CT4021 Introduction To Programming Fundamentals

Assignment 2 - A Personalized Home Security System

Sam Slade

Introduction

The product developed in this project is a home security system. It allows for the locking of a door with a servo motor, which can be opened with use of a key code. The key code can be entered on the security systems keypad. A small LCD is used to display messages indicating the current state of the security system, i.e. is the door locked or unlocked. Further feedback is given to the user by a small piezoelectric speaker and different coloured LEDs. These are used to communicate when buttons on the keypad are pressed and when the door is locked and unlocked. The most important component of the entire project is the Arduino Uno board, it's the brains of the entire operation and controls the input and output of every other component used. The use of the Arduino also allows for multiple codes to be stored and for the codes to be edited all without the need for rebooting or reflashing of memory.

During the development of the project a number of challenges were faced, some more difficult than others. A recurring issue was the lack of digital input and output pins (which will be referred to as I/O pins from now) on the Arduino Uno board. One solution to this problem would have been to move away from the Arduino Uno board to an Arduino Mega which has many more digital I/O pins allowing for the use of more components. However, the solution chosen to this issue was to use a 74HC595N shift register, more will be spoken about the shift register in the design section of this report. As well as the use of the shift register, wiring choices were carefully considered, by the end of the project the two LEDs used didn't use any of the digital I/O pins but would still light up when appropriate.

Design Of The System

Hardware

While developing this project, an agile-like methodology was adopted. Each two week sprint ended with a review of what had been done towards the project and what would be done in the next sprint. Focus shifted from sprint to sprint but each had to produce something useable towards the final product. Keeping track of tasks in each sprint and between sprints was made easier with tools such as Trello which uses the Kanban system for organising tasks.

In the earlier sprints, the focus was on understanding each of the components that would be used in the final product. As well as gaining understanding, the early sprints were also used to produce code blocks that could be utilised in the final code build. Later sprints looked at assembling what was learnt into the final circuit. Figure 1 shows one of the first designs for the circuit taken from an early sprint, there are a number of issues in this design that will now be discussed before looking at the final design to show the changes that were made throughout the development cycle. This initial design shows the LCD, keypad, piezoelectric speaker and LED all wired to the Arduino Uno. This design didn't include the servo motor required by the design specification as there weren't enough digital I/O pins to support it. Other issues with this design included the use of digital pins zero and one. Using pins zero and one can cause an issue where, during the flashing process, the program counter becomes de-synchronised, this is caused by the tiny amount of resistance of the wires being in pins zero and one. The wires could be removed each time the Arduino's memory is written to but this later became more difficult as larger numbers of wires were added to the system. In a later version, it was decided to remove the

use of pins zero and one to completely avoid this issue. The initial design also included only one LED, which was okay but to allow for better indication of what state the system is in, there was a need for more LEDs of different colours.

