Cyber Physical

Creating a Secure Compartmentalized Information Facility

Chris Soukup, Ross Jones, Dylan Garvis, Jonathan Sun, Nate Kibler, Jake Sperlazza

December 12, 2014

# Abstract

In this paper, we describe the development of a Sensitive Compartmented Information Facility (SCIF) that uses USB read times as a security feature.

# 1 – INTRODUCTION

Users concerned with creating a secure location use a SCIF. A SCIF provides the significant level of security necessary for classified work.

# 2 -SYSTEM OVERVIEW

The complete system is made up of at least three main server servers, with one more more secure work rooms that contain Pis and IO Controllers. A diagram of the system can be found at 8.3.

## 2.1 - The Servers

The system has at least three servers. An authentication server, boot server and virtualization server.

### 2.1.1 - Authentication Server

The authentication is a Java Spring web app designed to run on the GlassFish platform with a MySQL database supporting it. The application handles validating parameters supplied by the Pi and the user when attempting to log in.

The following parameters are supplied to the web app by the Pi:

* Username: A username inputed by the user.i
* Records: 50 read times of the key.
* Device ID: The device ID of the key.

The advised hardware for the authentication server is an i7 processor, 4GB of memory, gigabit ethernet port, and 150GB of hard disk space.

With the advise hardware, most requests can be handle by the authentication server within 90ms, assuming no more than 100 users are attempting to login at the same time. See chart 8.2 for the breakdown of larger simulations logins.

### 2.1.2 - Boot Server

The boot server supplies a common image for all of the Pis to network boot off. Each Pi accesses, loads and boots the image on the server on boot up. This allows the administrator to update all images on the Pis by updating the image on the server and rebooting the system.

The server could be virtualized but it is recommended to have a physical server for high number of clients. Within a hundred clients only a server with a 2.6GHz CPU and 2GB of RAM is recommended. Over a thousand clients a server with four 3GHz CPU with 4GB of RAM is recommended. These recommendations are made to keep the boot time to a minimum.

### 2.1.3 - Virtualization Server

The virtualization server supports several virtualized machines for which the Pis connect to on a successful authentication. There can be more than on virtualization server based on the amount of virtual machines needed.

The virtualization server is powered by VMware. The hardware for the server depends on the requirements of the user. If each user is given a install with a 25GB virtual hard drive disk, 1 Virtual Core, and 1GB of memory. Then the hard drive requirements should be 50GB. for the base operating system, plus 25GB per user. For the processor, since the users will be working on them full time, each virtual core should map to a physical core. Lastly the memory should have 2GB, for the base install, plus 1GB per user.

## 2.2 - The Secure Work Room

The secure work room, of which their may be one or many, is a room with no actual hardware accessible to the users except a mouse, keyboard, monitor, and a key plugin. The Pis, which act as then thin client for the user sessions, are placed in such a way that the IO devices can connect to them, but can not be easily accessed by unauthorized parties. The Pis also have had an IO controller attached to them to allow the suppression of all IO in the room at the press of a button.

# 3 – VNC

VNC (Virtual Network Computing) is used to between the thin clients and the virtualization server. The virtualization server has several virtualized desktop environments. The desktop environments are set up to allow a secure shell connection to them from the thin clients with x forwarding support. After a secure shell connection is formed to the virtual machine, a VNC Viewer is invoked to display the users desktop environment to them.

## 3.1 - Persistence

Since the desktop session are kept in virtual machine on the virtualization server and VNC is used to connect to them, when a user chooses to or is forcibly disconnected from their desktop environment the unsaved work is not lost. When the user reopens the connection, the desktop environment will be as it was left when the user was disconnected. (excluding temporal tasks)

# 4 – PROFILING USB

## 4.1 – Overview

Profiling a USB key to be used for authentication is a multi-step process that involves several layers of security. A single user account (tied to a single VNC instance) has exactly one USB key that is able to authenticate a user. This means that a single USB key may be used for multiple accounts, allowing a user to own multiple VNC instances that are possibly running different operating systems.

