

Sensory Glove

Design Document

CSE-453

Gino Notto, Jason Gregory, Sherif Attia, Fan Meng

TABLE OF CONTENTS

<u>THE SYSTEM</u>	<u>3</u>
<u>HARDWARE DIAGRAM</u>	<u>5</u>
<u>CHARACTER MAPPINGS</u>	<u>6</u>
<u>SOFTWARE DIAGRAM</u>	<u>7</u>
<u>SOFTWARE DESCRIPTION</u>	<u>8</u>
<u>PARTS LIST</u>	<u>11</u>
<u>APPLICATION SCREENSHOTS</u>	<u>12</u>

THE SYSTEM

For most people, we take our ability to see and hear for granted. These two vital senses allow us to communicate with others both verbally and visually. For those who are deaf-blind, this communication is much harder. Our project aims to bridge that communication gap that exists in the deaf-blind community.

The Sensory Glove utilizes very affordable hardware that is readily available. Built using a Raspberry-Pi, some Cat-5 ethernet wire, and 5 Flex sensors by adafruit, we were able to keep the cost of the prototype relatively low when compared to other similar products that exist. All this hardware is all wired through a Gertboard, an I/O expansion board for the Raspberry-Pi.

The Raspberry Pi is a low cost, credit-card sized computer that plugs into a computer monitor or TV, and uses a standard keyboard and mouse. It is a capable little device that enables people of all ages to explore computing, and to learn how to program in languages like Scratch and Python. For our project, it acts as the heart of the device. It is responsible for sensing the input from the flex sensors and communicating that to the Android device. In this prototype, the Pi, the phone, and the sensors are all wired together. However, in the final product the internals would communicate via bluetooth sockets. This is a feature we plan to add later on as we first wanted to focus on getting

a basic prototype working and then expanding that to include features such as bluetooth communication.

