Final Project Report: Secure Seat

Christopher Wolf

ITE 485

12/06/2017

Professor Van Devender

**Executive Summary**

Secure Seat will ensure that employees secure their computers with a password by doing it for them automatically when they leave their seat. Unlocked desktops are a major threat vector for networks in any office environment. My device seeks to solve this problem by doing it for employees automatically.

The Secure Seat will utilize a cushion with a pressure sensor, which is then connected to a Raspberry Pi running the Ubuntu operating system. Once an employee leaves their chair the sensor will send input to a Python program running on the Raspberry Pi. It will then send a signal via USB On the Go emulating a HID keyboard. The keystrokes will then send the windows command to lock the desktop.

I will be utilizing a flex force sensor connected to the Raspberry Pi GIO header to convert analog input into digital input. This is achieved by an analog to digital converter installed to the board of the Raspberry Pi. The software has a master configuration file that allows an administrator to configure the timeout period by editing a timeout.txt file on the Raspberry Pi device.

The added security provided by my device will benefit all stakeholders in the office and if properly configured, increase security with no impact to usability.

**Problem Statement**

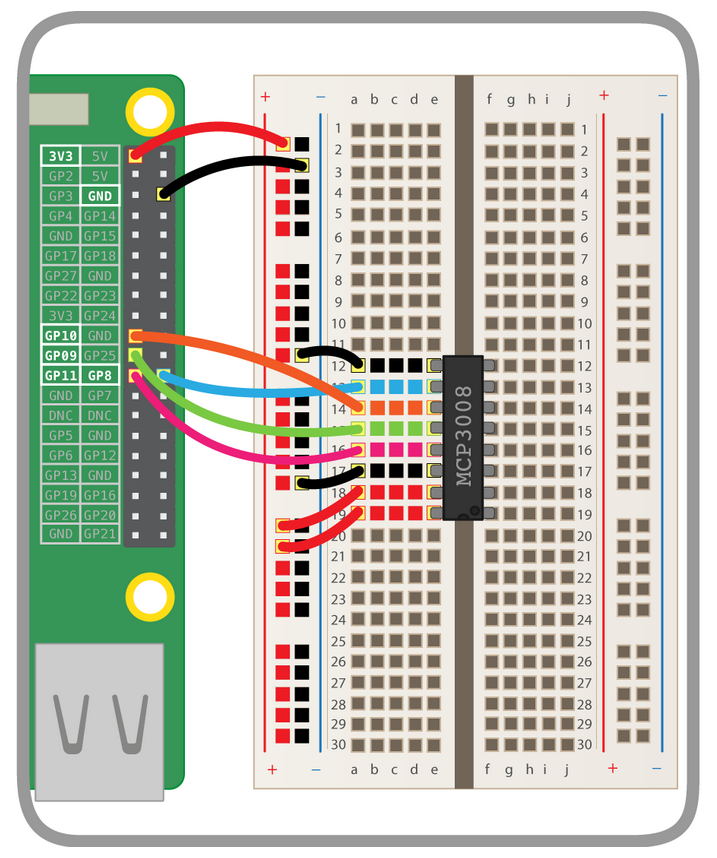
My device seeks to solve a common problem almost every network administrator has dealt with on a frequent basis: unsecured work stations. When employees leave their work station without locking their desktop it opens the network to those who were not granted access to the system. Someone with malicious intent could easily steal files or introduce a virus locally. This device will address the problem of users leaving work stations unsecure. Although locking a work station is simple enough, employees can be distracted and simply forget. This compromises the network and introduces threats that could be easily stopped if all employees locked their desktops when away from their terminal

Stakeholders- My device will benefit the entire organization that uses it, including the IT department and all users.

1. IT Department: The most direct stakeholder that benefits from my device is the IT department. Depending on how bad the issue of unsecured workstations is, it could relieve quite a bit of day to day issues interacting with employees who forget or refuse to do so. It could also decrease threat vectors that the department must worry about since unsecured workstations should no longer be a problem unless an employee tampers with the device.
2. All employees that use a workstation in the organization: Employees will no longer have to worry about unintentionally leaving their work station unlocked. Things happen all the time that can distract an employee, it is natural and will happen. My device alleviates the problem entirely.

