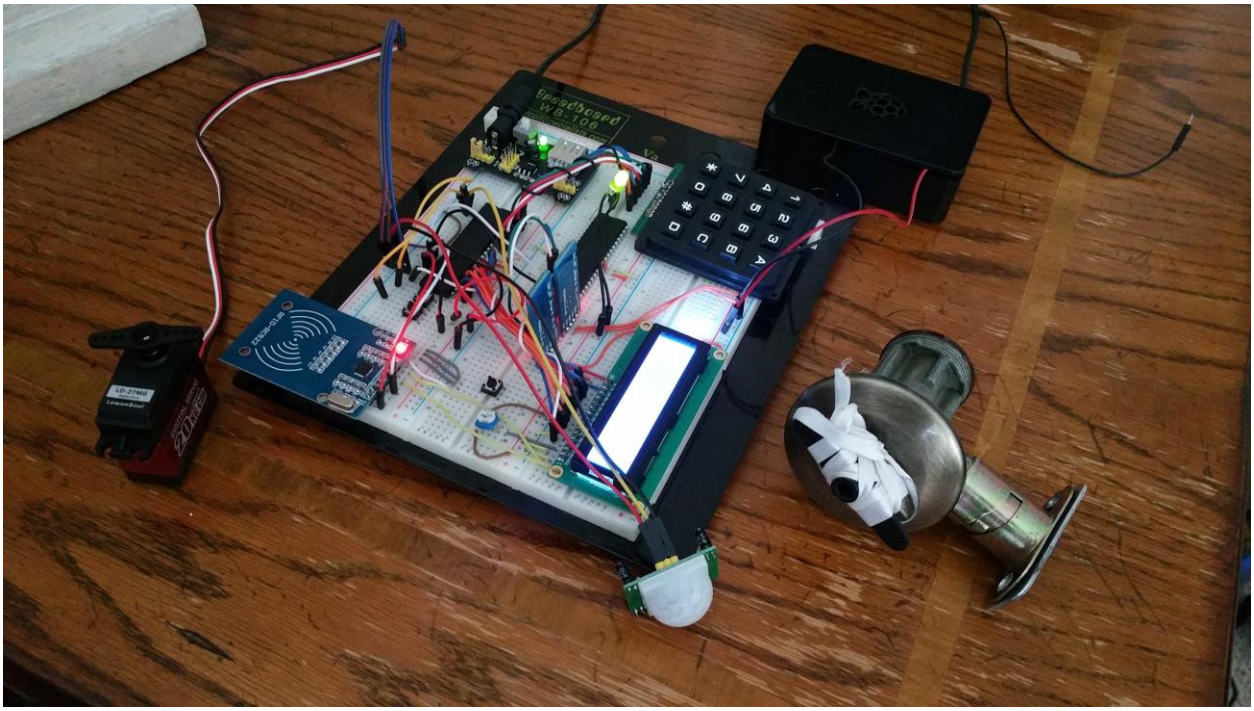
EZ-Lock

CS122A: Fall 2017

Alex Ardelean

Table of Contents



Introduction

I created a security system that could be used to lock and unlock a house door. You can setup the lock for practically any door in minutes. There is also a motion detector that detects if there is motion outside of the house. There is a button on the inside of the house that can lock the door and from the outside you can unlock it from the keypad or theoretically from NFC.

Insert a picture of your project (Your completed project).