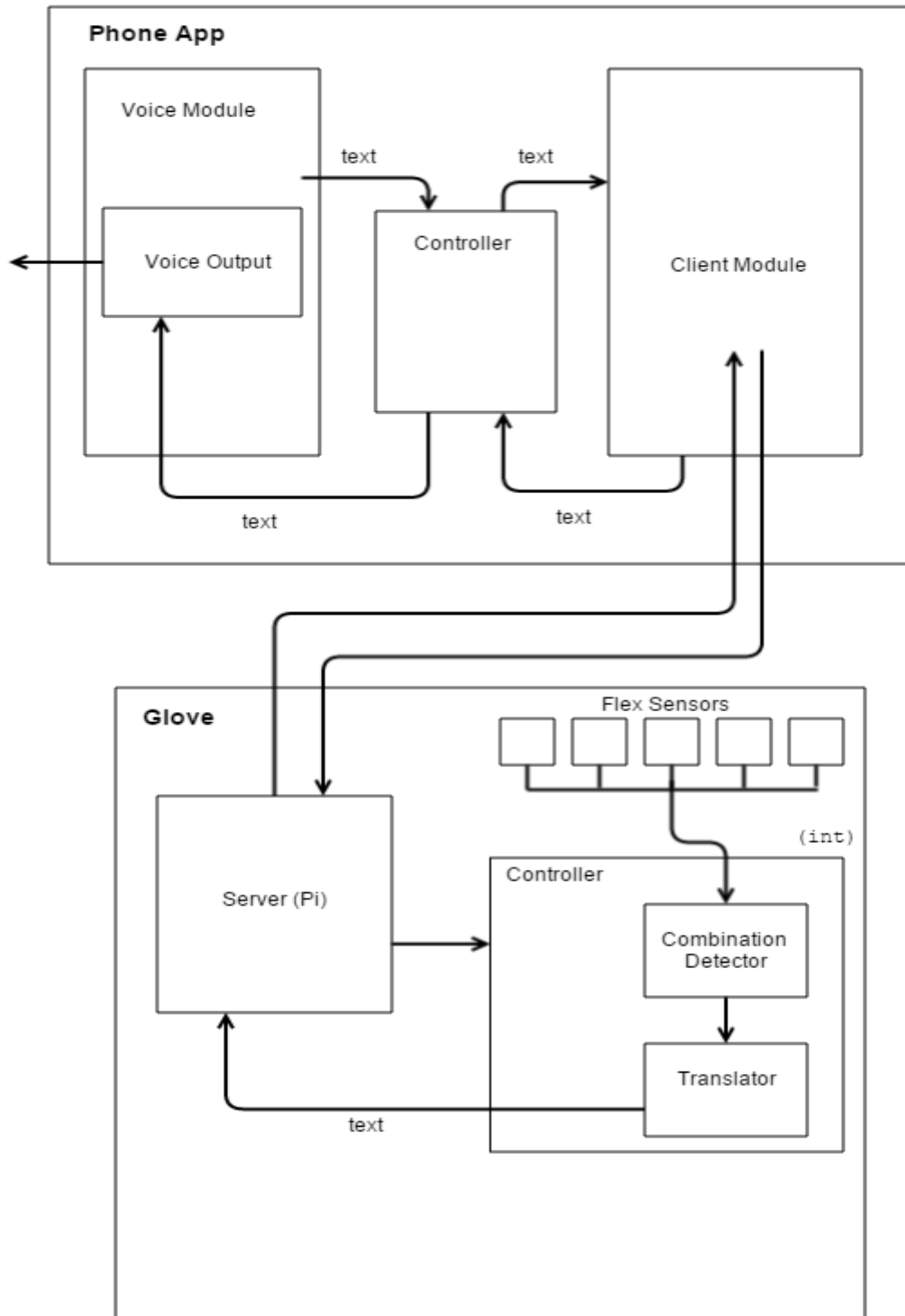
The adafruit flex sensors are ultra-thin and flexible printed circuits, which can be easily integrated into force measurement applications. We read the output of the sensors by connecting them to the Raspberry Pi through Cat5e ethernet wire. We give extra wire between the Raspberry Pi and sensors for the system so it is easier to maneuver.

However when a final product is created a small battery powered microchip would make it mobility and ease of use even better.

With five sensors attached to a glove and connect to a Pi, the user can perform a series of finger movements that map to characters of the alphabet. These characters are then sent to the Android device which utilizes Google's Text-to-Speech API to speak the words out loud.

This prototype, despite being very basic, served as an excellent device for testing and validating our initial idea. Now that we have a basic understanding of the hardware and a much more realistic set of requirements than what we originally had, we can use this prototype as a stepping stone to building a device that can truly bridge the gap of communication that exists in the deaf-blind community.