**Evaluation of technology requirements**

1. Python: I will be utilizing Python to program my device. I plan on creating a simple loop that reads off an analog sensor in the cushion. Once weight is removed the Python program will start a 10 second timeout by default before locking the computer. The input will be read from the user’s computer like it is a keyboard since the only input needed will be the Windows Key + L command.
2. USB On the Go: This interface allows my Raspberry Pi to emulate a Human Interface Device (HID). By configuring the Raspberry Pi as a gadget, I can then send HID keystrokes to the connected PC, emulating a keyboard. This will allow me to lock a user’s desktop via a signal sent from the Raspberry Pi.
3. Raspberry Pi Zero W: The most important hardware of my project. This device will be running the Raspbian operating system and will host the Python program. I chose the Raspberry Pi over other kits like the Arduino because it natively supports USB On the Go. This will help with communicating to the user’s desktop and reduce potential security concerns. The Raspberry Pi will be connected to an analog sensor in the cushion, which will send a resistance input into the Raspberry Pi. When it receives this signal my Python program will start the timeout countdown once the resistance is below a certain threshold, indicating that the user stood up from the sensor. The Raspberry Pi will then send a command input via USB On the Go, connected to the user’s desktop, that locks the computer.
4. Jumper Wires (Male to Male/Male to Female): These jumper wires will connect my flex force sensor to the GIO header and analog to digital convertor located on the Raspberry Pi.
5. Breadboard: The breadboard will be used to wire the MCP 3008 ADC onto the Raspberry Pi GPIO header.



1. Flex Force Pressure Sensor: This is an analog sensor that will be inserted in the cushion of my device. It will detect the presence of a user via weight pressure, and when removed it will send a signal to the Raspberry Pi GPIO header which will send my Python program a simple ON/OFF signal to start the loop that locks the user’s desktop.
2. MCP 3008 8-Channel Analog to Digital converter: The ADC will be connected to the GIO header on the Raspberry Pi. It will also use a variable resister and read the input based on voltage (from 0V to 3.3V).

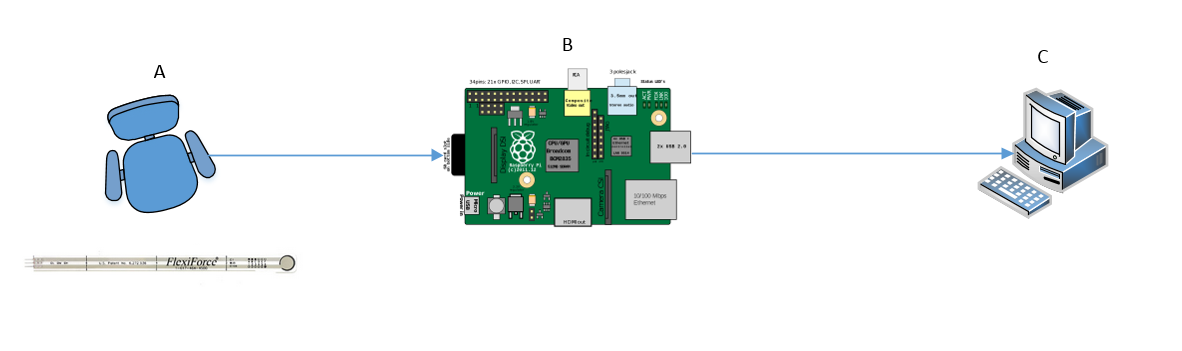
<https://www.adafruit.com/product/856>

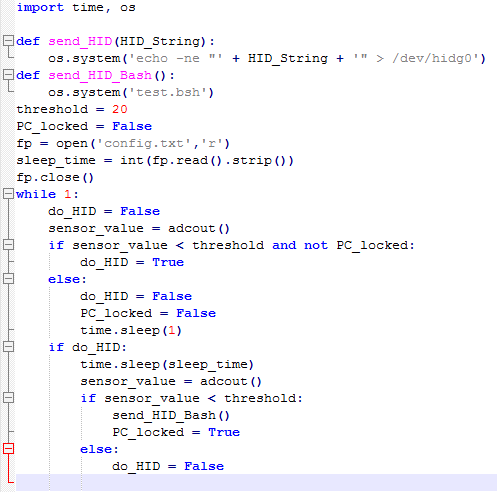
1. USB to TTL serial cable: This device allows me to easily display or work with my Raspberry Pi through a terminal on my laptop. The USB plugs into the laptop, and the serial cables plug into the GPIO pins on the Raspberry Pi. This allows me to open and control the Raspberry Pi terminal on a separately connected device.