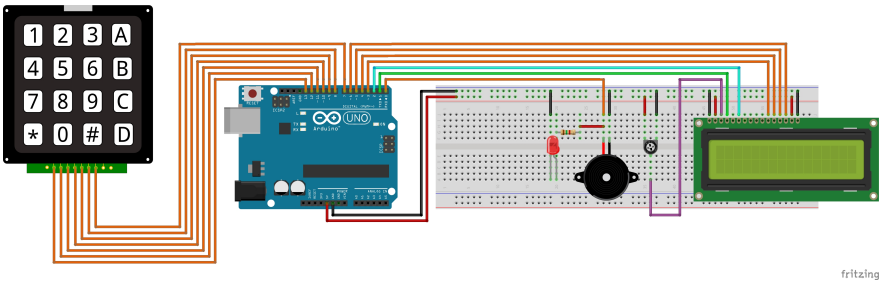

Figure 1: Initial design using Fritzing

The final design reached sought to correct each of these issues, Figure 2 shows the final design. The largest change from the initial design is that the LCD is no longer directly connect to the Arduino. Instead, the Arduino connects to a 74HC595N shift register which then connects to the LCD. lastminuteengineers (no date), sates that the [shift register] essentially controls eight separate output pins, using only three input pins. Normally, the LCD requires seven connections to the data pins on the Arduino, using the shift register allows for the LCD to be controlled using only three digital I/O pins. This means four more connections can be made to the Arduino while losing no functionality and freeing up digital I/O pins zero and one. The digital I/O pins used by the shift register had to be pins nine, eleven and thirteen, as these are the digital I/O pins that provide the Serial Peripheral Interface (SPI). SPI is a synchronous serial communication method, which the shift register requires in order to send the data correctly onto the LCD. Getting the shift register to work with the LCD also meant using a modified LCD header file provided by Juan Hernandez (Hernandez, 2016). The final design also makes use of multiple LEDs which are connected to the piezoelectric speaker's and the servo motor's digital I/O pin as well. This means that by carefully controlling when the piezoelectric speaker is used, the red LED will light up and when the servo motor is used, the green LED will light up. An RGB LED was originally planned to be used, but due to a wiring error, the LED was damaged and couldn't be used in the final product.

As was mentioned in the previous paragraph, the servo motor was also implemented in the final design using the extra digital I/O pins available. The servo motor is used to open and close the lock in the door, it is controlled by the Arduino and when the correct code is entered on the key pad, then the servo motor will open. The keypad, in both the initial and final design isn't fully connected. The right most column (containing the buttons A, B, C and D) is not connected, this is to free more I/O pins to be left available for other components.