HARDWARE DIAGRAM

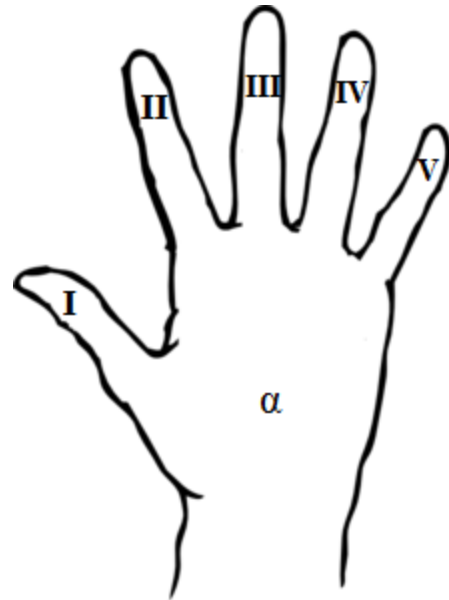


CHARACTER MAPPINGS

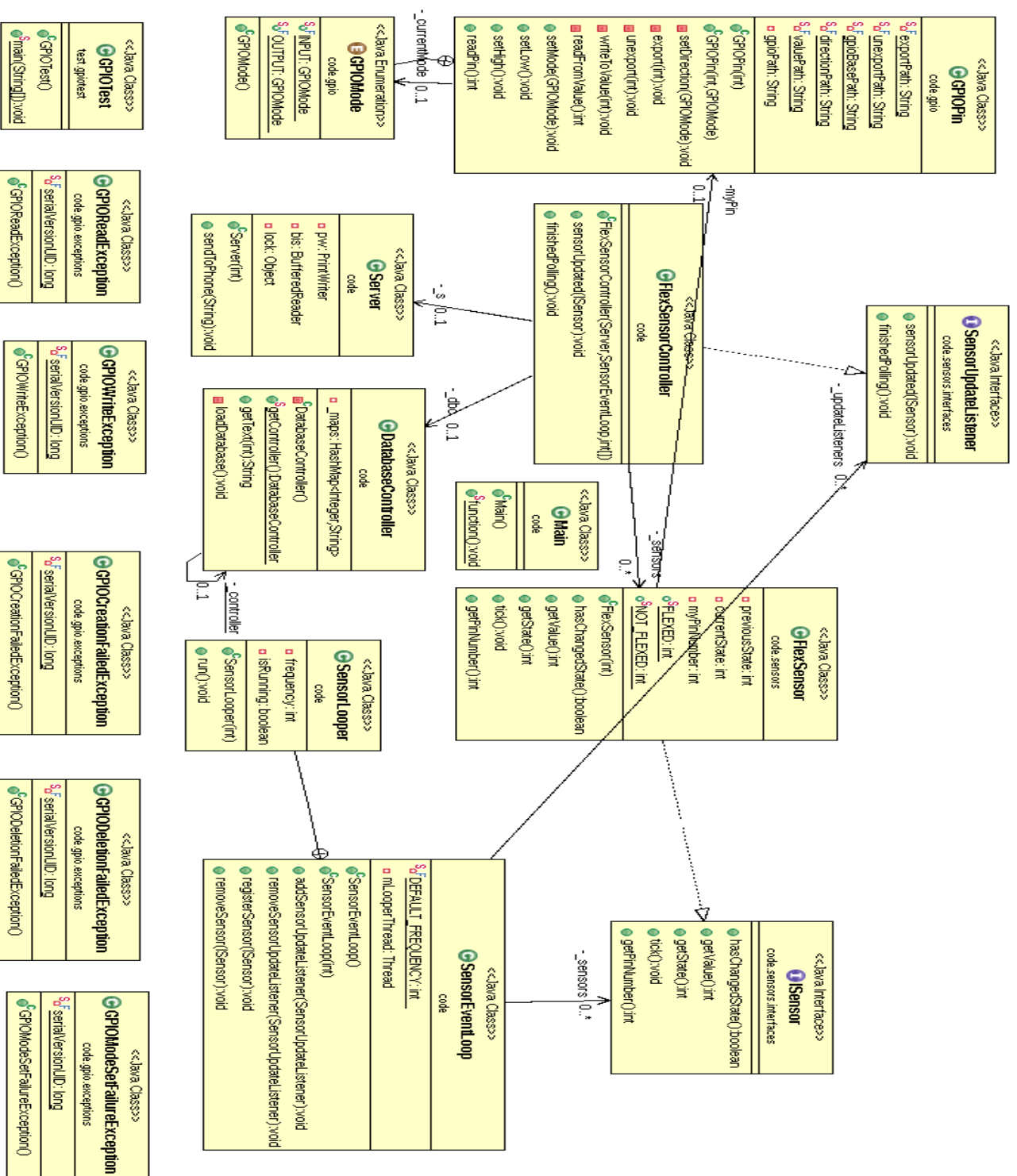
0 - finger straight
 $\alpha=0$ - palm facing down