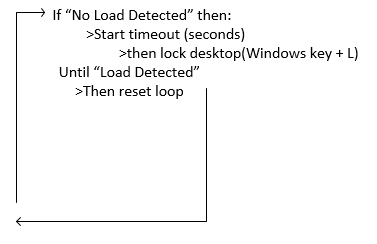


**Prototype design specification**

1. Architecture: Please refer to the diagram below. The process moves from left to right.



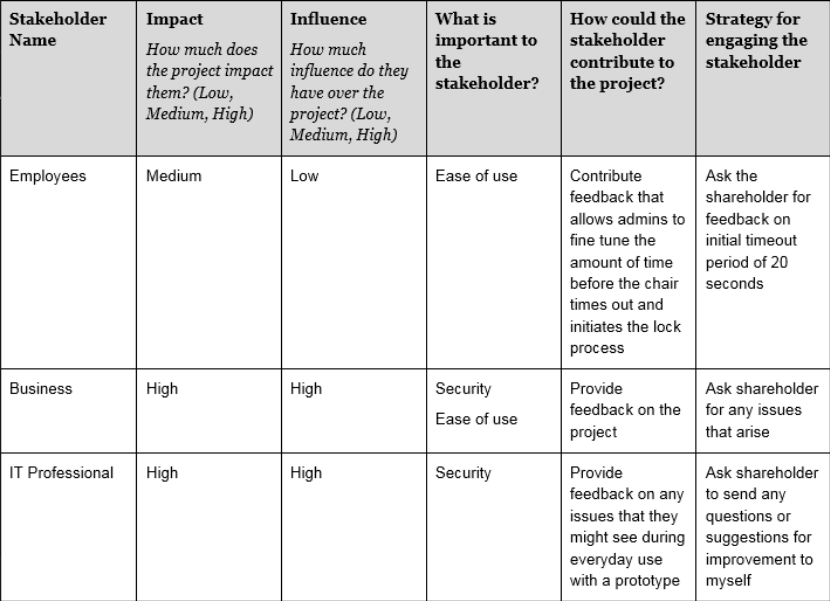
1. The secure seat will be contained inside a cushion which will can be placed on any office chair. I could not find a sensor large enough, so the prototype will use a much smaller sensor for the time being. The Raspberry Pi will be stuck to the bottom of the chair with either Velcro or adhesive. The cushion will contain an analog pressure sensor that is connected to the Raspberry Pi. The flex force analog sensor will give resistance reading as weight is removed from the chair, sending an input signal to the GPIO on the Raspberry Pi itself. I will have to connect the flex force sensor utilizing jumper wires (male to male and male to female) that are connected to a MCP 3008 Analog to Digital convertor. This allows the sensor to send the Python program input that it understands.
2. The Raspberry Pi is where the interface and application program will be running. The administrator will be able to modify the timeout of the chair that locks the keyboard by logging in with their credentials. Once the flex force sensor sends its input to the GPIO on the Raspberry Pi, the application will run a loop that starts the process of automatically locking the user’s desktop. This will be done by sending a signal to the user’s desktop via Bluetooth.
3. The user’s desktop will detect the Raspberry Pi input as a HID keyboard. It will detect a command that locks the computer (Windows Key + L), and then the loop in my Python program will reset and wait for the user to sit down and stand up again.
4. Software Design
5. The software for my project is simple. The Raspberry Pi will be running a Python program that receives input from the sensor in the cushion of a user’s chair. It will be using Ubuntu 16.04 as its operating system. The program itself will be a basic loop with IF and WHILE statements. The output will be a simple command for the desktop to lock after a certain amount of time (Windows Key + L). 



1. There is no interface associated with my device. I have a timeout.txt file in the program folder that allows the administrators to change the timeout value that initiates the lockout sequence on the Raspberry Pi.

