PPC final - StrongBox

Olivier Donker (s2862239) & John De Zwart (s2818442)

January 2022

1 A strong box

Having discussed a few ideas, an amusing though crossed our minds: To build a locking system that would be secure to a degree of hilarity. As such, we decided on a concept using both a keypad *and* an alcohol gas sensor, so the user would have to enter a correct code and complete a "breathalyser test" in order for a lock to open. This "lock" is simply to consist of a servo that opens the lid of a box. Consider the image below for an immediate impression. Additionally,

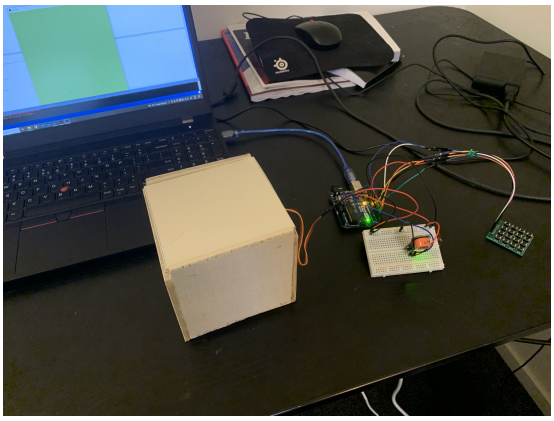

Figure 1: The finished product with box (left), Arduino (left-centre), MQ-3 Alcohol sensor (right-centre), and keypad (right).

if the user fails either of these tests, an alarm screen is shown and a sound is played in processing in order to deter the thief, along with the malicious login being logged in a `.txt` file. Passing the tests will results in a more favourable screen and noise.

2 Software

Below is all the code that was written for this project, consisting of both Arduino and Processing.

