the charger and power switch. The design also includes holes for the Velcro strap to come through in order to hold the housing unit on the user's arm. The design was realized using a 3D-printer. Figure 5 shows the Solidworks model of the final design.

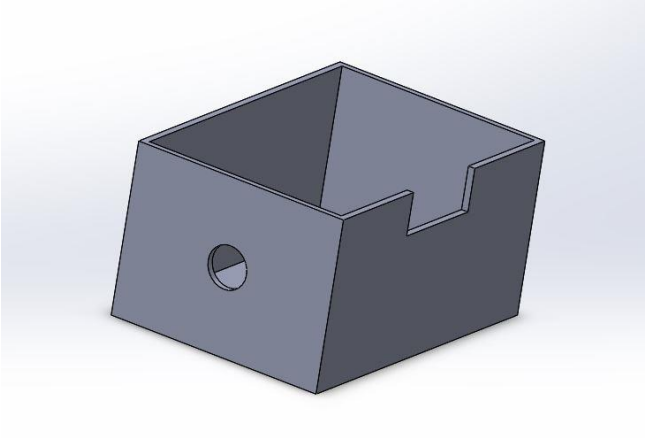

Figure 5:Solidworks model of housing unit

System Implementation-Software

Programming the Arduino began with adding libraries to the Arduino in order for it to work with the other hardware components such as the MMA4852Q. The program consists of a sleep function as well as measurement and print function. The sleep function sets the device to the lowest sleep mode in order to conserve the most battery power. A button is used for the low level interrupt to trigger the wakeup of the Arduino. This sleep function is called after every measurement is taken. The implication of this is that the device only takes one measurement per button press which then makes clocking the readings irrelevant.

The measurement and print function reads the data from the sensors, puts the data into a packet, and prints that packet in serial. Reading from the accelerometer requires the accel.read() function to be called which is a library function that takes a reading from the accelerometer. This data is stored in accel fields x, y, and z. Reading the flex sensor was simply reading the analog values from port A0. These values are stored in order to be put in the data packet.

There were several requirements for the data packet. First it needed to have a character to designate the start of the packet. Secondly it needed a way to separate the different measurements. Finally, the packet needs to have a terminating character. In order to fulfil these requirements the structure of our packet is shown in figure 6.

#	X Val	+	Y val	+	Z val	+	Flex Val	~

Figure 6: Data Packet

The Arduino also needed to be set up to properly send serial data over Bluetooth. This requires initializing the baud rate for data transfer to 9600bd. The print function is used to print the data packet which sends the packet to the desired device via Bluetooth.

On the Android side of the software implementation, the app is made up of a single activity and a background service. The activity displays a list of Bluetooth devices that have been paired with the phone. The activity also makes sure that the user has Bluetooth enabled on their Android device. The user is then able to select a device from this list, in this case the SiGlove module, to connect to. The selection starts a service which makes a thread for the Bluetooth connection with the SiGlove device. To verify that the service is started a message pops up on the screen that says "started" to signify that the app is working properly.

The service receives data about the Bluetooth device, such as MAC address from activity via intent. This service handles the data sent from the SiGlove device as well as managing the connection. The data packet sent from the Arduino is interpreted by the service which then sends out the proper key event. The service checks to make sure the data packet is complete and in the correct form. It does this by making sure that the packet begins with the '#' and ends with a '~'. Once this is confirmed, the program sends key presses corresponding to the sensor readings. These key events simulate physical button presses that match each desired. For this implementation, we set the up-tilt to be volume up, down-tilt to be volume down, right-tilt to be next song, left-tilt to be previous song, and closing the palm toggles the play/pause button. This service runs until the device is switched off in which case the service destroys itself and stops consuming resources.

Developing the desktop application for the SiGlove was an interesting dilemma. There were multiple facets that needed to be accounted for. Narrowing down the program led to developing in Java for Windows. The first step was to create a GUI for the system to use. This was done using Java FX scene builder to reduce the level of difficulty. The system had five separate text boxes, a check box, one button, and one on/off button. The text boxes stored the values to be entered for each gesture while the check box would turn on and off mouse controls. The on/off button was used for activating the robot in the second phase.