**Prototype implementation and evaluation**

Stakeholder Analysis:



|  |  |  |
| --- | --- | --- |
| **Use Case Name:** | Secure Seat | |
| **Scope:** | Full | |
| **Primary Actor:** | Employee | |
| **Stakeholders and Interests:** | Employees, Business, IT Department, Management | |
| **Preconditions:** | Employee is about to leave the room without locking desktop. | |
| (Tobi, 2016) | User can stand up from their seat and have their desktop lock after a preset timeout. | |
| **Main Success Scenario (aka Flow of Events)** | | |
| **Actor Action** | | **System Response** |
| 1. Employee sits down on chair with secure seat. | | 2. Python program is running on Raspberry Pi, waiting for the sensor to send input that weight on the chair has been removed |
| 3. Employee stands up from chair and walks away, forgets to lock their computer. | | 4. Flex Sensor detects lack of weight, sends input to Raspberry Pi which runs the Python loop, starts 20 second timeout. |
| 5. Employee has been away from their desktop for more than 20 seconds. | | 6. Raspberry Pi sends input signal to desktop, Windows Key + L, and locks the employee’s desktop. |
| 7. Employee comes back to desk, notices their computer is locked, sits down on chair to unlock. | | 8. Flex Sensor senses weight, restarts loop. |
| Extensions (aka Alternative Flows) | |  |
| User sits back down before timeout finishes, loop resets and desktop stays unlocked | | |

1. User analysis
2. Describe Users and Requirements:
3. IT Staff: The IT staff’s top priorities in order are security, reliability and usability. Secure seat should improve security by ensuring that employees do not leave their workstation unsecured. However, the device has potential to be a threat vector if implemented poorly. Since my prototype is hardwired to the computer there’s much less of a security threat. I previously attempted to make Bluetooth work for this step of the project and it opened vectors of attack. Someone can spoof the device and stage a man in the middle attack. Although this is not a very feasible attack and most likely will not happen unless the person is in the facility. By that point employees would hopefully notice a stranger in their office trying to gain access to their devices. Reliability ties directly into security. If the device stops working, or only works occasionally, the employees will not take it seriously and view the device as a burden. This would mean more unsecured workstations, or complaints about lack of usability. This is the last requirement, although it is just as important as the others. There must always be a balance between security and usability or employees will ignore the operating procedures put in place by the IT staff.
4. Employees (Users): The employee’s primary requirement will be usability. If the device makes it harder for them to use their desktop or produce work, they will not like the device and make it known to their superiors. The device aims to increase overall usability by locking the desktop without the employees needing to worry about it. However, they should always try to lock their desktop on their own, Secure Seat is a failsafe.
5. Management: The management requirements are simple. If the IT staff ensures that the device is secure and reliable while maintaining usability for the employees to stay happy and productive, the management requirements should be satisfied.
6. User Analysis
7. **What is the users average level of expertise?**

The users will be employees in an office and the IT staff. The IT staff should have a high level of expertise while the employee’s expertise level could range from high to low depending on their job.

1. **What is the age range of the users?**

The users age covers a wide range, anywhere from early 20’s to retirement age.

1. **Is the device an integral part of the user’s everyday productivity?**

Not necessarily. It cannot increase productivity, but it will be used every day to ensure network security.

1. **What are the consequences if a user makes a mistake using the device?**

Catastrophic depending on who gains access to the network. The users of the device, office employees, should not have to worry about proper usage of the device since it is going to be working automatically on their seat/desktop. If the IT staff do not properly configure or install the device, it could create issues for employees. For example, If the timeout is not configured properly it could be annoying to be locked out of your computer when standing up to grab a book from the shelf. Outside of inconveniencing employees, a device that is not configured properly could open the possibility to network intrusion if the employee relies on the device to lock their computer and does not notice it is not being locked.

1. **Do users want to know about the technology that sits behind the interface?**

I assume that the IT staff wants to know. They need to be aware of everything happening on their network, and how to configure the device in the best way for their users.

**Description of Costs and Benefits**

Detail of existing system (ProxMat):

ProxMat is a 24x24 pressure sensitive floor mat that accomplishes exactly what I have intended to do with the Secure Seat. It is very similar to my prototype, but much more advanced and expensive. One ProxMat costs $375.

