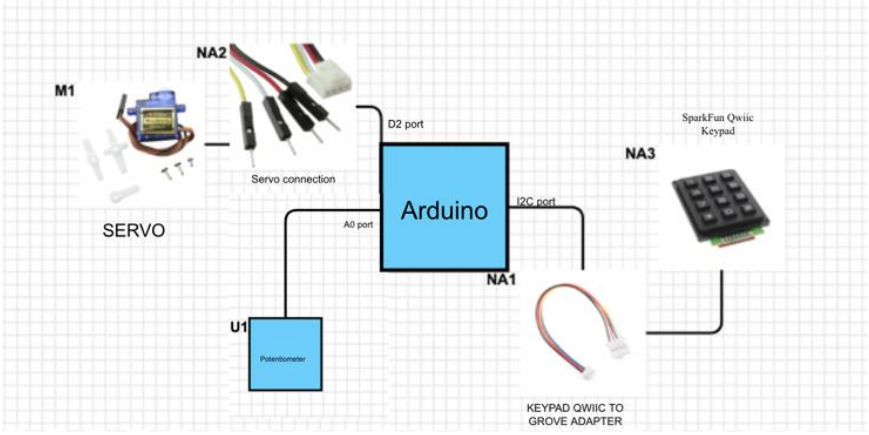
Matteo Pulcini
217 536 756
EECS 1021
Tuesday April 12th, 2021

INTRODUCTION
With the growing prevalence of technology in our society, security and privacy have become more important. Storing personal belongings and accessing information should be protected without the risk of being compromised. Banks store information in safe places behind a vault that cannot be accessed. I tried to replicate this idea with a continuous Arduino program verifying password combinations and granting users access with specific input, all while storing user access times in Java.

CONTEXT:
This project was a lockbox which was representing the importance of the connection between software and hardware that has been introduced into the security industry for recent decades. I implemented more actions by adding a new library called Twilio which allowed me to send SMS messages directly to my phone whenever someone tries to run the java program and access my live records and admin controls. This is important and an excellent example of the social context of security because it allows us as humans to be notified when someone is trying to access our personal information in a manner that is understood by a major of people, i.e., SMS. Organizations that want to hide their customers' information and make it more difficult for wrong users to access that information. I hope I was able to demonstrate this through my implementation of Twilio, which notifies me whenever I am that someone is trying to access my belongings.

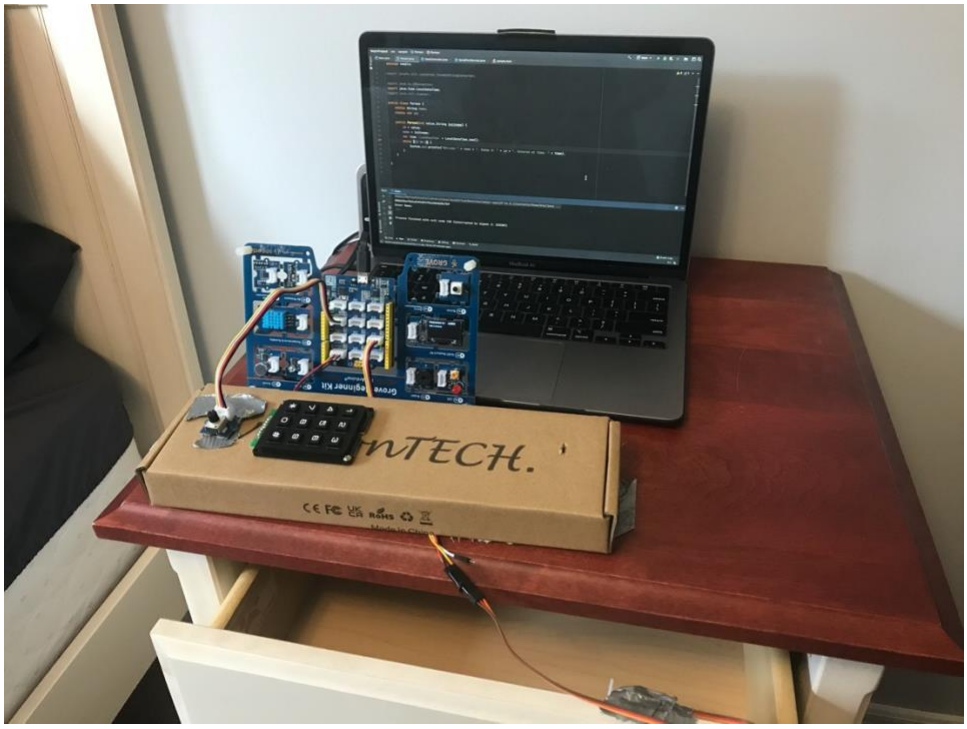
TECHNICAL REQUIREMENTS / SPECIFICATIONS
The Arduino should prompt the user to input a password displayed in the OLED. The user should input a password. The Arduino will compare the password entered to the stored password. The user will have to move the potentiometer which represents an employee id. Java prompts the user to enter their name and sends them a welcome notification with their badge number, time entered, and name. Then the representation of a badge number and time are display in a live graph in Java when the user enters the correct password of 1234. This acts as a representation for keeping information secure and by monitoring who accesses information and by keeping a record of who had access. Then the servo should rotate to a position that allows the door to open if the password was correct. If the password is incorrect the servo will display a message to the display which states the user has n tries remaining, decreasing n from two through zero. In JavaFX, an operator can remove all access by clicking a button called “remove access” in case of a security breach. If the user enters in the correct password and this button has pressed a red LED turns on while printing “locked” to the OLED, before running an infinite while loop (while (1)), to freeze the program until it is reset manually. Also, we someone accesses my java program by running it, I receive an SMS on my phone.