**4.2 – USB Recognition**

The time it takes to recognize a USB device is largely dependent on the Pi’s device drivers. Our program is set to check for the USB device every 0.5s. *Table 3.1* shows the average time it takes from when you plug in the USB device to the time you are prompted for a username during login, as well as the time it takes from when you pull out the USB device to when you are logged out.

|  |  |
| --- | --- |
| **Time to Recognize for Login (s)** | **Time to Recognize for Logout (s)** |
| 4.52 | 0.50 |

*Table 3.1*

## 4.3 – Initial Profiling

The USB key that is associated with a user account only needs to be initially profiled once. This will create the linkage between the USB key, the user account, and the VNC server instance created specifically for the account. The initial profile runs a *dd* command that takes in a 1KB file and sends it to */dev/null*. It runs this command 200 times, which is configurable. These times, along with the username, device ID, VNC server string to hook the account to, and a secret key string, are sent to the authentication server for storage. The device ID is read from the *lsusb* command, which is a combination of the vendor ID and device ID of the actual USB storage device. The VNC server string is an IP address and port number that will be returned back on a successful login. The secret key string is a random key that is issued to each user for a one-time use to create a profile. This prevents any random requests to profile a device from creating an account if the user isn’t authorized to do so. *Table 3.2* shows the average time it takes to perform the device time profiling, as well as the percentage of the total time (round-trip to the auth server).

|  |  |  |
| --- | --- | --- |
| **Avg. Profile Time (s)** | **Total Profile Time (s)** | **Percentage of Total** |
| 31.41 | 34.29 | 91.6% |

*Table 3.2*

**4.4 – Secondary Profiling**

Secondary profiling occurs each time the user attempts to log in to their account. The user plugs in their USB key and enters their username. The USB key is then profiled (through 50 *dd* commands, also configurable) and the times (along with the device ID) are sent to the auth server for verification. If the login was successful, the associated VNC server string is sent back, at which point the Pi connects to the VNC instance using the server string. If the login was unsuccessful, the appropriate error message is sent back and displayed to the user. After 10 seconds, the program is reset and the user has the option to enter their username and login again, assuming their USB device is still plugged in. *Table 3.3* shows the time it takes to perform this secondary profiling, the total round-trip time to the server, and the percentage the profiling takes of the total time.

|  |  |  |
| --- | --- | --- |
| **Avg. Profile Time (s)** | **Total Auth Time (s)** | **Percentage of Total** |
| 8.10 | 8.35 | 97.0% |

*Table 3.3*

**4.5 – Observations & Complications**

During our research for determining the best approach to uniquely identify a USB device, we found that one possible way was to take advantage of the unique properties of the hardware of the device. Specifically, we implemented the read speed profiler to try to build a unique characteristic profile for each USB key. We found through several trials that read speed cannot be trusted to be unique, nor consistent. Our allowance on the auth server side was drastically increased to prevent false negatives when authenticating a device. The problem with this is that false devices have an increased chance of being falsely authenticated, assuming they pass all of the other layers of security (username, spoof device ID, VNC password, etc.). Purchasing a higher-quality and more consistent flash drive may fix this issue, but for the budget parts we are using, we could not reliably use the read speeds for authentication.

# 5 – PHYSICAL COMPONENT

**5.1 – Overview**

The physical component of the system is responsible for controlling the connection of required I/O devices based on whether or not the user is authenticated or not. A list of parts used is included in the appendix.



Figure 5.1 - The Physical Component

**5.2 – Arduino**

The Arduino 2560 serves as the main control unit for the physical component of the system. It is used to control the relays to manipulate the KVM switches and communicate with the Raspberry Pi. The Arduino also communicates with the Pi about the state of the system using 5v logic. A constant 5v signal is being sent when the system is operational. If the kill button is pressed, the signal is dropped. This arrangement was chosen so that in the event of a loss of connection, it fails safe to being locked.