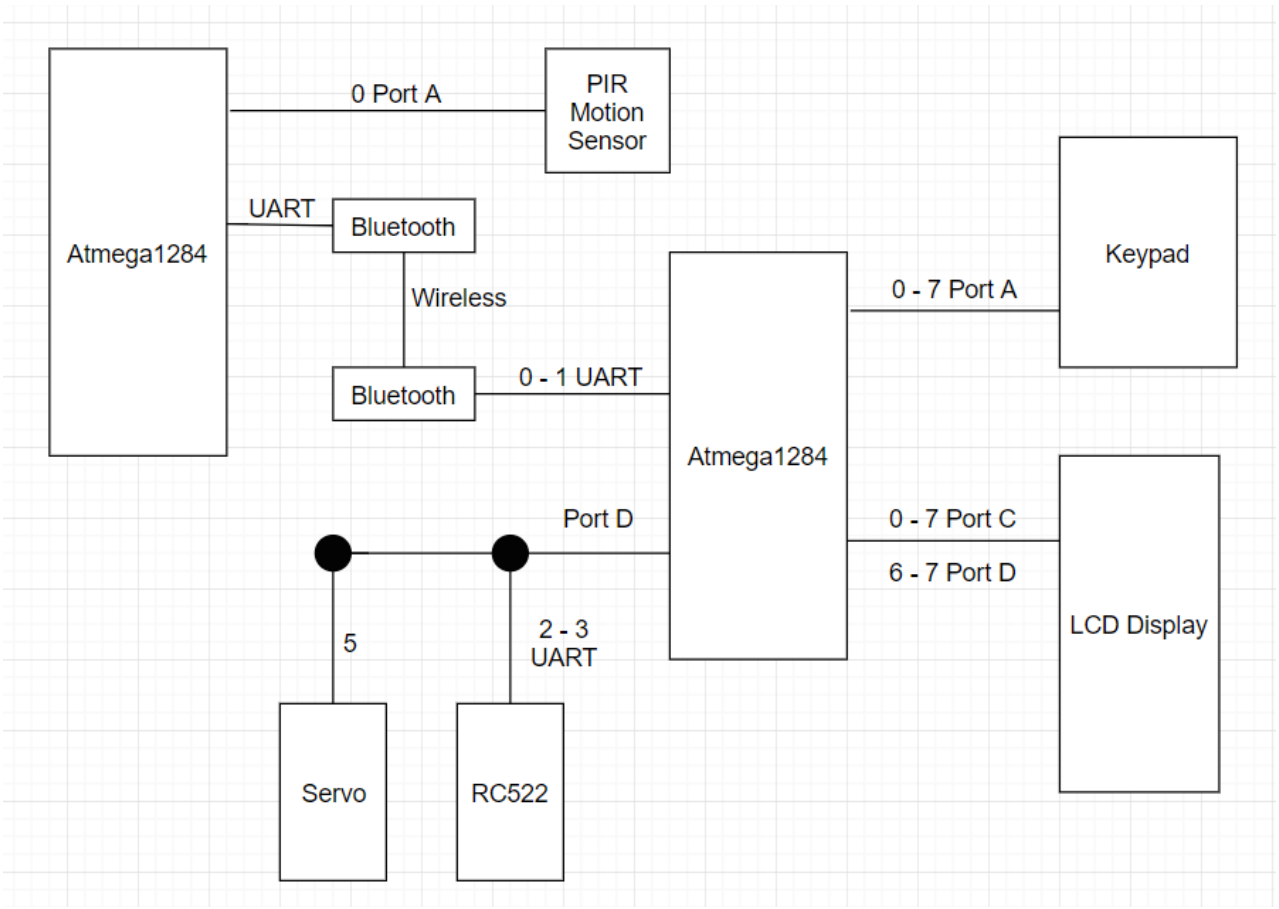
Hardware

Parts List

The hardware that was **used** in this project is listed below. The equipment that was not taught in this course has been bolded. *Include part numbers when available*.

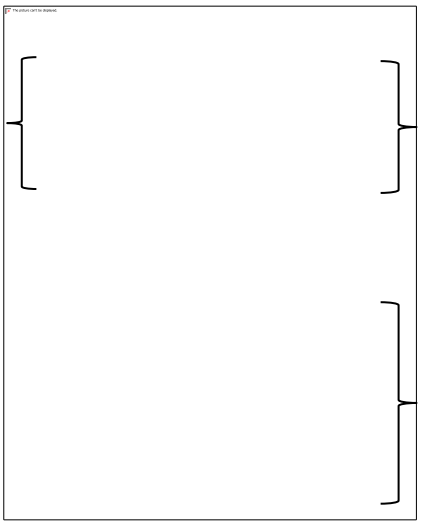
Part	Part #	Quantity	Price (optional)
ATMega1284	ATMega1284	2	$8
RFID-RC522	**RC522**	1	$8