While Secure Seat functions based off a user sitting on a chair, the ProxMat functions by the users placing their feet on the 24x24 mat for the system to detect their presence. This can be utilized at workstations or kiosks. When the user walks away, the workstation is locked. While my prototype utilizes USB on the go emulation to send HID keystrokes to the desktop, ProxMat claims to use its proprietary KIT (Keyboard Interface Technology) to send keystrokes to the desktop.

Configuration for the ProxMat appears to be highly customizable. The device includes a configuration utility that allows keyboard sequences to be programmed, timeouts to be adjusted, and other modifiable options that are stored on board the device.

This vendor also offers 3 other products that aren’t as similar as my device, but accomplish the same goal. TF2000 is an ultrasonic presence sensor that locks the user’s PC when they walk away utilizing ultrasound waves. It’s mounted on top of a user’s monitor. Another product offered is the SonarLocID, an auto locking keyboard. This device has a smaller sonar detector mounted on the keyboard that initiates the lock sequence. It also offers the same configuration utility that the ProxMat provides. The DocLok is the last product offered by this company. It is mounted on the doorway of an office and is intended for Doctors office use. When the doctor leaves his office the DocLok sensor breaks magnetic contact and sends a signal to the workstation to be locked.

Detail of proposed system:

My proposed system utilizes a custom-made pressure sensor that sends a signal to a Python script running on a Raspberry Pi, which then sends the screen lock input to a user’s desktop utilizing a USB on the go connection. The user’s desktop thinks it’s connected to a USB keyboard and accepts the HID command for Windows Key + L (“\x8\0\xf\0\0\0\0\0”).

While the devices I mentioned in the previous part of this section have much more refined and developed devices, my prototype is more than four times less expensive. While a ProxMat costs $375, my prototype should cost around $70. This includes the pressure sensor, wiring, Raspberry Pi, and extra cables. The total cost of components to develop my project was roughly $250. The development time, if I was being paid a $25 an hour rate came out to $2250. This estimate was found by charging 6 hours per week for the 15 weeks during the semester.

One of the negatives of my prototype, especially compared to the ProxMat is that it’s not as easy to configure or setup. My prototype has very tiny wires that need to be expanded in a newer version. With the current iteration wires are daisy chained all over the place which will almost certainly lead to device failure when a user trips over the cords and breaks the device.

While my prototype isn’t as refined as the other items being offered, it is much cheaper and much more open to improvement. The code is easily viewable with the ability to be modified. The circuit can be revised completely, and a larger pressure sensor can be added. The only real negative that can’t be resolved is how the Raspberry Pi interfaces with the Desktop. Currently the only way for this to be achieved is utilizing USB on the go. I could abandon the cushion idea entirely and fashion a mat like the ProxMat, or I need to make the pressure sensor circuit smaller and self-contained, so the prototype has at the most one long wire connecting to the user’s desktop from the cushion.

**Lessons Learned**

I have learned a great deal, most of it in unexpected areas, while working on this project. I had no idea how to get started with my project, and researched what I could based off my previous knowledge and assumptions of how the project would work and be assembled. Because of this I wasted quite a bit of time learning and experimenting with different approaches and utilization of technologies for my project.

Assembling the pressure sensor and circuit board was a great example of this Starting out I thought I would be able to purchase a USB pressure sensor, but as I got deeper into the project I realized these simply do not exist or are way too expensive. This meant I would have to wire up my own. I had to learn the basics of the Raspberry Pi and how it receives input from different devices, in my case analog signals that a pressure sensor uses. This meant I had to learn about how analog to digital conversion works, specifically with the MCP3008 chip I’ve used in the project.

The Mobile Makerspace was a great place to work on the project and gain knowledge, and I regret not finding it sooner. Here I received knowledge and feedback on the processes I decided to use to create my prototype. There were tools I was able to use that I had never experience before. For example, my Raspberry Pi Zero didn’t come with the GPIO headers installed. I used a soldering iron for the first time at the makerspace and successfully installed the GPIO header myself.

Once I broke my project into three separate components it became easier to manage. My sensor, Python code and USB on the go emulation, once tied together, should result in a complete and working prototype.

