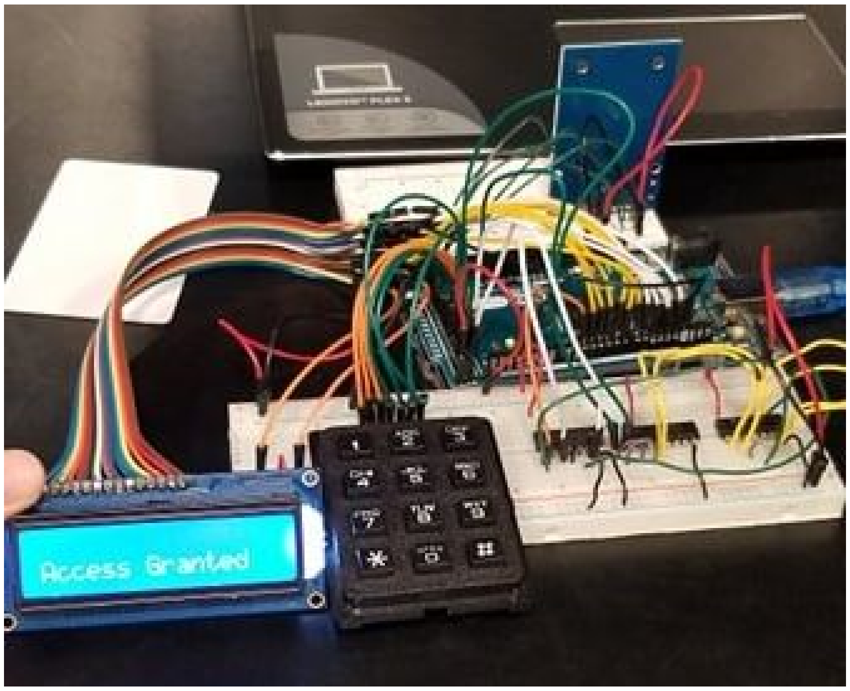
Dual-Method Security System



Sahil Patel (sahilsp2)

Anna Miller (annam4)

Mustafa Jamal (mustafa6)

May 5, 2019

ECE 120 Honors Lab

Thursday 7-8 PM

Introduction

Security is a prevalent issue on college campuses. Consider dormitories, for example. Often, students require I-cards and keyfobs to gain access and are constantly warned against allowing “tag-alongs”. Security precautions, however, when entering certain buildings and boarding the bus can make the process inefficient. Time is wasted in physically presenting I-cards when boarding the bus. One way to simplify security and ensure that only those with the proper credentials can obtain access would be to create lock system that implements an RFID scanner. The RFID scanner would only allow certain frequencies to pass through a gate. Additionally, in the event that the person forgot his or her fob, the gate would also have a password-based backup system. These systems could be installed at the entrance of buses and before the doors of buildings. They would function much like the anti-theft systems in retail stores by alerting staff members and security when someone tries to gain access without the proper permission.

Design Concept

The proposed design concept consists of a keypad circuit, an RFID circuit, an Arduino Mega and EEPROM, logic gates, and an indicator output. Figure 1 provides an image of the design.

The first part of the design is the keypad circuit, which consists of a 4x3 matrix keypad, a 16x2 LCD, and the Arduino Mega. The purpose of this circuit is to enable the user to input passwords. The keypad serves as the input and feeds data to the Arduino. The Arduino uses code to interpret the values being entered. The LCD portion serves to facilitate the use and troubleshooting of the project. The Arduino outputs to the LCD and display both the numbers on the keypad and whether or not the entry attempt was successful.

Alternatively, there is the RFID circuit. This circuit simply consists of a scanner that communicates with the Arduino Mega. Along with the scanner are various tags. The tags are detected by the scanner and compared with existing information to determine if the user should be allowed access.

