# Overview

Our project is a prototype for a high-security RFID door lock system that would be used in an area where multi factor authentication (e.g. government building). The program involves 2 Arduinos, a master and a slave. The master runs all the hardware components, including an RFID scanner, 2 buttons, a piezo speaker, a screen and a servo (to represent the door lock mechanism). The slave Arduino controls the security and has no hardware components. They’re both connected through Bluetooth serial, using HC-05 modules.

# Software

Face Detection: For the prototype, we’ve decided to go with facial detection, as it is significantly easier to implement, given the time constraints. In order to do this, we’ve created a python 3 program using a library called Open CV. This has inbuilt face detection algorithms. Of which we’ve decided to use the HAAR algorithm (Cascade).

PC-Arduino Comms: In order for the face detection to be linked with the Arduino, the python program must send a serial ***command*** to the master Arduino. This is done using a library called PySerial, and allows for the main program to be run only when a face has been detected for long enough

Master Program: The master program controls the majority of the components used in the system, including the RFID scanner, the TFT screen, the buttons, and the “door lock”, which is a servo in the prototype.

Master-Slave Comms: The master must communicate with the slave every time an rfid tag is scanned, or a password is entered. This is then checked with the correct/expected output(s), and another signal is sent back to the master.

Slave Program: The slave is mostly checking the expected outputs against the actual outputs. However, it’s also in charge of controlling the bubble machine whenever the timer runs out, or they get 3 passwords or RFID tags wrong

# Hardware

PC: The computer simply needs python 3.X with the previously mentioned libraries installed, as well as a webcam, and a spare USB port to plug the master in to. In a final version of this program, the PC-Arduino coms might be done through Bluetooth/another wireless communication protocol

Master: The master contains: An ***[Name]*** RFID receiver; a TFT screen; 2 buttons; a piezo speaker; a 9g servo and a HC-05 Bluetooth module in master mode. In a final version of this system, a pin pad would replace the buttons, so the pin can be harder to crack (> 256 options)

Slave: The slave contains: A ***[Name]*** relay connected to a bubble machine and a HC-05 Bluetooth module in slave mode. In a final version of this system, the bubble machine would be replaced with something higher security, such as a taser

Communications: We decided to go for a Bluetooth serial connection between the two Arduinos, because in the final version of this system, one of the modules would be in a different area. This would probably use WiFi/ethernet, however using this for our prototype would have been a challenge due to issues regarding the network used on campus. Using Bluetooth also made development of the project easier, as the Arduinos didn’t need to be right next to each other while testing and developing features

# Difficulties

PC-Arduino Coms: The main issue we had here was involving the fact that we couldn’t see the serial display for the master Arduino while the python program was communicating with it.

Master-Slave Coms: The main issue we had when it came to getting the Arduinos to communicate was setting up and connecting the two HC-05 modules together. Once they were together, the process of using them was the same as typical serial if they were wired together

Connecting the TFT Screen and RFID Reader Simultaniously: When we were trying to connect the two devices to the master at the slave time, we noticed that some of the pins on each device can’t be reassigned using code (MOSI, MISO and **[Insert]**) pins were preassigned and need to be used in order for the device to function. After some ***research***, we found out that the 6 female ***[NAME]*** pins on the Arduino board can be used as secondary pins for such features. Therefore all we needed to do was plug them into there.

Fritzing Schematic: While making the schematic, we ended up not having all of the components we used in our system available for use. On top of this, the 6 female ***[NAME]*** pins that we had to use weren’t available on the Arduino schematic on the program. This meant building the schematic was more challenging, as we had to “build” our own schematics for each of the new components

Connecting the Bubble Machine: Due to the bubble machine being an off the shelf product, it wasn’t inherently designed to work with the Arduino. This led to power issues whenever the device was plugged in for controlling, due to the machine containing 4 AA batteries (for 6v power), and a button, which was removed for the process of wiring it up. The solution to this was using a relay, and connecting the machine along the closed portion of the switch, with the coil being controlled by a digital out pin. This meant it could then be programmed simply by using digitalWrite(X,HIGH/LOW), just like an LED.

Inherent Issues: Due to time constraints, as well as the limitation of people/knowledge, we wouldn’t have been able to complete the full project. However, we made a working prototype that represents the goal of the system

# Conclusion

# Appendix

# Contributions

Hardware: Jake Misfud – 50%; Tom Lancaster – 50%.

Software: Jake Misfud – 50%; Tom Lancaster – 50%.

Research: Jake Misfud – 50%; Tom Lancaster – 50%.

Report: Jake Misfud – 50%; Tom Lancaster – 50%.

# Images

***Schematic Here***

# References

<https://www.youtube.com/watch?v=Q8QlNuTUe4M>

<https://realpython.com/face-detection-in-python-using-a-webcam/>

<https://docs.opencv.org/3.4.3/d7/d8b/tutorial_py_face_detection.html>

<https://www.arduino.cc/en/Tutorial/AnalogInputPins>

[https://www.arduino.cc/en/Guide/TFT#toc9#](https://www.arduino.cc/en/Guide/TFT#toc9)

<https://www.arduino.cc/en/Reference/SPI>

<https://playground.arduino.cc/Main/PinChangeInterrupt>

<https://www.forward.com.au/pfod/ArduinoProgramming/TimingDelaysInArduino.html>