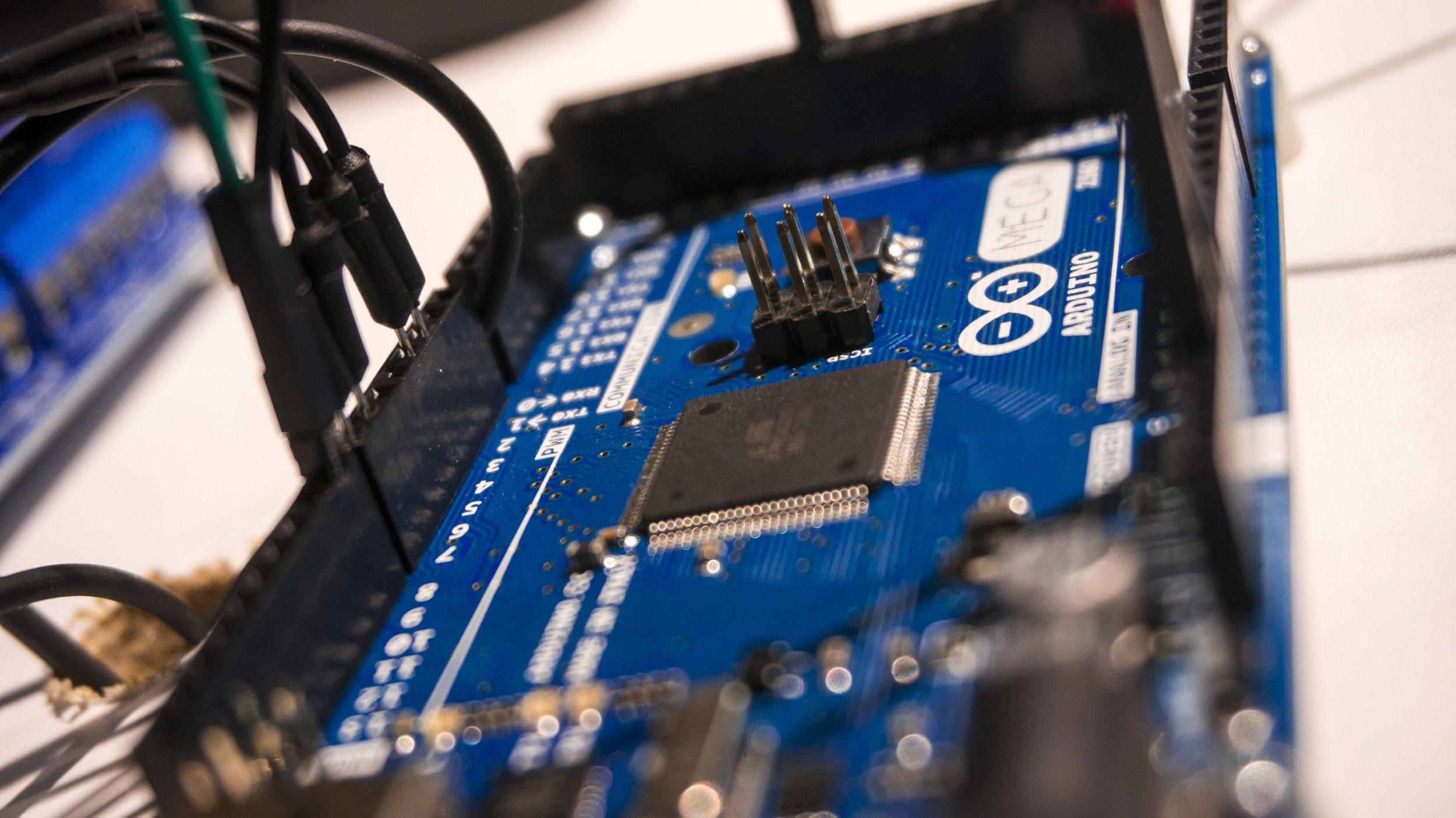


Figure 5.2 - Arduino 2560

**5.3 – Raspberry Pi**

The Raspberry Pi is primarily used to authenticate the USB key as well as provide access to a users session. The Pi also communicates with the Arduino about the state of the system using 5v logic. If a user is authenticated, the Pi will send a constant 5v signal to the Arduino. If a user is not authenticated, no signal is sent. This arrangement was chosen so that in the event of a loss of connection, it fails safe to being unauthenicated.