The Arduino Mega is used to relay information between different portions of the circuit. It contains the libraries and code necessary to read the input from the keypad and the RFID scanner, output messages to the LCD, and receive information from the logic gates that are used to compare data in the circuit. Additionally, the Arduino includes an EEPROM, a component that is used to store the acceptable passwords and IDs.

Possibly the most important part of the design is the logic element. This portion of the circuit is used to compare the actual input against the acceptable input. This part of the design consists of a CMOS D-type flip-flop, an XOR chip, and a NAND chip. The input information is compared to the stored information to determine if the user should be allowed entry.

Finally, the circuit ends with an indicator. This output is simply tells the user whether or not the attempt was successful. Initially, the LCD reads “Enter Password.” If the proper password or RFID tag is used, the LCD displays “Access Granted. If unsuccessful, the display reads “Access Denied.” Additionally, when the stored password is changed the display indicates this with the message “Password Changed.”

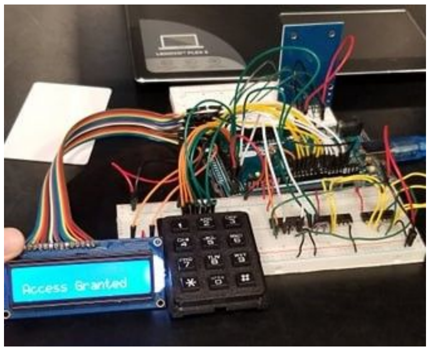


Figure 1: *LCD displays "Access Granted" after the correct password is entered.*

Analysis of Components

Keypad and LCD Circuit

The matrix keypad works by setting each key to a specific row and column. For example, if "3" is pressed, column 2 and row 0, which are pins 3 and 7 respectively, will read as HIGH. By registering the combination of the row and column, the Arduino will determine the number or character that has been pressed.

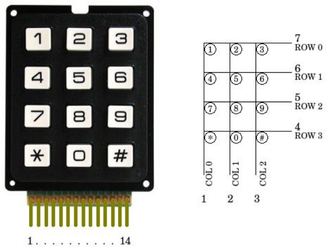


Figure 2: *The matrix keypad designates each of the 7 pins as a specific row or column.*

RFID Scanner

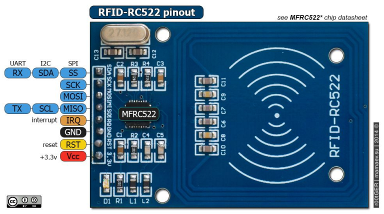


Figure 3: *The MFRC522 pinout.*

The MFRC522 RFID reader uses Serial Peripheral Interface (SPI) to communicate with the arduino. It has 4 main pins, shown in blue above. The SS (Slave Select) pin selects which component is being communicated to. SPI supports one master and any amount slaves. In this project, there was only one slave: the MFRC522. The SCK pin is the clock on which the two communicate. MOSI (Master Out Slave In) is the information channel for the arduino to the

MFRC522, and the MISO (Master In Slave Out) is goes the opposite way. The SPI communication is handled by an arduino library, detailed in the *Arduino Code* section below.

EEPROM

The EEPROM is used to store the correct passwords/card IDs and the currently inputted passwords/card IDs. In the Arduino EEPROM library, each address refers to a byte of memory, which is enough to store one ASCII character. For compatibility with non-numerical keypads and the LCD, passwords are stored as characters in sequential addresses in the EEPROM. The card IDs are also stored as sequential bytes, but not as ASCII characters, as the ID is not printed to the LCD screen. Most RFID cards have a 4-byte ID, and our code is configured to process 4-character passwords as well, although both these numbers can easily be changed.

Arduino Code

The code performs three core tasks: 1) interpret keypad input, 2) control LCD output, and 3) interpret RFID input. To complete the first task, the implementation of the Arduino Keypad library was required. In the loop function, character is being pressed (if any) is checked and stored to be used in task 2. The # and * keys have special functions, acting as the enter and change password buttons respectively.