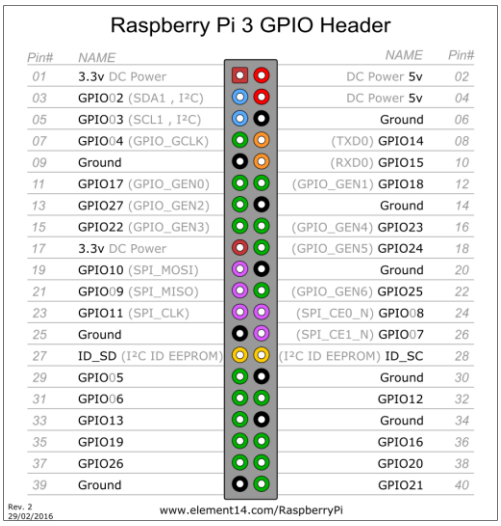
Wireless Bluetooth	**DSD TECH HC-06**	2	$18
Raspberry Pi 3	**Model B**	1	$35
Pir Motion IR Sensor	**HC-SR501**	1	$6
Digital Servo with 20kg High Torque	**LewanSoul LD-27MG**	1	$18
		Total	$93



Block Diagram

Pinout (For each microcontroller/processor)





Software

I have a state machine that allows you to unlock the door by typing the correct password. It also allows you to change the password by first typing #{oldpin}#{newpin} with a pin being a combination of 4 numbers or letters. It also allows you to change the degrees of rotation and rotation direction.

I have another state machine that polls the motion detector and sets a variable if there was motion detected. A different state machine takes that variable that was set and sends it through UART to my bluetooth controller. I have a state machine on a secondary atmega that allows waits to receive something from bluetooth and sets an LED light on or off depending on what it receives.

I have a state machine that allows me to click a button to lock my door.

Implementation Reflection

I think that the project I completed is a nice foundation that would be easy to expand upon to ultimately complete the entire vision. Currently I can use most the code directly, or with minor changes if I was to finish the proof of concept that I started.

Although I do enjoy engineering features on top-of working parts, I do not enjoy the troubles that it takes to get some parts to simply work. Once I figure out how to get a part to work as expected, then I can start designing the code of the feature exactly how I want it. The labs are nice because the lab maker has already figured out how the parts work with the atmega and we built from that. This seems like an issue that would completely go away with more experience, but it's something that still bothers me.

If I was to do the project again I'd seek out people that already did similar things and ask for their advice on how to get things to work together. I tried finding guides or tutorials online for

some of my issues, but I wasn't able to find help that was useful. The most useful "help" was to read the datasheets of the items that I got.

My favorite thing about my project is that I got the lock to work exactly how I wanted it to with the LCD screen and keypad.

Milestone

My milestone was to "have the controllers and raspberry pi talking to each other. The door lock with NFC and keypad should work [and to] begin working on other sensors."

I fell short of my milestone. The first thing I started working on was trying to get my raspberry pi to communicate with the atmega chip through SPI. The reason I began with this is that this was a fundamental step into moving on with my 80-90 or 90-100 project. The reason for choosing SPI over something like UART would be so I'd be able to program the atmega from the raspberry pi so that in the future hypothetical customers could update the software.

Of course he problem was that I couldn't figure out how to get the raspberry pi to communicate with the atmega chip. I first figured out how the GPIO pins of the raspberry pi worked using C language. After that I started looking for SPI tutorials for the raspberry pi. I made sure that the serial connection in the settings were turned off so that the bus was open for SPI connection through the GPIO pins. After this I found some apparently working SPI code and I tracked down the registers that it was setting in the raspberry pi's datasheet. I compared this to the registers I was setting in the atmega, which i tested previously as working. After trying to figure it out for easily 10s of hours I decided that I would first get the basics working before shooting for a fully featured project.

Completed components

I got most of my 70-80 project finished and working. I was able to unlock and lock a door using a keypad and a servo motor. To set up the servo for a specific door I give the option to change clock direction of the unlocking motion to either clockwise or counterclockwise and I allow you to change the amount of degrees that the servo turns to unlock the door. You can also change the password from the keypad.

Instead of figuring out how to store data I asked McDaniel if instead I can connect two bluetooth modules together for wireless communication. This makes more sense for my project because for the final vision I'd need wireless communication to the keypad.

I got the motion sensor to detect as intended.