1 - finger bent
 $\alpha=1$ - palm facing up

	I	II	III	IV	V	α
A	0	0	0	1	1	0
B	0	0	1	1	0	0
C	0	0	1	1	1	0
D	0	1	0	0	0	0
E	0	1	0	1	1	0
F	0	1	1	0	0	0
G	0	1	1	1	0	0
H	0	1	1	1	1	0
I	1	0	0	0	0	0
J	1	0	0	0	1	0
K	1	0	0	1	0	0
L	1	0	0	1	1	0
M	1	0	1	0	0	0
N	1	0	1	1	0	0
O	1	0	1	1	1	0
P	1	1	0	0	0	0
Q	1	1	0	0	1	0
R	1	1	0	1	0	0
S	1	1	0	1	1	0
T	1	1	1	0	0	0
U	1	1	1	1	0	0
X	0	0	0	1	1	1
Y	0	0	1	1	0	1
Z	0	0	1	1	1	1

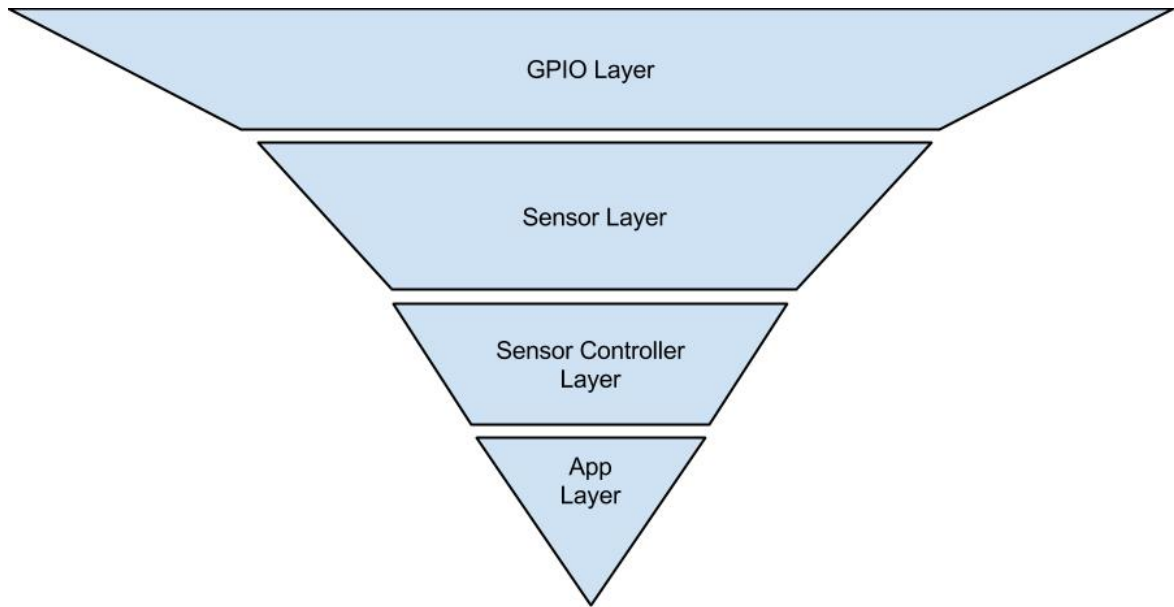


SOFTWARE DIAGRAM



SOFTWARE DESCRIPTION

The software is built upon the idea of abstractions. The software first aims to abstract away the lowest details by creating a sensor base class. Each sensor is then derived from this class. On the next level, we have the sensor driver. This is a real time feedback loop that polls the sensors at a set rate. For this particular project, we are polling and updating the sensors at a rate of 30Hz. This event loop employs the Observer pattern for a robust and easily extendable system by further abstracting away the fact that there is a piece of hardware at all. By registering an update listener, the user can receive asynchronous updates through the dedicated method *public void sensorUpdated(ISensor)*; From there, handling each sensor can be done via polymorphic code to reduce code smell and improve maintenance and readability of the code. At this level in our abstraction, you are given some easy accessor methods to tell you some information such as whether the sensor changed or the value of the sensor. Using this top down approach, building applications on top of these in-house library is relatively easy as it is backed by a large code base underneath. See the diagram below.