2.1 Arduino code

```
//Made by Olivier Donker & John De Zwart in January 2022.
//Inspired by https://microcontrollerslab.com/mq3-alcohol-sensor-arduino-tutorial/.

#include <Servo.h>

int ethanolSensorPin = A0;
int ethanolSensorValue;
int ethanolUpperLimit = 100; //Some number, will depend on sensor
boolean userBoozed1 = true;
boolean userBoozed2 = true;
boolean booze;
boolean alcUndefined = true;
int servoPin = 12;
Servo lockServo;

int keyPadDebounceDelay = 100;

int rowPins[] = {2, 3, 4, 5};
int columnPins[] = {6, 7, 8, 9};
int codeEnteredArray[5];
int currentCodeNumberToEnter = 0;
int keyCode[] = {1, 2, 3, 4, 5};

int padNumberMatrix[4][4] = {
  {0, 1, 2, 3},
  {4, 5, 6, 7},
  {8, 9, 0, 0},
  {0, 0, 0, 0},
};

boolean codesMatch = true;
boolean letUserInFromCode;

void setup() {
  Serial.begin(9600); //Open connection
  lockServo.attach(servoPin);  //setup servo
  lockServo.write(0); //Set it to a low position

  //Choose Arduino pin 2, 3, 4 and 5 for the rows, 6, 7, 8 and 9 for
the columns.
  //Step 1 of algorithm below.
  for (int i = 0; i < 4; i++) { //Cycle through array of length 4
    pinMode(rowPins[i], OUTPUT);
    pinMode(columnPins[i], INPUT_PULLUP);
```

```
  }
}

void loop() {
  //Algorithm to determine keypresses in the keypad matrix:
  //1. Set all pins connected to the rows to HIGH
  //2. Set the first row to LOW
  //3. Listen to every column, one after another. If the respective
key of a column is pressed, we will find LOW on that column, instead
of HIGH.
  //4. Set the first row to HIGH and set the next row to LOW.
  //5. Continue this until the fourth row has been cycled from HIGH
to LOW
  //6. Then loop this whole procedure.

  if (currentCodeNumberToEnter < 5) { //Only check for keypad inputs
if we want another number in the entered code
    for (int i = 0; i < 4; i++) { //Big for loop to run through rows,
works as step 5
      //Serial.print(i);
      digitalWrite(rowPins[i], LOW);   //Step 2 of algorithm
      for (int j = 0; j < 4; j++) {  //Step 3 of algorithm
        if (digitalRead(columnPins[j]) == LOW) { //If button pressed
there
          Serial.println(padNumberMatrix[i][j]);              //Element
(i, j) in pad numbers matrix. Works for individual pins, just not the
right number yet
          codeEnteredArray[currentCodeNumberToEnter] = padNumberMatrix[i][j];
          currentCodeNumberToEnter += 1;  //Go to the next number to
enter
          delay(keyPadDebounceDelay);
        }               //FUCKING WORKS RWOJRWAPENFPAWF
      }
      digitalWrite(rowPins[i], HIGH);    //step 4 of algorithm
      delay(keyPadDebounceDelay);
    }
  }

  else {
    currentCodeNumberToEnter = 0; //reset input
    for (int i = 0; i < 5; i++) {
      if (keyCode[i] != codeEnteredArray[i]) {
        codesMatch = false;
      }
    }
    if (!codesMatch) {
```

```
      letUserInFromCode = false; //don't allow user in
      Serial.println("Denied");
      for (int i = 0; i < 5; i++) { //clear the code entered array
        codeEnteredArray[i] = 0;
      }
      codesMatch = true; //Reset codesMatch checker
    }
    else {
      letUserInFromCode = true; //do allow user in
      for (int i = 0; i < 5; i++) { //clear the code entered array
        codeEnteredArray[i] = 0;
        codesMatch = true;
      }
    }
  }

  if (letUserInFromCode == true) {
    delay(200);
    while (alcUndefined == true) {
      ethanolSensorValue = analogRead(ethanolSensorPin);  //Read the
value
      Serial.println(ethanolSensorValue);
      //check 1 for alcohol in breath
      if (ethanolSensorValue < 500) { //Only send if there is too much
ethanol sensed. value of 940 seem to be normal for air, dipping two
200 for a shot glass of rum close by.
        userBoozed1 = true;
      }
      else {
        userBoozed1 = false;
      }
      delay(2000);
      //checks after 2 secinds to see if values remained within the
limits
      if (userBoozed1 == userBoozed2) {
        //Serial.println("boozed");
        booze = true;
        alcUndefined = false;
        Serial.println("Denied");
      } else {
        // Serial.println("not boozed");
        booze = false;
        alcUndefined = false;
      }
    }
  }
```

```
  //if all checks were passed access is granted
  if (alcUndefined == false && booze == false && letUserInFromCode
== true) {
    lockServo.write(120); //open. Use Arduino reset to close box again
    Serial.println("Access");
  }
}
```

2.2 Processing code

Main tab

```
import processing.sound.*; //sound library
import processing.serial.*; // use serial port libraries
Serial myPort;                 // make a fresh serial port
boolean access;
boolean deny;
Wrong Alarm;
Access Open;
String validate = "Access";
String state;

void setup()
{
  size(700, 700);

  Alarm= new Wrong (width, height);
  Open = new Access (width, height);

  println("Available serial ports:");

  for (int i = 0; i<Serial.list().length; i++) {
    print("[" + i + "] ");
    println(Serial.list()[i]);
  }
  myPort = new Serial(this, Serial.list()[0], 9600); // open port 0
in the list at 9600 Baud
  myPort.bufferUntil(10);
}

void draw()
{
  if (state!=null && state.equals("Access") == true ) {
    Open.display();
    Open.check();
  }

  if (state!=null && state.equals("Denied") == true) {
    Alarm.popUp();
    Alarm.securityLog();
    Alarm.alert();
  }
}

void serialEvent(Serial p) {
```

```
  validate = p.readStringUntil(10);
  println(validate);
  state=trim(validate);
}
```

access tab

```
PImage checkmark;
SoundFile granted;
boolean playing=false;

class Access {
  float initX;
  float initY;

  Access(float x, float y) {
    initX=x;
    initY=y;

    checkmark = loadImage("checkmark.png");

    background(152, 190, 100);

    granted = new SoundFile(security.this, "granted.mp3");
  }

  void display() {
    pushMatrix();

    translate(initX/8, initY/8);
    image(checkmark, 0, 0);
    fill(0);
    text("ACCESS GRANTED", 200, 550);

    popMatrix();
  }
  void check() {

    if (!playing) {
      playing = true;
      granted.play();
    }
  }
}
```