The next step was to develop a system to read values from the textboxes and write them when the system received the appropriate signal values. This was done by developing a 'robot' from the imported java.awt.Robot file that would

print out the characters in the text boxes. In order to ensure that there was only one character in each textbox, a simple algorithm was made that would check how long the length of the text box values were and prevent the system from writing any more than one character. As for how the 'robot' would know what to print, the system would analyze the same string of Bluetooth values used by the Android program to determine the appropriate signals. Getting the mouse controls to work were simple in that, instead of using the textbox values, the system would look at the mouse's current location and add a few of the appropriate values to change the mouse's location.

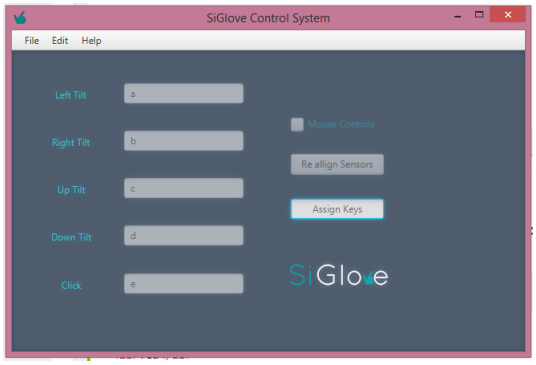

Figure 7: PC Interface

The last step was to implement the Bluetooth system and test the overall program. This was a particularly challenging issue because of connectivity issues with the COM ports on Windows. The first step was simply to pair the glove device to the computer. The program the used the RXTX system to connect with the COM port that was being used by the computer. While this sounds easy in theory, issues did arise with the COM port appearing to be taken by the glove itself when the glove attempted to connect. As a result, rebooting the system was often required to ensure that connections worked on the first attempt. Once the Bluetooth was working, implementing it with the robot system was simple in that the string that the Bluetooth read would be used by the Robot system.

Experimental Procedures and Results

The first sensor tested was the accelerometer. In order to be able to detect gestures, we needed to understand the raw data of the accelerometer. To test this we simply connected the accelerometer to the Arduino and had the sensor values printed to the console. From the raw data we were able to determine the threshold we set for the up-tilt, down-tilt, left-tilt, and right-tilt gestures to be 500 units in each direction from neutral. For the flex sensor, we measured its resistance as it gradually becomes more bent. From this we were found that constructing a voltage divider circuit with the flex sensor and a 10K Ohm resistor would provide adequate readings for the analog input of the Arduino. We took raw data from the analog port of the Arduino and determined that 815 is the value of a 90 degree bend and use that data accordingly to set the hand bend gesture.

Challenges

The first major obstacle we had to overcome was using components that would fit a small form factor to make an unobtrusive product. The sensors were not an issue since those would be outside of the housing unit, however the microcontroller, battery, and Bluetooth module sizes all needed to be taken into consideration when we made our design. Initially, we used an Arduino Uno as our microcontroller but that in and of itself was a problem because the single microcontroller was wider than our wrists. The Arduino Nano provided the same functionality that the Uno did but in a smaller, reasonably sized package. Choosing a proper power supply was also a difficult task because we wanted to choose one that could provide a reasonable lifetime while also being small enough for the package.

The other major obstacle was securing the sensors and housing unit onto the glove in the least unwieldy way. We used a Velcro strap to secure the housing unit and it worked out fairly well. Using some glue and sewing, we made it feasible to put the device on with one hand. The sensors were originally attached with electrical tape onto the gloves but that proved to be unreliable as they came off often. We opted to use hot glue to fasten the sensors on.

Future Work and Discussion

This design is very open-ended and has much room for potential growth. There is enough room on the glove to add flex sensors for each finger which would add a significant amount of functionality. In addition, we have room to explore more features that come with using an accelerometer such as more complex, multi-movement gestures. There is also enough space to accommodate a gyroscope sensor which will add more possibilities for our device. However with these changes comes increased costs which must be taken into consideration.

Conclusion

With the rapid growth of wearable devices, SiGlove brings a new dimension to interacting with all of the devices we use on a daily basis. Using a combination of various sensors, the device is able to detect different gestures and send that data to another device via Bluetooth. Using the applications that we built for android and PC, the user can control media, games, and other facets of their device with SiGlove.

REFERENCES

1. Android And Arduino Bluetooth Communication. 2012. Retrieved April 21, 2015 from https://bellcode.wordpress.com/2012/01/02/android-and-arduino-bluetooth-communication/
2. Something Tech.Android Send/Receive data with Bluetooth. 2014. Retrieved April 20, 2015 from https://wingoodharry.wordpress.com/2014/04/15/android-sendreceive-data-with-arduino-using-bluetooth-part-2/