The Software Abstraction Diagram

GPIOPin - This class deals with bridging the hardware and software gap. It abstracts away dealing with the files that point to the actual pins as well as provides error handling and detection.

Sensor Looper - This class creates a new thread that manages the event loop. It works in a “plug-and-play” fashion such that sensors can be registered or deregistered in real time.

ISensor - The template of which each sensor is implemented with. This generic interface allows for polymorphic code rather than cumbersome and broken imperative code using large switch statements. It also allows any sensor to be compatible with the Sensor Looper.

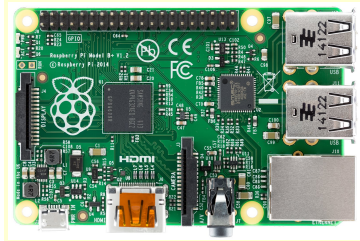

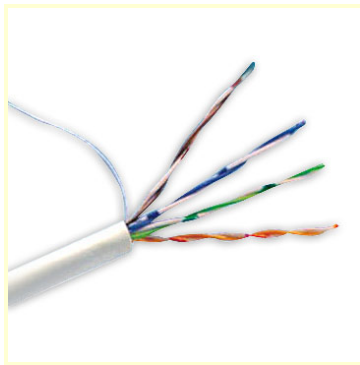
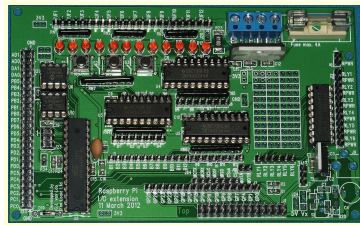
FlexSensor - This is the actual implementation of a flex sensor and is specifically for it. It provides a differential range that stops it from swinging from one state to the other.

Flex Sensor Controller - This is at the application level. It ties n flex sensors together and manages their state. Once all sensors have been polled and updated, it then turns all of their combined states into a byte. For instance, if 2 sensors are bent and 3 aren't then the result may be something like *10010*. This byte can then be sent easily to a server or to the DatabaseController.