I fell short in getting the door to unlock when putting a known NFC tag on the NFC reader. This was the other major problem I had in my project. The NFC chip should work through either SPI, UART or I2C. I tried to get it to work with SPI and UART, but with no avail for either one.

The NFC chip uses specific instructions for reading or writing to registers based on the different types of communication protocols you use. I understood these protocols and tried to write to addresses and then immediately read them. This would give unexpected results.

I also tried giving commands to the NFC chip to see if it would accept anything but that didn't work either. The baud rates were the same for the atmega and the NFC chip. I tried to make sure that all the UART and SPI register settings were compatible with the UART chip. The NFC chip automatically detects the communication protocol on startup. The data sheet for the NFC chip is a little over 100 pages long, I essentially read the entire thing. I must be doing something wrong, but what exactly I wasn't able to figure out.

Incomplete components

From my final vision I didn't get to the part where I connect a camera to the raspberry pi that is ultimately connected to the motion sensor to determine when the camera turns on. After completing that I'd start using this software called Blynk that would allow me to control the entire system through my phone. I'd set up alerts to when the front door unlocked or when the motion sensor was detected and then I'd be able to watch a stream of the camera through my phone.

The difficulty was purely that I didn't figure out how to connect the raspberry pi with SPI to the atmega microcontroller.

Youtube Links

- Short video: https://www.youtube.com/watch?v=zu1cq5sy2eU&feature=youtu.be
- Longer video: https://youtu.be/FGIL9cvSYHU

Testing

- Servo Lock
    - Be able to click lock button rapidly in succession
    - Unlock door in succession
    - Different angles at different clock or counterclockwise rotations for accuracy
        - Out of bound numbers
    - Make sure there is enough torque to open a door lock
    - Password is able to change and works
        - Out of bound numbers
- Keypad
    - All buttons input correctly
- LCD Display
    - When out of bounds keys are pressed it doesn't give unexpected results
- Bluetooth
    - Test with phone that I can send and receive large strings of data without error
    - Connection between bluetooth modules can send and receive large strings
- Motion Detector
    - Motion detector detects motion at approximately 5 feet.
    - Motion detector doesn't set off false positives
        - Put it in a box and set a flag if it gives a positive, test for about 30 minutes

To test that it generally works I gave it to my brother with verbal instructions on how to set everything up and make it work.

Known Bugs

The biggest known bug is that my servo sometimes fails, it's very rare but happens. I've tried delaying before or after the servo turns with minimal success. I've tried using different thicker wires because it is recommended in the datasheet to use good wires. I believe it is a power issue where my servo is not getting the power it needs 100 percent of the time.

My next step would be to either get an op amp or a different power source that can supply the recommended 6 volts of power. I've "fixed" this temporarily because now instead of completely breaking when not turning the full way you can do another servo command and it will reset itself. Once again it is very rare, but this prevents a full reset temporarily.

Resume/Curriculum Vitae (CV) Blurb

My project EZ-Lock taught me how to create something with minimal tutorials. Understanding how something works is enough to begin building and modifying it, regardless of lack of specific tutorials. Specifically I now have a deep understanding of PWM and how communication protocols SPI and UART work.

Future work

I would of course add a raspberry pi with a camera as the next feature. I have a 3d print of a case that could work but it doesn't necessarily look completely professional so I'd modify the design and make a PCB board that would allow me to fit everything in the case. I'd also need to get another bluetooth module and create a case and PCB for the LCD Display and keypad. With this I'd need to use better network security to prevent direct attacks on the lock portion. I may be applying to the company I used for the company research report but I'm more interested in other areas of computer science.