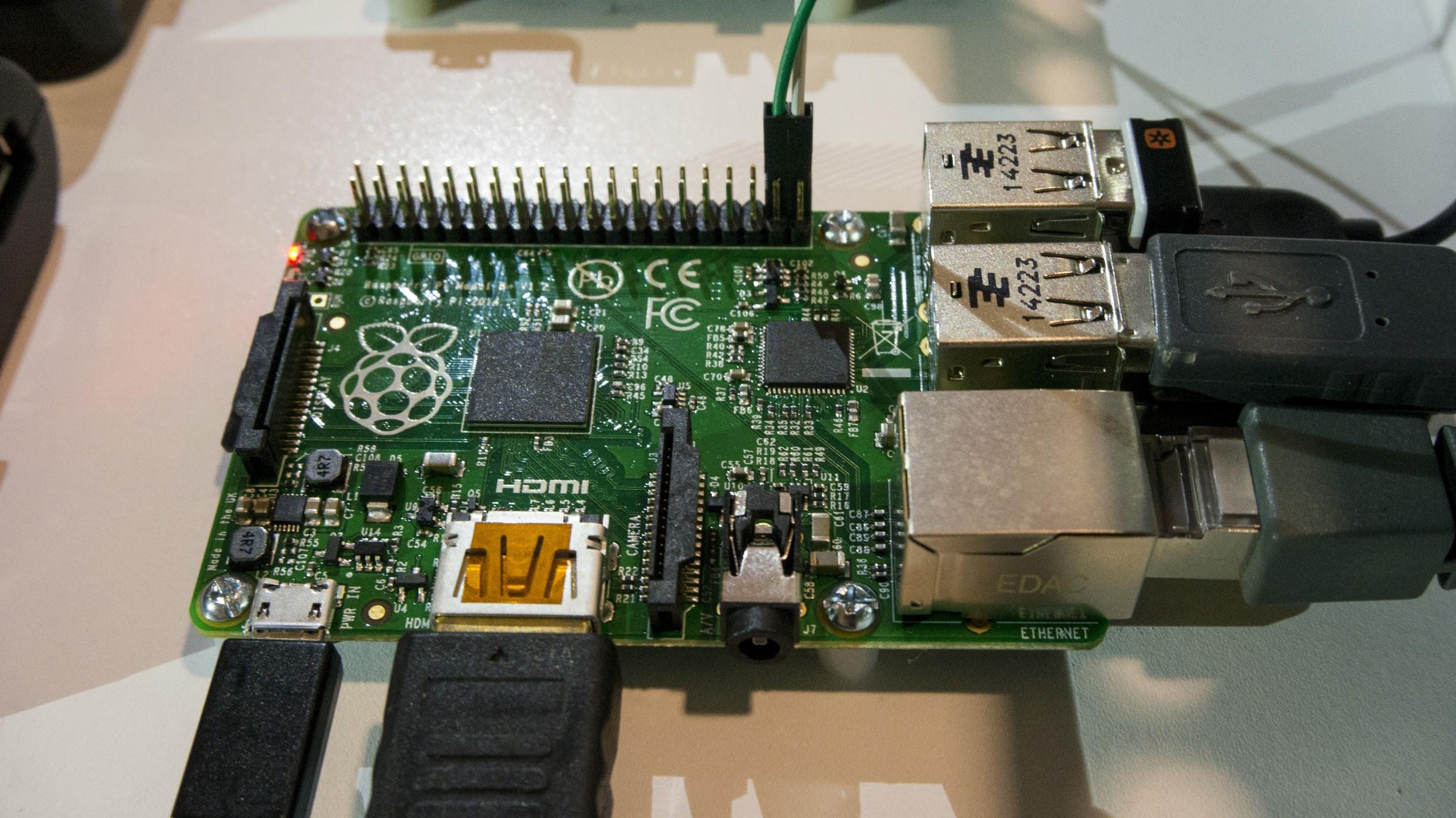


Figure 5.3 - Raspberry Pi

**5.4 – KVM Switch**

The KVM switch is used to control the I/O connections to the Raspberry Pi. One output of the KVM is connected to the Pi, while the other is left unconnected. When the user is authenticated, the KVM routes the I/O devices to the Pi. When the user is not authenticated, the KVM routes the I/O devices to the unconnected side which effectively disconnects the I/O devices from the Pi.