wrong tab

```
SoundFile alarm;

class Wrong {
  float initX;
  float initY;
  Wrong(float x, float y) {
    initX=x;
    initY=y;

    alarm = new SoundFile(security.this, "alert01.mp3");
  }

  void popUp() {
    float posChangeX;
    float posChangeY;

    posChangeX = random(-initX, initX);
    posChangeY = random(-initY, initY);

    fill(255);
    strokeWeight(10);
    stroke(255, 0, 0);

    pushMatrix();
    translate(posChangeX, posChangeY);
    triangle(-100, 100, 100, 100, 0, -100);
    fill(0);
    text("ACCESS DENIED", -50, 150);
    popMatrix();
  }

  void alert() {

    if (!playing) {
      playing = true;
      alarm.loop();
    }
  }

  void securityLog() {
    int min = minute();
    int h = hour();
    int d = day();
    int mon = month();
```

```
    int y = year();
    String attempt="login attempt on: "+ str(y) + "/" + str(mon)+ "/"
+ str(d)+ "/" +str(h)+ ":" +str(min);
    String[] list = split(attempt, " ");
    saveStrings("login4.txt", list);
  }
}
```

3 Hardware

The electronics setup that was designed and constructed is as follows:

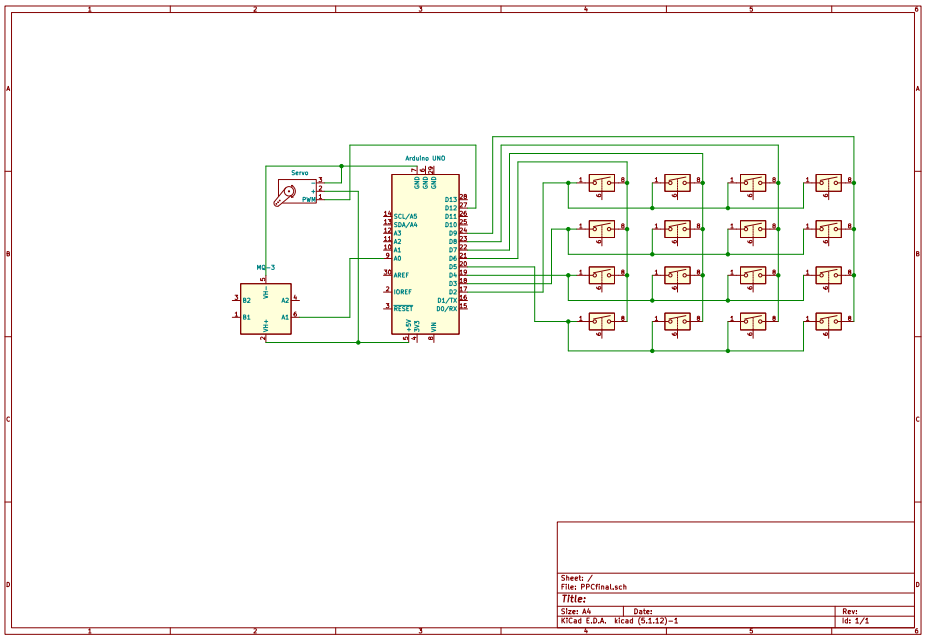

Figure 2: Caption

4 Reflection

It is very rare that things develop according to plan, and this instance was no exception. A very large factor was something of a lack of planning in advance; We did not consider that we needed very specific connectors for the keypad, so we had to postpone handing the project in, borrowing them in the meantime. A second issue was Wi-Fi: We originally planned to have the Arduino communicate with Processing over Wi-Fi, but this turned out to be rather too complicated, along with the fact that the eduRoam network has a special kind

of authentication. Finally, (this is not much of a mistake yet still insightful), getting the keypad to work was much more difficult than expected, requiring a little algorithm.