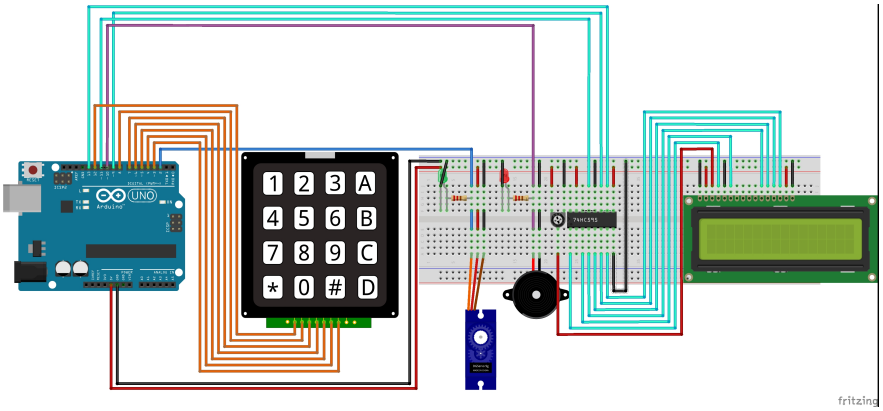

Figure 2: Final design using Fritzing

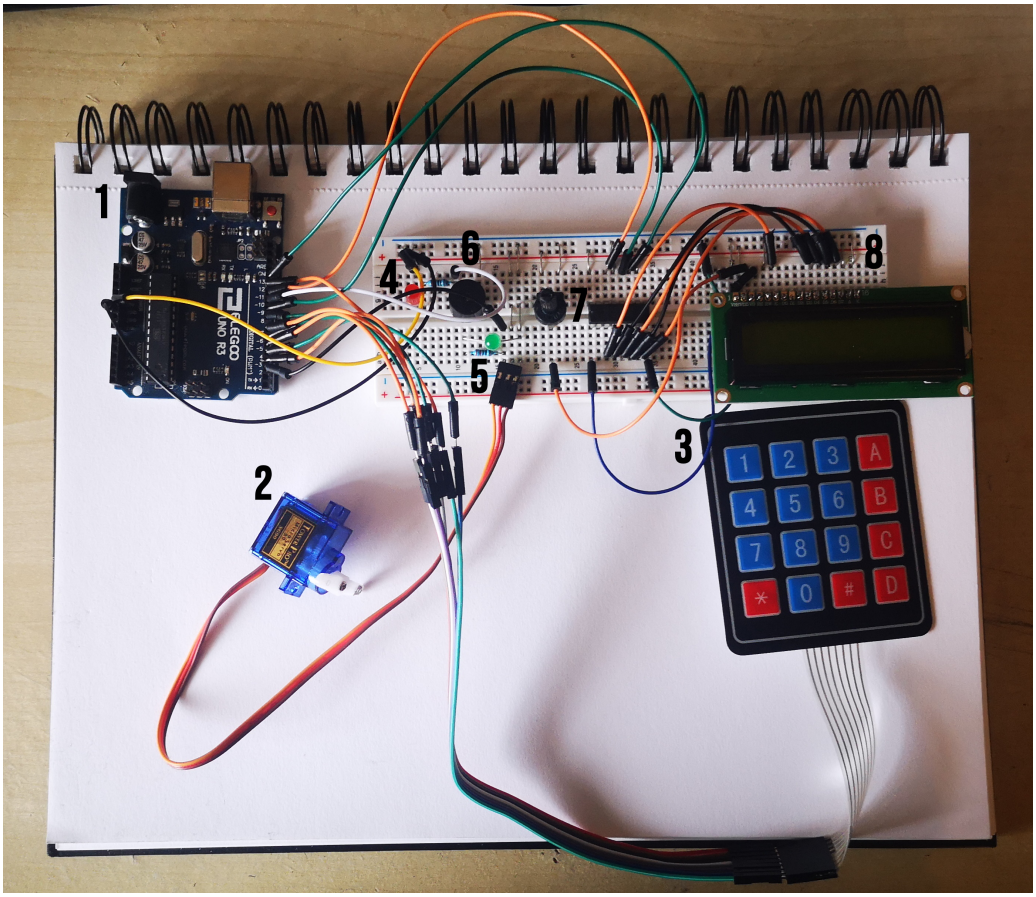

Figure 3: Labelled circuit of the final design

1. Arduino Uno
2. Servo Motor
3. Keypad
4. Red LED
5. Green LED
6. Piezoelectric Speaker
7. 74HC595 Shift Register
8. LCD

Software

When developing the software for the project, it was decided early on to use heap allocated arrays to store the codes entered by the user. This choice allowed for better control of when and where memory was allocated and de-allocated, this is important as Guyer and Helbling (2016) note, typically [the Arduino has] 32 KB of flash memory for the program and 2 KB of RAM for both the stack and heap. These limitations place significant constraints on how the boards are programmed.

Listing 1: Declaring pin code storage variables
(assignment_code.ino line 50-58)

```
/*==== Pin code storage ====*/
// An array of pointers to the pin codes
char** pinCodeArray = new char*[10];

// Array used to store user input
char* input = new char[4];

// Value to keep track of which digit has been entered into the "input" array
int pointer = 0;
```

Listing 1 shows the variable `pinCodeArray` being defined, this is an array of char pointers to other char pointers, often called a multi-dimensional array. The array will be used to store the different user codes. This method does require more memory with the use of pointers to each array, but it does remove the need for a larger number of named variables, which was considered to be a valuable trade off. The variable `input` was also defined here, which is a char pointer that can hold four characters. This is used to store the code being entered by the user on the keypad. The variable `pointer` is used to keep track of which digit has been entered by the user. Listing 2 shows the Arduino `setup` function and how it was used to set the user pin code array. Also in the setup function, line 2 is for setting up the serial debugging mode, line 7 is to initialise the LCD with a display size of two lines and 16 characters per line and line 10 is to set up the pin connected to the piezoelectric speaker. From line 14, the pin code array is being setup, 14 to 16 is a for loop that gives each of the pin code array elements a pointer to a four character array. These four character arrays are the codes used to unlock the system. Lines 18 to 23 are a set of nested for loops used to set each of the values in the four character arrays to be `NULL`, this is done to remove the risk of a memory leak. Finally, lines 27 to 35 are used to set the admin code which is used to edit the other codes in the system. The admin code is set to the value `0, 1, 2, 3` using a for loop and the `itoa` function. This function takes three parameters, an integer, a char pointer and a second integer which is used to describe the base of the first integer. The function converts the first integer into a string and stores it in the character pointer. The value in the character pointer is then stored in the four character array at index zero in the pin code array, this can be seen on line 33.

Listing 2: Setting up the code storage array and setting the default admin pass code (assignment_code.ino line 77-111)

```
1  // Arduino setup function, runs when the arduino starts
2  void setup() {
3    // Allow for listening to the serial bus
4    Serial.begin(9600);
5
6    // Set LCD Screen dimensions
7    lcd.begin(16,2);
8
9    // Set the buzzer pin to output
10   pinMode(BUZZER, OUTPUT);
11
12   // Give a pointer to an empty array to each of the elements
13   // in the pincode array
14   for (int i=0; i<10; i++) {
15     pinCodeArray[i] = new char[4];
16   }
17
18   // Set each array element to be NULL
19   for(int i=0; i<10; i++) {
20     for (int j=0; j<4; j++) {
21       pinCodeArray[i][j] = NULL;
22     }
23   }
24
25   // Setup admin code, always stored in the 0 element of the
26   // pin code array
27   for (int i=0; i<4; i++){
```

```
28      char int_str [1];
29
30      // itoa converts an int (i in base 10) to a string and stores
31      // it in int_str
32      itoa(i, int_str, 10);
33      pinCodeArray[0][i] = int_str[0];
34    }
35  }
```