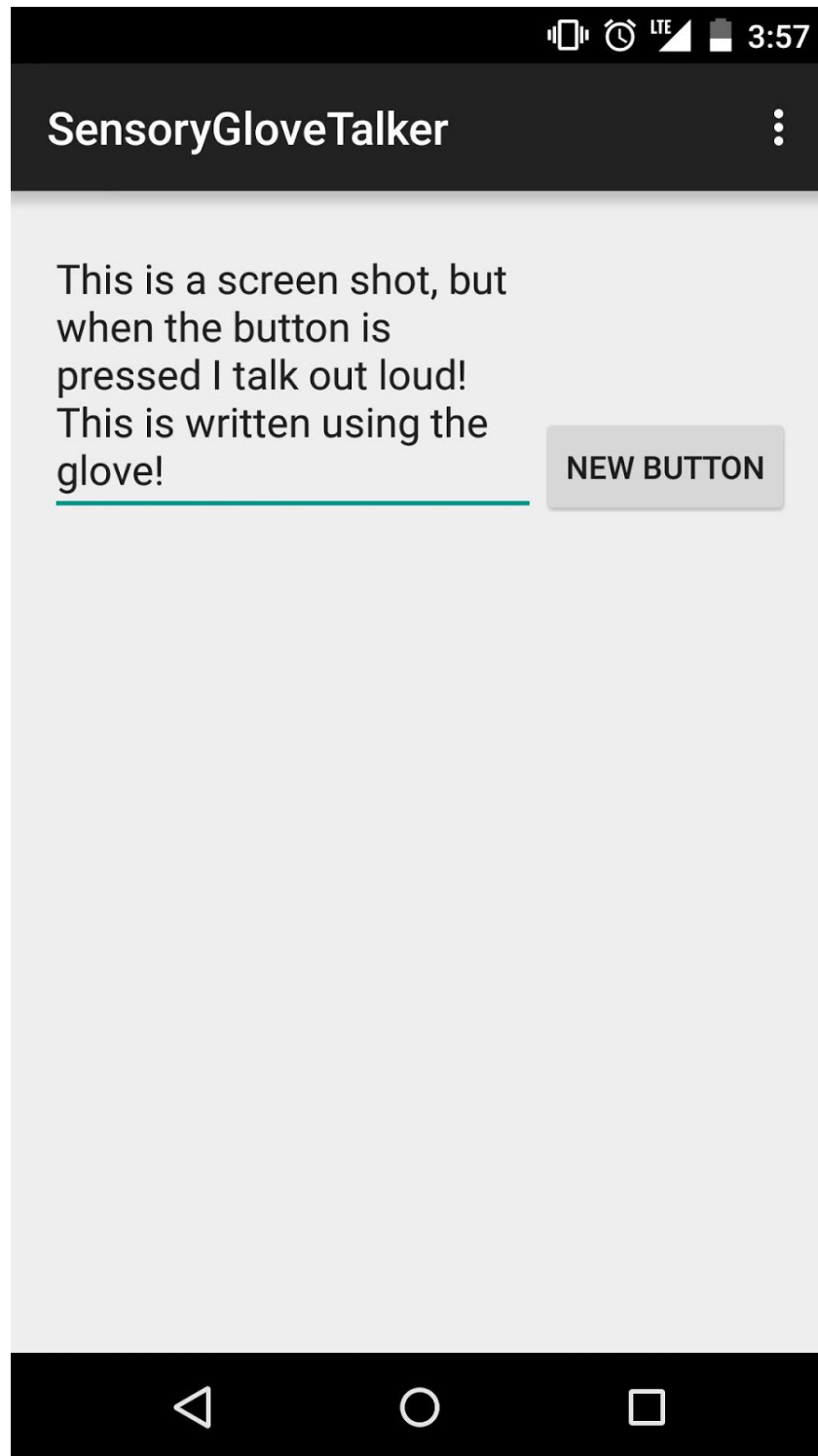
DatabaseController - This loads in the mapping database and allows for bytes to be translated to words.

Server - This class creates a server for which the phone can connect to all. All connected devices will receive updates in the form of well formatted JSON objects which contain the string that was "typed" using the glove. This allows for a wide range of phone or desktop applications to communicate with the device.

PARTS LIST

Manufacturer	Part Number	Price	No.	Link	Image
Raspberry Pi	RASPBERRY-P I-STK	\$34.99	1	goo.gl/tSs0Aj	
Short Flex/Bend Sensor by adafruit	adafruit Product ID: 1070	\$7.95	5	http://goo.gl/tKrYrS	
Cat5e	CPA-1276-1	\$7.45/ 100ft	1	goo.gl/egDjGn	
MCM Gertboard	83-14460	\$39.99	1	goo.gl/MiJdyJ	

APPLICATION SCREENSHOTS



REFERENCES

1. http://www.petervis.com/Raspberry_PI/Gertboard_Raspberry_Pi_Expansion/Gertboard_Analog_to_Digital_Converter.html
2. http://www.petervis.com/Raspberry_PI/Gertboard_Raspberry_Pi_Expansion/Gertboard_GPIO_Pins.html
3. <http://www.mcmelectronics.com/content/ProductData/Manuals/83-14460.pdf>
4. <https://www.raspberrypi.org/documentation/hardware/raspberrypi/spi/README.md>
5. <https://www.sparkfun.com/tutorials/389>
6. <https://www.youtube.com/watch?v=2rWkdftDh5o>