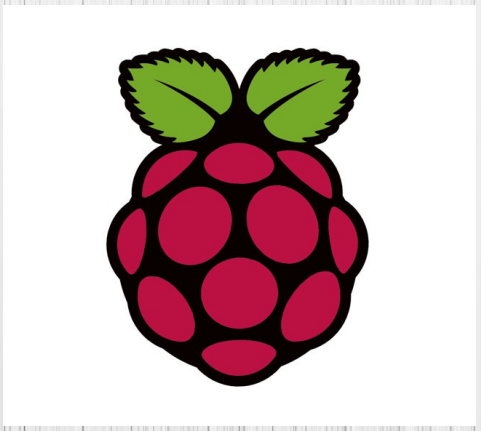
# Finale

LEAP MOTION



## Contents

I. Aim of the Project
II. Background Information
III. Procedure
IV. Results
V. Conclusion

# Aim of the Project

- Simply put, the aim of my project was to create a program that would allow for the control of an RGB led using only hand gestures.
- I was to write a program for a “host” PC connected to the Leap Motion motion controller that would allow for gesture input to a remote Raspberry Pi.
- I was then to create a program for the Raspberry Pi that would accept these inputs and control an RGB led.
- My project was to include the ability to turn the led on, turn the led off, change the color of the led and dim the led.
- A self-imposed restriction on my project was to make it as simple, approachable and user-friendly as possible.

# Background Information: General

- The first and possibly most important thing to understand about this project is that the Raspberry Pi does not directly support the use of the Leap Motion. I discovered this early enough to be able to formulate a new plan.
- I decided that the best way of mitigating this possibly project-breaking issue was to separate the Raspberry Pi and Leap Motion controller; to have a communication protocol active between a PC (connected to the Leap) and the Pi.
- In this way, I was able to effectively control the Pi's GPIO functions through the use of the Leap Motion.

# Background Information: Leap Motion

- The Leap Motion controller is a device that tracks the motions and gestures of a user's hands and uses this information to control myriad functions of a PC.
- The controller was released in July 2013 and came with it a suite of programming tools.
- The SDK provided to developers can be programmed using various languages including C, Python and Java.
- I decided to write my Opus in Java given my aptitude with the language and the wide availability of free, open-source IDE's.

# Background Information: Raspberry Pi

- The Raspberry Pi's pinout has 17 General Purpose Input/Output(GPIO) pins.
- GPIO pins are a processors way of interfacing with peripherals and allow for the transmission and acceptance of external signals.
- By default, control over these pins is only granted to superusers and requires the manual modification of the “gpio” file on the Pi.
- There is however an amazing suite of utilities called wiringPi that allows for much easier control over the Pi's GPIO capabilities.
- WiringPi is written in C, so I needed to brush off the C cobwebs and write my GPIO controlling programs for the Pi.

# Procedure

- The first step in accomplishing my goal was to devise a plan of attack that would make all the moving parts fit.
- At one point, I had imagined a suite of small programs on the Pi, each controlling a separate function I wished to perform on the leds.
- It quickly became clear that this was a bulky, unwieldy approach, so I opted for one big program that would allow for the control of each function.
- I wrote this program in stages, first working on getting the led to turn on and off using keyboard inputs.
- I then began working in the ability to change the color of the led.

# Procedure

- Finally, I utilized pulse width modulation to control the brightness of the led.
- I then needed to concoct a method of controlling all these functions using simple inputs, as it was my final goal of controlling the program not with a keyboard but the Leap Motion controller.
- I settled on a number system. Each color was coded to a number 1-7.  Any number above 7 (up to 100) would be considered a brightness level.
- This allowed for minimal code and simple controls.

# Procedure

- I then began working on the PC end of things.
- I broke out Eclipse and got a simple sample application provided in the Leap SDK up and running.
- The sample program tracked and output information on every frame/gesture captured by the Leap Motion.
- I started by stripping out all the code I knew was unnecessary for my purposes.
- I then configured the program to perform specific actions whenever a gesture was recognized.

# Procedure

- It then came time to decide how I would remotely connect to the Pi in my Java program.
- I landed on the SSH protocol and after many failed attempts at implementing it in my program, I discovered JSCh.
- JSCh is a set of tools written in Java that allow for simple integration of the SSH protocol in java programs.
- After importing the libraries into my project, I began working with a sample program provided by JSCh that allowed for shell access and control.

# Procedure

- After integrating the SSH protocol into my Java program, I began the long process of debugging my code.
- Finally, without going into too much detail, I was able to connect to the Rpi and send commands using the Leap Motion controller.
- The last step was making sure the project would function correctly on school grounds.
- I was unable to connect to the Rpi wirelessly, but with a few tweaks I was able to successfully demonstrate my project using a LAN cable.

# Results

- This project was challenging and at times very frustrating.  This however made the final experience of seeing my project function properly all the more satisfying.
- I was ultimately able to implement all the functions I had originally intended: the ability to turn the led on and off, 7 distinct colors the user can switch between using swipes, and the ability to dim the led. And all of this in a discrete, easy to use package.
- Given the restrictions placed on my project, self-imposed or otherwise, I'd say it was a – if not resounding – success.

# Conclusions

- It goes without saying that the Raspberry Pi and the Leap Motion controller are devices of immense power and potential.
- The Pi has established itself as a staple in the computing world, a device experts and hobbyists alike need to have in their toolkit.
- The Leap Motion, while a new and relatively untested product, has incredible potential for projects large and small.  It just takes some effort and imagination.
- While my project was not earth-shattering, it is a good indication of the Leap's capabilities. If the Leap can control a led, why not a robotic hand utilized in a surgery, or a space-faring rover exploring the outer reaches of our galaxy.

# Conclusions:
# What I Learned

- The most important lesson I learned working on this project was how to work with pre-made code.
- Previously, I'd only ever written my own code. I had control over what everything did and, more importantly, I knew exactly how everything worked.
- Utilizing an SDK means you have to work within the confines of what others have previously coded.
- I technically used two SDKs, one for the Leap Motion, and another for JSCh. I also used the wiringPi module.  Each of these packages had their own methods and restrictions.
- If I was to successfully complete my project, I needed to learn to work with/for people I'd never met, nor would I meet in person.

# Conclusions: What I Learned

- Another important skill I developed was the use of the Linux command line.
- I had previously worked in small measure with the Windows command line, but never to the extent of the command line in Linux.
- I also brushed up on my C and Java coding skills, both of which proved invaluable to my project's success.