The largest piece of the source code is taken up by the `admin_menu` function. The code snippet for the `admin_menu` function is too large for this section of the document and can be found in Appendix 0.3 at lines 282 to 429. The function allows the user to edit codes on the system and can be accessed by entering the admin code on the keypad. To do this, the function first enters a while loop, this is used to keep checking key presses from the user, from here the user can be sent back to the main locked screen or enter the next while loop and choose a code, number 0 through 9, to edit. When the user enters a valid number the next while loop is entered, this prompts the user to enter a new code and then repeat the new code. The new code and the repeated code are store inside heap allocated arrays, to protect the system from a memory overflow, the two arrays are deallocated on lines 359 and 360. Lines 339 to 374 are used to store the new codes that are entered by the user. The two codes are first checked to see if they are the same, if not then the user is prompted to enter new codes. If the two codes entered are identical then, using a for loop, the new code is copied into the `pinCodeArray` variable, the new code arrays are deallocated and then the function returns.

Testing The System

The testing process for the product began with the writing of user stories. The user stories can be found in Appendix 0.4, these detail the criteria that the product should meet to be feature complete. From the user stories, a set of unit tests were developed. The unit tests can be found in Appendix 0.5. The first unit test 001, concerns the admin menu, for this function to succeed the system needs to identify the admin code correctly and then take the users input for the new codes and store them correctly. This test covers a large portion of the code base, including the feedback functions and the logic inside the `admin_menu` function. Unit test 002, is then making sure that the user cannot enter an invalid admin code and gain access to the admin menu, this again tests the systems logic to make sure the code is checked properly. Test 003 and 004 move to check the user codes which open the door but don't allow access to the admin menu. Test 003, checks to see if a valid code works and so covers code validation as well as checks to see if the servo motor functions properly. Test 004 then makes sure that an invalid code cannot be used. Test 005, will have been partially tested by the other unit tests but tests the systems feedback to the user from pressing buttons on the keypad. These unit tests cover the main cases for using the system and checks nearly the entire code base. There maybe edge cases which haven't been considered which is why there is hesitance to commit to all of the code being fully checked. However, each key area of function has been unit tested individually and all have succeeded.

Appendix

0.1 Evidence Video

https://www.youtube.com/watch?v=eQMCCoCKMT0t=6s

0.2 User Guide

0.2.1 System diagram

0.2.2 Unlocking a door using the system

To unlock a door using the system, a valid *user code* must be entered. A *user code* can be set in the admin menu, details for doing this can be found in section 0.2.5. The *user code* is entered using the keypad. The number of digits the user has entered will be displayed on the LCD. When a valid user code is entered, the LCD will display a "SUCCESS" message accompanied by the green LED lighting up. Finally, the servo motor will move into the unlocked position. After a few seconds, the servo motor will return to the locked position.

0.2.3 Opening and closing the admin menu

To open the admin menu, the *admin code* must be entered, by default the *admin code* is `0, 1, 2, 3`. The *admin code* is entered on the keypad. When the valid *admin code* is entered, the admin menu is then displayed on the LCD.

To close the admin menu, simply press 2 on the keypad.

0.2.4 Changing the default admin code

To change the default *admin code*, first navigate to the admin menu, details on how to do this can be found in section 0.2.3. After navigating to the admin menu, press 1 on the keypad, this will begin the process of editing a code. When prompted to choose the code to edit, press 0 on the key pad to edit the *admin code*. Follow the prompts on the LCD by entering the new *admin code*, then re-entering the *admin code*. After the *admin code* has been entered and re-entered successfully, you will be taken back to the main screen on the LCD. If an invalid code is entered, you will be asked to re-enter the number of the code to edit.

0.2.5 Adding and changing a user code

To add and change a *user code*, first navigate to the admin menu, details on how to to do this can be found in the section 0.2.3. After navigating to the admin menu, press 1 on the keypad, this will begin the process of editing a code. When prompted to choose the code to edit, enter a number between 1 and 9 on the keypad to edit or create the *user code* assigned to that number. Follow the prompts on the LCD by entering the new *user code*, then re-entering the *user code*. After the *user code* has been entered and re-entered successfully, you will be taken back to the main screen on the LCD. If an invalid code is entered, you will be asked to re-enter the number of the code to edit.