Basic Schematic

COMPONENTS LIST: [as you built the device]

- Servo Motor, acts as lock
- Table drawer, stores belongings
- Potentiometer
- SparkFun Qwiic Keypad - 12 Button - COM-15290, to enter password
- KEYPAD QWIIC TO GROVE ADAPTER, to connect keypad to Arduino
- Arduino Uno
- JavaFX
- USB-C to USB adapter
- Cardboard box


Project Image

PROCEDURE:
Creating this project started with the simple idea of creating a lockbox. In creating a lockbox, I used methods and applications that we have learned in class such as event-driven actions. I used them to open my lockbox if the entered password was the same as the password stored in memory. I used APIs (jSerialComm library) to communicate between JavaFX and the Arduino. This was done by displaying login information in a table in real-time when a user successfully logged in with their identification displayed. I used accessors, constructors, classes, scanners, user inputs, extra libraries, live graphs, servo motors. To use all of these I compared how it was done in previous labs and classes and modified work to be different and unique. I believe this project to be exceeding expectations as I researched and implemented libraries that were never before, something real work engineers do when using preconstructed solutions. I used Twilio to send a notification to my phone when a user tried to access the Java Program and I also implemented features that were recently taught i.e., constructors and accessors (Person.java) which I hope to use more of in the future.

TEST:
Test 1
This was the main test to determine if my equipment was comprised. I purchased the Servo motor and the keypad from digikey and had to demonstrate that they functioned correctly. To do this I downloaded sample programs to test each device separately, making sure the connections were set up. Thanks to Arduino's example Sweep, Figure (1). Thanks to Nathan Seidle, at SparkFun Electronics for this example code to demonstrate if the keypad is operating expectedly, Figure (2). Both pieces of equipment ran expectedly. This process saved me a lot of time by demonstrating the technology was indeed working and problems that I would most likely encounter would be with software.

Sweep §

```
#include <Servo.h>
Servo myservo;
int pos = 0;
void setup() {
  myservo.attach(2);
}
void loop() {
  for (pos = 0; pos <= 180; pos += 1) {
    // in steps of 1 degree
    myservo.write(pos);
    delay(15);
  for (pos = 180; pos >= 0; pos -= 1) {
    myservo.write(pos);
    delay(15);
  }
}
```

Figure (1)

```
#include <Wire.h>
#include "SparkFun_Qwiic_Keypad_Arduino_Library.h"
KEYPAD keypad1; //Create instance of this object

void setup(void)
{
  Serial.begin(9600);
  Serial.println("Qwiic KeyPad Example");

  if (keypad1.begin() == false)   // Note, using begin() like this will use default I2C address,
                                  // You can pass begin() a different address like so: keypad1.begin(Wire, 0x4A).
  {
    Serial.println("Keypad does not appear to be connected. Please check wiring. Freezing...");
    while (1);
  }
  Serial.print("Initialized. Firmware Version: ");
  Serial.println(keypad1.getVersion());
  Serial.println("Press a button: * to do a space. # to go to next line.");
}
void loop(void)
{
  keypad1.updateFIFO();  // necessary for keypad to pull button from stack to readable register
  char button = keypad1.getButton();

  if (button == -1)
  {
    Serial.println("No keypad detected");
    delay(1000);
  }
  else if (button != 0)
  {
    if (button == '#') Serial.println();
    else if (button == '*') Serial.print(" ");
    else Serial.print(button);
  }
  delay(25); //25 is good, more is better
}
```

Figure (2)