**Broader Impact**

There were a few courses I took during my time at USA that helped with my project. All three of the programming courses I took helped with my programming and scripting portions of the project. CIS 115, ITE 285 and ITE 370, all Visual Basic courses, helped with the understanding of Python. Even though it wasn’t the same language it followed the same logic and I was able to use the same pseudo code strategies to plan my project.

ITE 272 helped me understand how to use Linux, and helped speed up the process since I wasn’t constantly looking up commands for the terminal. ITE 475, project management, helped familiarize me with the terms and processes used to manage a project and its timeline. These prepared me and helped create the project timeline for this course.

My project was a little bit unorthodox for IT students since it also required a little bit of electric engineering knowledge. After meeting with Dr. Russ and reading materials he suggested, I was able to sketch and complete a circuit that allows me to connect my pressure sensor to my Raspberry Pi and have it receive inputs. This was a major portion of my project and I had never had a course explaining or demonstrating these concepts. It put me outside of my comfort zone.

**Conclusion**

This project has helped to expand and improve skillsets I previously did not have. Before I started working on this project I had a very different idea of how to make my prototype work. Every time I pursued a method that didn’t work I would have to reevaluate my approach and figure out another way. I never thought I would have to learn basic circuitry and dive deeper into physical computing.

When I started work on the project I thought I would be able to purchase most of my components and tie them together with code running on the Raspberry Pi. I was surprised to find out there weren’t any USB enabled sensors for sale, and would have to create my own. This project tied together knowledge and techniques from multiple areas of technology. This included programming, scripting, basic circuitry, emulation and basic Linux computing.

Although the tools and devices that I utilized were cumbersome and fragile at times, I enjoyed working with them. I believe the Raspberry Pi Zero might be a little bit too delicate, but I can still appreciate the device. It allows users a very open playing field when it comes to whatever project they want to create.

# References

*24 x 24" ProxMat*. (n.d.). Retrieved from esecurity products: http://www.esecurityproducts.com/ProxMat\_p/rdr-pm2424-00.htm

*ADC (MCP3008) Raspberry Pi Tutorial*. (2017, August 26). Retrieved from youtube.com: https://www.youtube.com/watch?v=jDJAklvIuPY

*Beginning With Python*. (n.d.). Retrieved from anh.cs.luc.edu: http://anh.cs.luc.edu/python/hands-on/3.1/handsonHtml/ch1.html

Caird, A. (2015, January 07). *Using a Force Sensitive Resistor with a Raspberry Pi* . Retrieved from acaird.github.io: https://acaird.github.io/computers/2015/01/07/raspberry-pi-fsr

*Connecting to your Raspberry Pi via the USB to TTL Serial Cable*. (2014, August 31). Retrieved from workshop.raspberrypiaustralia.com: http://workshop.raspberrypiaustralia.com/usb/ttl/connecting/2014/08/31/01-connecting-to-raspberry-pi-via-usb/

*docs.microsoft.com*. (n.d.). Retrieved from Raspberry Pi Pinout: https://docs.microsoft.com/en-us/windows/iot-core/media/pinmappingsrpi/rp2\_pinout.png

*How to get and install NOOBS*. (n.d.). Retrieved from raspberrypi.org: https://www.raspberrypi.org/help/noobs-setup/2/

Monk, S. (n.d.). *Adafruits Raspberry Pi Lesson 5. Using a Console Cable*. Retrieved from learn.adafruit.com: https://learn.adafruit.com/adafruits-raspberry-pi-lesson-5-using-a-console-cable/software-installation-windows

*Physical Computing with Python*. (n.d.). Retrieved from projects.raspberrypi.org: https://projects.raspberrypi.org/en/projects/physical-computing

*Programming a Raspberry Pi with Python*. (2017, October 1). Retrieved from raspberrypi.org: https://www.raspberrypi.org/magpi/learn-programming-Python/

*The USB HID protocol*. (n.d.). Retrieved from docs.mbed.com: https://docs.mbed.com/docs/ble-hid/en/latest/api/md\_doc\_HID.html

Tobi. (2016, February 26). *Composite USB Gadgets on the Raspberry Pi Zero*. Retrieved from isticktoit.net: http://isticktoit.net/?p=1383#more-1383

*usb.org*. (2004, October 28). Retrieved from HID Usage Tables: http://www.usb.org/developers/hidpage/Hut1\_12v2.pdf