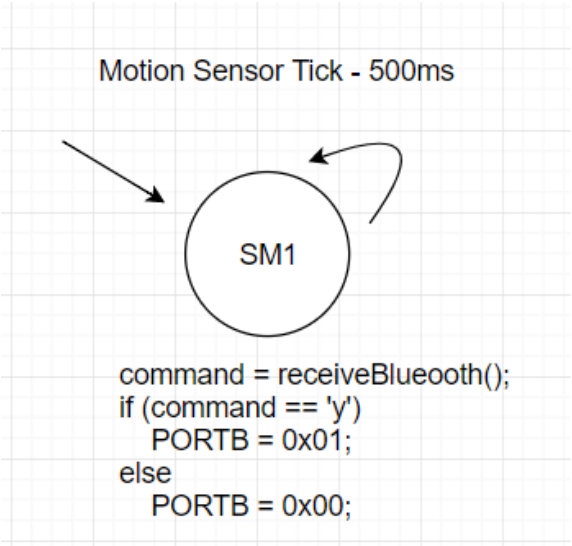
Extra Credit

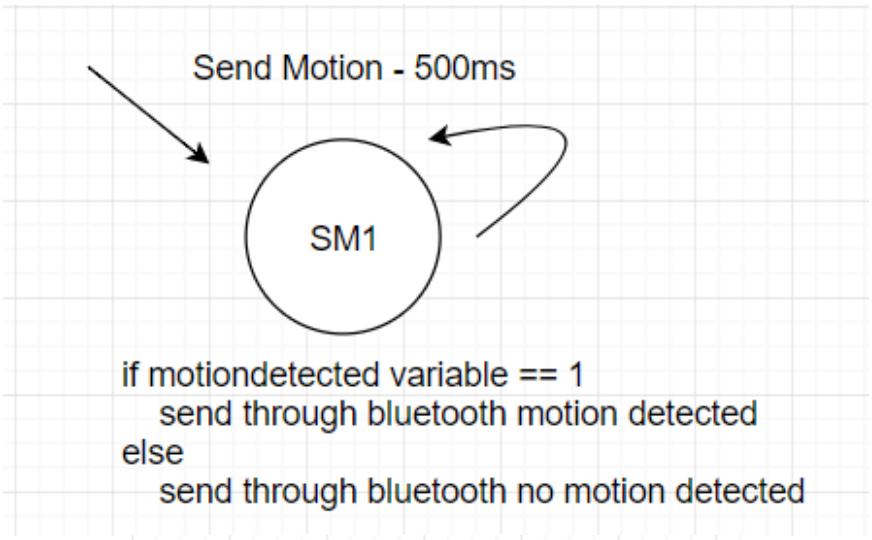
Include a link to a public DIY for your project: N/A