0.3 Code

Listing 3: assignment_code.ino

```
/*==== Header files ====*/
#include <Keypad.h> // Keypad header file
#include <Servo.h> // Servo motor header file
#include <SPI.h> // SPI to all for talking to the lcd using a shift register
#include <LiquidCrystal.h> // LCD library header file

/*==== Servo Setup ====*/
// Set servo pin to be digital pin 2
int servoPin = 2;
```

```
10
11  // Servo object
12  Servo servo;
13
14  // Declare the two servo functions used
15  void open_door();
16  void close_door();
17
18  /*==== Keypad setup code ====*/
19  // The keypad dimensions
20  const byte ROWS = 4;
21  const byte COLS = 3;
22
23  // Keypad array of buttons
24  char keymap[ROWS][COLS] = {
25    {'1', '2', '3'},
26    {'4', '5', '6'},
27    {'7', '8', '9'},
28    {'*', '0', '#'}};
29
30  // The digital pins used by the keypad
31  byte rowPins[ROWS] = {12,8,7,6};
32  byte colPins[COLS] = {5,4,3};
33
34  // Create keypad object
35  Keypad keypad = Keypad(makeKeymap(keymap), rowPins, colPins, ROWS, COLS);
36
37  /*==== LCD ====*/
38  // LCD object, attack the data pin to pin 9
39  LiquidCrystal lcd(9);
40
41  /*==== Buzzer tone function declaration ===*/
42  // Buzzer pin
43  const int BUZZER = 10;
44
45  // Declare the three functions used by the buzzer
46  void buzzer_success();
47  void buzzer_fail();
48  void buzzer_tap();
49
50  /*==== Pin code storage ====*/
51  // An array of pointers to the pin codes
52  char** pinCodeArray = new char*[10];
53
54  // Array used to store user input
55  char* input = new char[4];
56
57  // Value to keep track of which digit has been entered into the "input" array
58  int pointer = 0;
59
60  /*==== Pin code function declaration ====*/
61  // Clear the user input code
62  void clear_code();
63
64  // Check to see if user input is a number
65  bool is_num(char key);
66
67  // Hide the user input with *'s
68  void hide_input();
69
```

```
// Check to see if the code is valid
bool check_code(int code);

// The administrator menu function
int admin_menu();

/*==== Main arduino functions ====*/
// Arduino setup function, runs when the arduino starts
void setup() {
  // Allow for listening to the serial bus
  Serial.begin(9600);

  // Set LCD Screen dimensions
  lcd.begin(16,2);

  // Set the buzzer pin to output
  pinMode(BUZZER, OUTPUT);

  // Give a pointer to an empty array to each of the elements
  // in the pincode array
  for (int i=0; i<10; i++) {
    pinCodeArray[i] = new char[4];
  }

  // Set each array element to be NULL
  for(int i=0; i<10; i++) {
    for (int j=0; j<4; j++) {
      pinCodeArray[i][j] = NULL;
    }
  }

  // Setup admin code, always stored in the 0 element of the
  // pin code array
  for (int i=0; i<4; i++){
    char int_str [1];

    // itoa converts an int (i in base 10) to a string and stores
    // it in int_str
    itoa(i, int_str, 10);
    pinCodeArray[0][i] = int_str[0];
  }
}

void loop() {
  delay(200);

  // Clear the LCD and display the menu text
  lcd.clear();
  lcd.setCursor(0,1);
  lcd.print("# to clear!");
  lcd.setCursor(0,0);
  lcd.print("Enter code: ");

  // Hide the users input
  hide_input();

  // If the user has entered 4 digits
  if (pointer == 4) {
    pointer = 0;