LCD output is implemented using the Arduino LiquidCrystal library. The first row of the LCD shows the previously entered characters, and the second row shows the current status of operation (waiting for user to enter the password, accepting it, or denying it).

The third task uses a third-party RFID library (cited below). This library only alerts our code of a card upon its entrance within range (it does not continuously output the card ID). The loop function also repeatedly checks to see if a new card has been read. If so, it proceeds as if the # key had been pressed, using the comparator circuit to determine if access should or should not be granted.

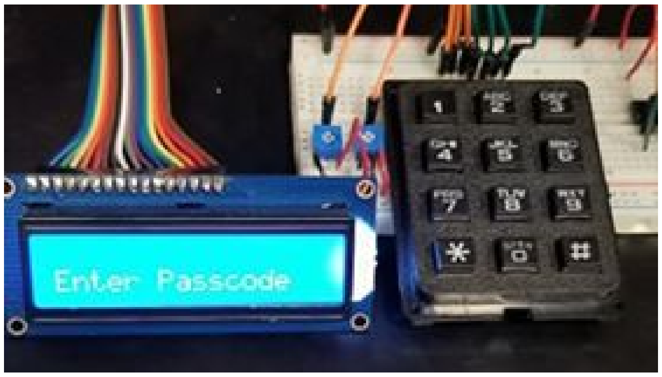
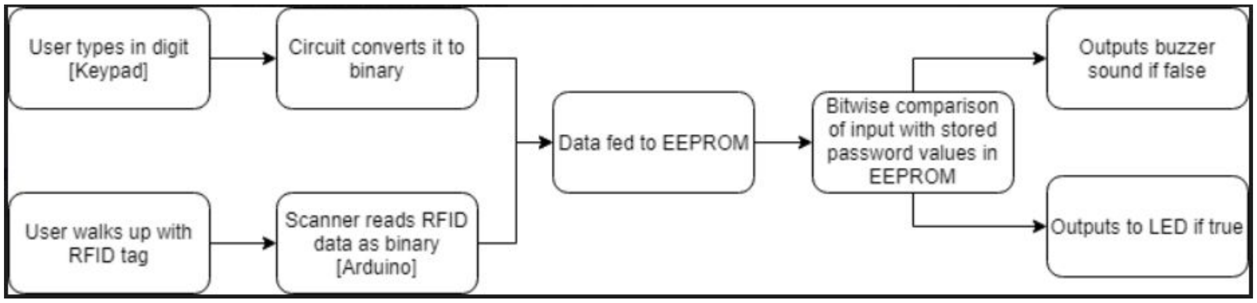


Figure 4: *LCD prompts user and displays keypad input.*

Design Description

The original design concept of the proposed solution is depicted in Figure 6. It differs from the final stages, where an LCD screen is used for output rather than a buzzer and LED. This block diagram shows that the keypad circuit receives input from the user and feeds that to the EEPROM. Alternatively, the RFID scanner picks up the signal from the tag and send its ID to the EEPROM. The Arduino then sends the bits from memory to a comparison circuit, and based on the result of the comparison, the LCD displays the relevant message.



User types in digit [Keypad]
Circuit converts it to binary
User walks up with RFID tag
Scanner reads RFID data as binary [Arduino]
Data fed to EEPROM
Bitwise comparison of input with stored password values in EEPROM
Outputs buzzer sound if false
Outputs to LED if true

Figure 5: *Basic structure of the circuit.*

Bitwise Comparison

The passwords are 4 characters long, requiring 32 bits each. With this in mind, a parallel comparison circuit would require more pins on the arduino than possible. To work around this, a serial comparison circuit was built.