Test 2

The second tests were smaller and throughout the whole process which determined if the code was doing what it was intended to do. This was determined by sending time and potentiometer data to Java, the graph, similar to Lab I with a few of my implementations. This was shown in Figure (3) where I determined if my results were the same as the lab or if they were of my implementations, the different non-constant time intervals, were me logging in with different ID at the time I logged in, unlike the lab. I also needed to determine if the buttons in Java were being listened to and sent to the Arduino to lock the program or to turn on the red LED. I put a label on the left side of the screen when the "Remove Access" button is pressed "OFF", which blocks the user only if they enter a correct password. Figure (4) is an example of a function that was separated from the main loop, allowing me to easily determine if something was wrong, and if so where it was. I had in total around 6 functions that all allowed me to identify which parts of my code were wrong when I was debugging, and this was important for sorting my code and finding problems and where they occurred.

Figure (3)

```
void checkAdmin(){ //check Admin external access
const auto recievedData = Serial.read(); //assign data from Java
char buf[16];                            // format the data
sprintf(buf, "%03d", recievedData);
display.println(recievedData);

    if (recievedData == 48) {
      display.clear();
      display.println("LOCKED");
      while(2>1){
        delay(1000);
        digitalWrite(REDLED, HIGH);
      }
    }
}
```

Figure (4)

LEARING OUTCOMES:

1. Demonstrate the ability to test and debug a given program and reason about its correctness.
   - This project was tested throughout each step of the implementation process, mainly through being proactive by creating functions. When problems arise my work was organized, and I could visualize where code was running and where it was going wrong by commenting out specific sections. I printed output to the OLED and to the command window to also see the information my code was receiving and when it was receiving it, so I could debug.

2. Given a problem specification and a suitable API, build an application that meets the given requirement. (GAI 4b: Conceive design solutions to solve the defined problem)
   - The problem was safety and introducing hardware and software to store my information in a safe place. I decided to implement a solution that would make my drawer more secure by installing a servo motor and a keypad (controlled by Arduino and java) which I used to limit access and keep a record of who accesses my belongings in the desk in Java.

4. Build an event-driven application that controls sensors and actuators in order to connect events to physical actions. (GAI 4b)
   - The events and corresponding actions I used were. Keypad displaying entered password on OLED. Potentiometer controlling spinning of servo motor. JavaFX, button controlling OLED display "blocked" and turning on red LED. These are all events followed by the action.

5. Program common applications from a variety of engineering disciplines using an object-oriented language and solve them on the computer. (GAI 4c)

- Common applications that can be used in a variety of engineering disciplines using object-oriented programming would be, if statements, while loops, for loops, creating constructors and accessors, and using libraries. By iterating over all of these multiple times and by using the in different ways to control my code, I was able to create a solution that solved my problem.

CONTINGENCY

I had a few ideas in mind that I would like to implement but could not be introduced due to a lack of time. Next time I will order my hardware well before the deadline so I can see its limits and be able to expand my options if I deem appropriate. Next time, I will also plan well in advance and try to better manage my time to finish up the overall project well before the deadline, allowing time for reflection.

ADDITIONAL MATERIAL

The social aspect of this project is the benefit in security with the addition of software and hardware, and the communication with the vast majority of users. By using Twilio, and sending SMS as I did, I communicated with my java program. This is a real-life example of the communication organizations use with their customers about their information, in ways in which they can understand. The idea of this project is to get an understanding of what we learned in class and develop our solutions to our problems. I used recourses and tools that we have learned in class. But I also like to think that using Twilio and adding accessors in my classes is the largest component to be reviewed for exceeding expectations. To be proficient in any software disciple you must be willing to look for solutions everywhere and do research on your own, which I believe successfully.

CONCLUSION

Causing an event-driven action and using the software and hardware to do this demonstrates the importance of automation, not limited to security. Communication between the two allows us to create new things every day. We use software to program hardware every day, and there are practically limitless opportunities, and this example only just scratches the surface.

Citations
1. article title 2. website or book or magazine name; 3. Author name(s) 4. Date (year at least) 5. URL (usually)

Send SMS and MMS Messages in Java. Twilio. 2020-09-29
https://www.twilio.com/docs/sms/tutorials/how-to-send-sms-messages-java?code-sample=code-send-an-sms-using-the-programmable-sms-api&code-language=Java&code-sdk-version=8.x

Sweep. Arduino. *SM. 2015/08/18*
https://www.arduino.cc/en/Tutorial/LibraryExamples/Sweep

Qwiic Keypad Hookup Guide. SparkFun. Nathan Seidle.
https://learn.sparkfun.com/tutorials/qwiic-keypad-hookup-guide/resources-and-going-further