Figure 5.4 - KVM Switches

**5.5 – Relays**

A relay is used to manipulate each KVM switch. The KVM switch came with a button that when pressed would switch which output the KVM routed to. By using a relay to simulate a button press, the Arduino is capable of switching the KVM output.

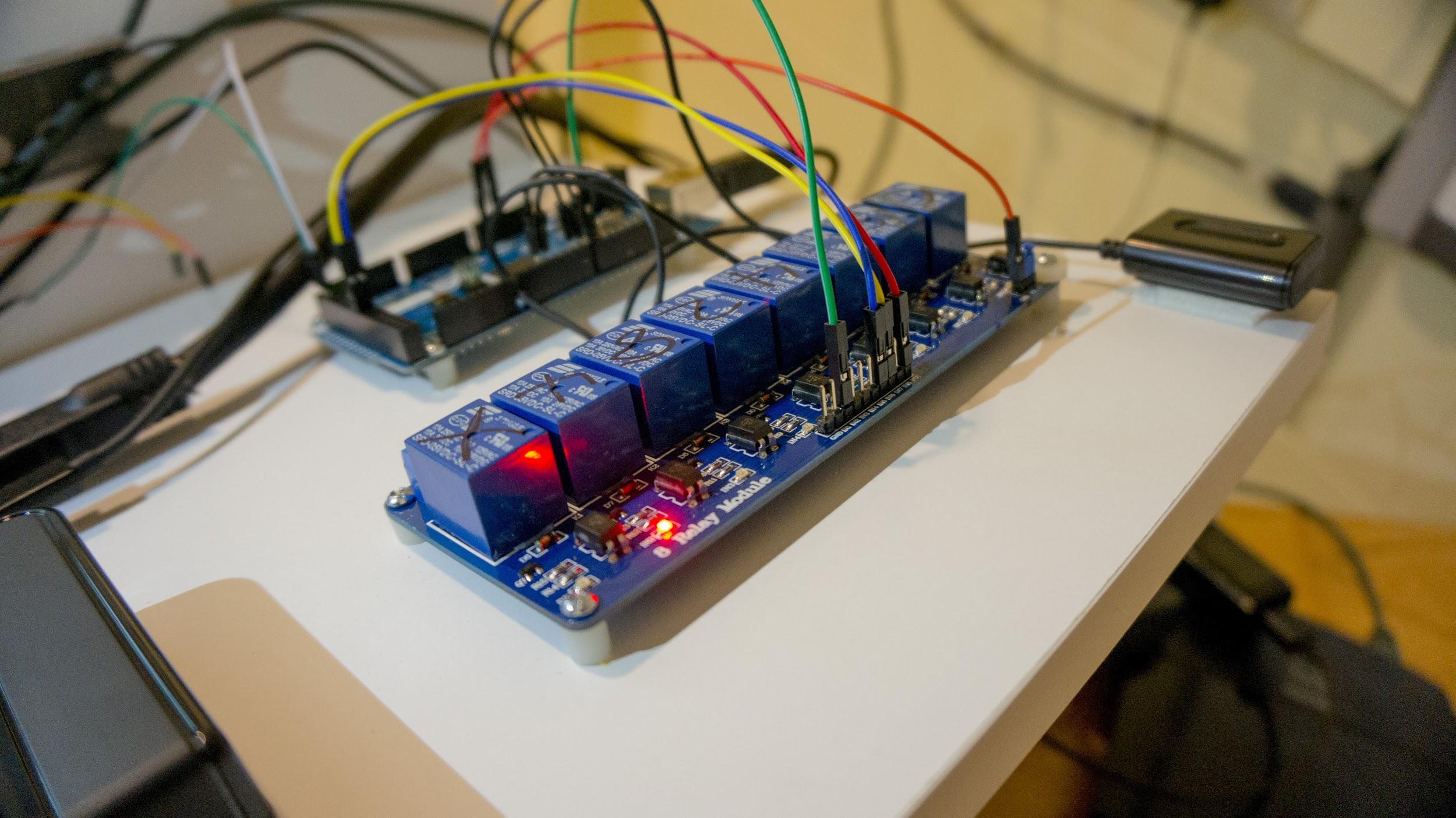


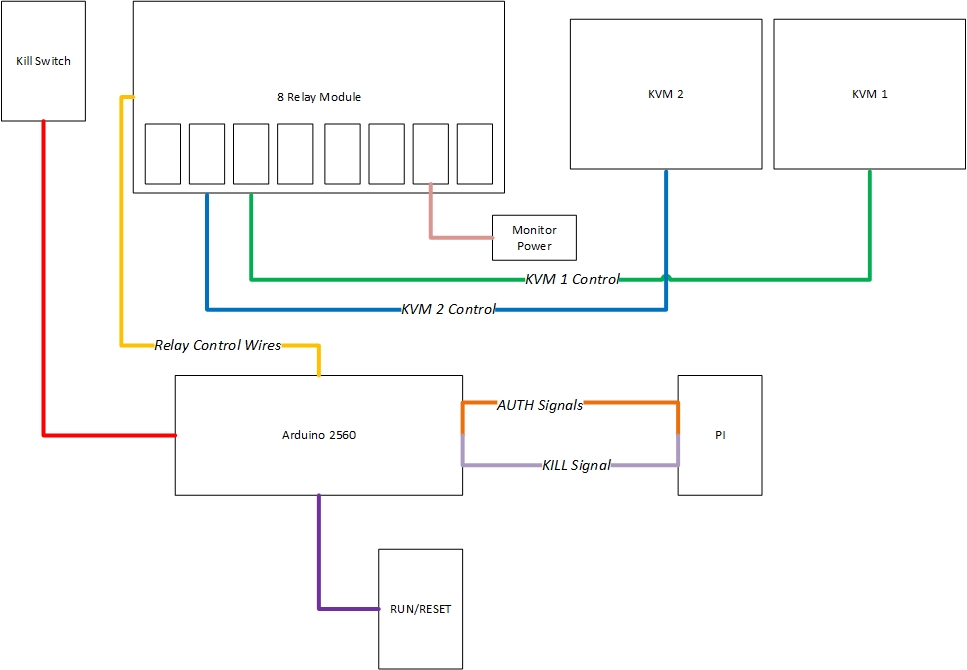
Figure 5.5 - Relays

**5.6 – Problem Areas**

The physical set up we created for our SCIF has some concessions due to monetary constraints. We obtained VGA KVM switches rather than HDMI switches even though the Raspberry Pi only has HDMI output because VGA KVMs are significantly cheaper. Conversion between VGA and HDMI is inconsistent at best so we resorted to simulating what would happen when the kill switch is pressed. Rather than the KVM switching and the monitor and keyboard not working, just the keyboard stops working and the monitor displays a locked message. The user is not given access to anything aside from a keyboard, a monitor, and a USB port. The USB port is for the USB security key. The monitor and keyboard only work when a valid USB key is in and the user has not been unauthorized for some reason.

Another potential problem area is the use of 5v logic. If either the Pi or Arduino begin sending signals randomly due to a malfunction, there is no way to determine which signals are legitimate. This could be solved by using a protocol for communication. However, for purposes of this project it was determined that 5v would suffice as a proof of concept due to the increased complexity of implementing a protocol.