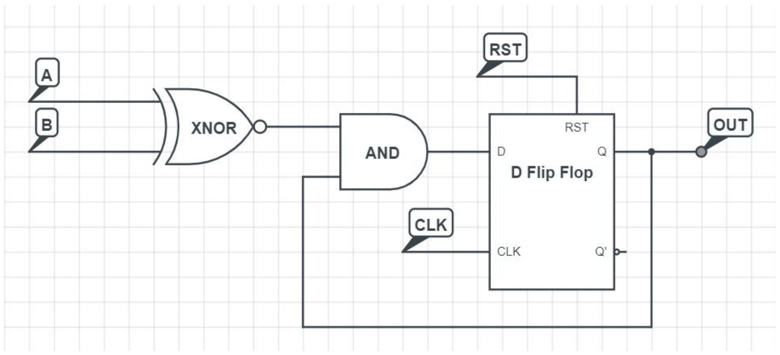


Figure 6: *Full Comparison Circuit Diagram.*

The XNOR gate is the bulk of the comparison, outputting HIGH when both incoming bits are the same. All labeled nodes in Figure 6 are outputs from the Arduino with the exception of OUT, which is an input to the Arduino. The D-type Flip Flop allows for every bit to be compared on the same circuit, serially. It stores a bit which is HIGH if all bits so far have been the same and LOW if any one bit is ever not the same. This is accomplished with the AND gate, compounding the stored value with itself, assuring that a HIGH output is unstable, and a LOW signal is stable. The flip flop is reset before each comparison of passwords, but not between each bit.

The XNOR and AND gates on this circuit are actually built from XOR and NAND gates, as those were the gates at our disposal.

Troubleshooting

While implementing the design, there were a couple of issues that required special attention. Certain elements of the circuit and functionalities were not compatible with the original design. In these cases, a new design had to be created.

One such change was made regarding the original design for the logical element of the project, which depicted a parallel bitwise comparison of the passwords. However, even the Arduino Mega did not have enough pins to support the parallel comparison and the RFID scanner. As a result, a second comparison circuit was designed to operate serially. Although a serial design is typically slower than a parallel design, for this project, the timing difference is so small it is negligible. Additionally, the serial design required less area, saving breadboard space, wires, and chips that would have been used in parallel.

Another change in the original design was in the output of the circuit. Originally, a buzzer and LED were planned to work along with the output on the LCD to either grant or deny access when a password is entered. After implementing the serial comparison design, however, because of the timing aspect of the flip-flop and the need to finish other important functions of the project, it was determined that the LED and buzzer were superfluous. Instead, the LCD would simply display the appropriate message.

Conclusion

This project gave us practical experience with circuitry in general and the Arduino in particular. We learned about the different functions of Arduino pins (whether they are digital or analog I/O, power, or controls), we researched RFID technology and SPI communication to be able to use them in our project, learned how to use basic Arduino libraries, and, most impactfully, we gained first-hand experience with the benefits and detriments of serial and parallel design.

Initially, we wanted to use a parallel design because it wouldn’t require any type of memory or the generation of a clock signal. However, as we sought to expand our system to work with larger passwords, it quickly became impossible to fit the size of the circuit in with the Arduino because we ran out of I/O pins. By switching to a serial design we drastically reduced the size of our circuit, and the time it takes to complete a comparison increased, but not by a noticeable amount.

In the end, we were able to successfully complete almost all of our goals. We did not include a buzzer or LED as output sources, which would have been nice to do in tandem with our substitute, the LCD. However, our final product delivers on every functionality goal we made, taking both keypad and RFID input and correctly processing that input to determine if access should be granted or not. We even developed a relatively realistic password setting and resetting scheme, putting our product a step above a basic prototype.

References

Libraries and Documentation

- RFID: https://github.com/miguelbalboa/rfid
- EEPROM: https://www.arduino.cc/en/Reference/EEPROM
- SPI: https://www.arduino.cc/en/reference/SPI
- LCD: https://www.arduino.cc/en/Reference/LiquidCrystal
- Keypad: https://playground.arduino.cc/Code/Keypad/