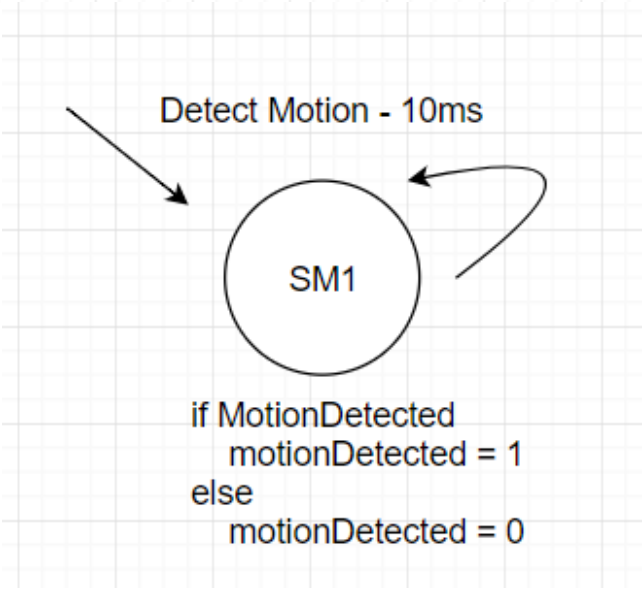
References

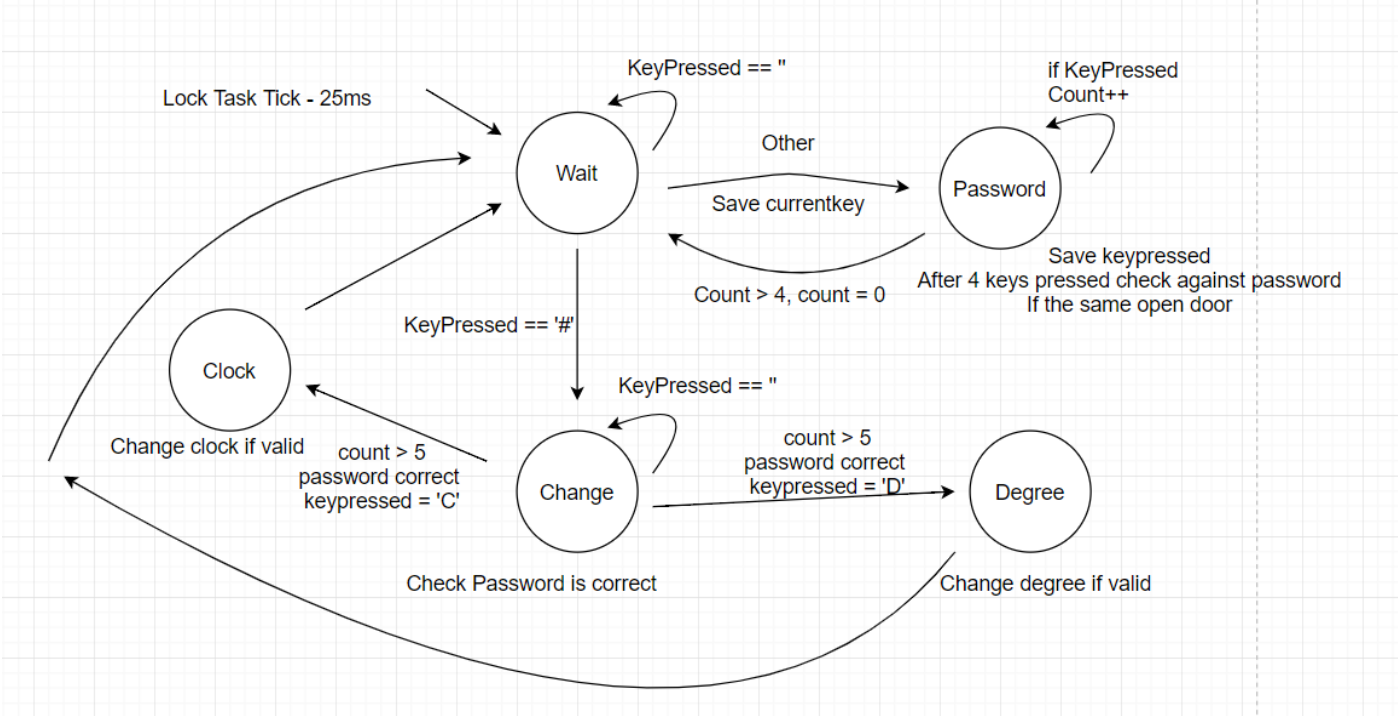
https://www.youtube.com/watch?v=bAcK80fm1_0&t= (Also contains relevant 3d printer design)

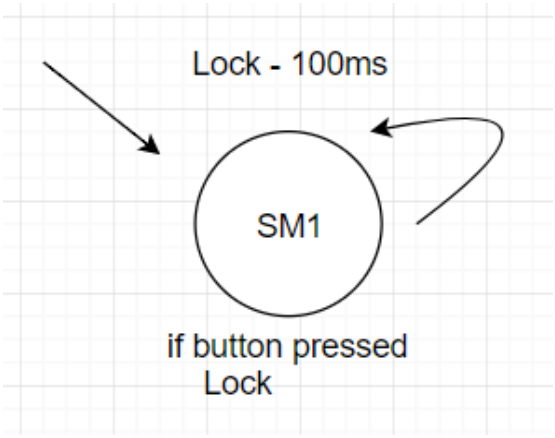
Appendix


Motion Sensor Tick - 500ms
SM1
command = receiveBlueooth();
if (command == 'y')
PORTB = 0x01;
else
PORTB = 0x00;


Send Motion - 500ms
SM1
if motiondetected variable == 1
send through bluetooth motion detected
else
send through bluetooth no motion detected


Detect Motion - 10ms
SM1
if MotionDetected
motionDetected = 1
else
motionDetected = 0


Lock Task Tick - 25ms
KeyPressed == ''
Wait
Other
Save currentkey
Password
if KeyPressed
Count++
Save keypressed
After 4 keys pressed check against password
If the same open door
Count > 4, count = 0
KeyPressed == '#'
Clock
Change clock if valid
KeyPressed == ''
count > 5
password correct
keypressed = 'C'
Change
count > 5
password correct
keypressed = 'D'
Degree
Check Password is correct
Change degree if valid


Lock - 100ms
SM1
if button pressed
Lock