**5.7 – Wiring Diagram**



# 

# 

# 6 – BOOT

### **6.1 – Network Boot**

The full following data can be viewed in the table appendix 8.1. The average time for the Raspberry Pi to boot the custom image (1.83 GB) is 15.64955 seconds from the SD card. When doing a server boot, there is additional time to transfer the boot image from the server to the Pi. Through direct Ethernet connection, the transfer time was on average 18.2348 seconds. When connected to a wireless router the transfer time would range from 2-10 times the time of the Ethernet connection based upon the router traffic. When booting ten plus simulated Pis simultaneously on a simulated server, the transfer times could spike up to 100 times the Ethernet connection time. This time spike is due to the small limit of resources available on the testing machine. To achieve the faster boot time, it is recommended that each Pi have a direct Ethernet connection to a physical boot server. This would provide an average of 33.5 seconds for a complete boot.

**6.2 – Boot Server**

Each Raspberry Pi runs U-Boot which boots the Pi from an image on the network at a given IP address. The boot server is located at the given IP address and only holds the image.

**7 – SUMMARY**

Our SCIF serves more as a proof of concept rather than a functional SCIF due to concessions because of monetary concerns. However, it is possible to create a SCIF using our design if the correct parts are used.

**8 – APPENDIX**

|  |  |  |
| --- | --- | --- |
| **Part** | **Price** | **Source** |
| Arduino Mega 2560 R3 | $45.95 | https://www.sparkfun.com/products/11061 |
| Wosang Solderless Flexible Breadboard Jumper Wires M/M 100pcs | $6.00 | http://www.amazon.com/Wosang-Solderless-Flexible-Breadboard-Jumper/dp/B005TZJ0AM/ref=sr\_1\_2?s=electronics&ie=UTF8&qid=1412878259&sr=1-2&keywords=wire |
| Opto-Isolated 8 Channel Relay Board | $20.00 | http://yourduino.com/sunshop2/index.php?l=product\_detail&p=156 |
| 2 x IOGEAR 2 Port USB Cable KVM Switch with Audio and Mic (GCS72U) | $21.99 x 2 | http://www.amazon.com/IOGEAR-Cable-Switch-Audio-GCS72U/dp/B002K0TU2C/ref=sr\_1\_1?ie=UTF8&qid=1412878032&sr=8-1&keywords=kvm+switch |
| 2 x 512MB Raspberry Pi Model™ Model B+ Project Board | $35.00 x 2 | http://www.mcmelectronics.com/product/83-16317?utm\_campaign=e14&utm\_source=e14community&utm\_medium=forum&utm\_term=83-16317&COM=superwidget-link\_RaspberryPi |
| Wall Adapter Power Supply - 9V DC 650mA | $5.67 | http://www.amazon.com/Wall-Adapter-Power-Supply-650mA/dp/B003XZSZWO/ref=pd\_bxgy\_pc\_text\_y |
| **Total** | **$191.60** |  |

**8.1 – Boot Times**

|  |  |  |  |
| --- | --- | --- | --- |
| **Times (sec)** |  |  |  |
| **Boot** | **Transfer Time**  **Ethernet** | **Transfer Time**  **Wireless Router** | **Transfer Time**  **Wireless Router (10 Pis)** |
| 15.6495 | 18.2563 | 38.4521 | 1786.3249 |
| 15.6496 | 18.2172 | 63.2563 | 1239.4829 |
| 15.6495 | 18.2423 | 40.5693 | 1723.8492 |
| 15.6496 | 18.2378 | 130.3049 | 1593.3827 |
| 15.6495 | 18.2483 | 83.2199 | 1329.3835 |
| 15.6495 | 18.2212 | 173.7489 | 1574.3923 |
| 15.6495 | 18.2319 | 128.3411 | 1330.4728 |
| 15.6495 | 18.2198 | 78.2398 | 1662.3948 |
| 15.6496 | 18.2301 | 114.6849 | 1821.1839 |
| 15.6496 | 18.2431 | 154.8392 | 1439.2837 |

**8.2 – Auth Server Request Time**

|  |  |
| --- | --- |
| **Requests Made**  **(started within a 10sec window)** | **Average Request Time (ms)** |
| 1 | 50 |
| 100 | 91 |
| 500 | 135 |
| 1000 | 5911 |

**8.3 – System Overview**