```

```
    // Check the users code against each of the codes in
    // the pin code array
    for (int i=0; i<10; i++) {

      // If the code is valid...
      if (check_code(i)) {

        // The 0 code is the admin code
        if (i == 0) {

          // Make success tone and display success message
          buzzer_success();
          login_success();

          // Open the admin menu
          admin_menu();
          break;

        // If is user code
        } else {

          // Make success tone and display success message
          buzzer_success();
          login_success();

          // Open the door with the servo
          open_door();
          delay(2000);

          // Clse the door with the servo
          close_door();
          delay(2000);
        }
      }
    }
  }

  // Get key pressed by the user
  char key = keypad.getKey();

  // If the # key is pressed
  if (key == '#') {

    // Make fail tone
    buzzer_fail();

    // Clear the code entered by the user
    clear_code();

  // If a key is pressed
  } else if (key) {

    // Make tap sounds
    buzzer_tap();

    // If the key pressed is a numeric key
    if (is_num(key)) {

      // Add the key to the input array
      input[pointer] = key;
```

```
190
191       // increment the pointer
192       pointer++;
193     }
194   }
195 }
196
197 /*==== Pin code function definition ====*/
198 // Clear the code entered by the user
199 void clear_code() {
200   for (int i=0; i<4; i++) {
201     input[i] = "";
202   }
203   pointer = 0;
204 }
205
206 // Check to see if key pressed is a number
207 bool is_num(char key) {
208
209   // loop through numbers 0-9
210   for (int i=0; i<10; i++) {
211
212     // int_str is used to store the int converted to str
213     char int_str [1];
214
215     // Convert int to string
216     itoa(i, int_str, 10);
217
218     // If valid, return true
219     if (key == int_str[0]){
220       return 1;
221     }
222   }
223
224   // If the for loop ends, then key isn't a number
225   // return false
226   return 0;
227 }
228
229 // Print a * to the lcd for ever key entered into
230 // the user code
231 void hide_input() {
232   if (pointer != 0) {
233     for (int i=0; i<pointer; i++) {
234       lcd.print("*");
235     }
236   }
237 }
238
239 // Check to see if the code entered by the user is valid
240 // parameter is the code to check the user code against
241 bool check_code (int code) {
242
243   // In this function the code is valid until it is proven
244   // to be false
245   int valid = 1;
246   for (int i=0; i<4 ; i++) {
247
248     // If the digits don't match, set valid to false
249     if (pinCodeArray[code][i] != input[i]) {
```

```
      valid = 0;
    }
  }
  return valid;
}

// Function gives feedback to the user to show a
// successful login
void login_success () {

  // Clear the lcd and write message to screen
  lcd.clear();
  lcd.print("### Success ###");
  delay(1000);
}

// Function gives feedback to the user to show a
// failed login
void login_fail () {

  // Clear the lcd and write message to screen
  lcd.clear();
  lcd.print("### Failed ###");
  delay(1000);
}

// The administrator menu function is where the user
// can edit codes used by the system
int admin_menu () {

  // Valid is used to check the new code entered in the admin menu
  // is valid
  int valid = 0;

  // First while loop for the admin menu system
  while (1) {

    // Clear the LCD and display the admin menu ui
    lcd.clear();
    lcd.print("press 1:EDIT");
    lcd.setCursor(0,1);
    lcd.print("2: EXIT MENU");

    // Listen for user key input
    int key = keypad.getKey();

    // Exit the admin menu
    if (key == '2'){
      buzzer_tap();
      break;

    // If key is one open menu to allow for code editing
    } else if (key == '1') {
      buzzer_tap();

      // While loop for entering new code to edit
      while (1) {

        // Clear lcd and ask for which code to edit
        lcd.clear();
```

```
        lcd.print("Enter user code");
        lcd.setCursor(0,1);
        lcd.print("number to edit: ");

        // Listen for user input
        char key = keypad.getKey();

        // If the user input is a number
        if (is_num(key)) {
          buzzer_tap();

          // Convert the key pressed into an int
          int edit_code = (int)key - '0';
          Serial.print("edit code: ");
          Serial.print(edit_code);
          Serial.print("\n");

          // New code array and pointer
          char* new_code = new char[4];
          int new_code_pointer = 0;

          // New code repeat and pointer
          char* repeat_code = new char[4];
          int repeat_code_pointer = 0;

          // loop to enter the new code and new code repeat
          while (1) {

            // If both codes have been entered
            if (new_code_pointer == 4 and repeat_code_pointer == 4) {

              // Check to see if both codes are the same
              valid = 1;
              for (int i=0; i<4; i++) {
                if (new_code[i] != repeat_code[i]) {
                  valid = 0;
                }
              }

              // If both codes are the same, save the new code to the pin
              // code array
              if (valid) {
                Serial.print("i: ");
                for(int i=0; i<4; i++) {
                  Serial.print(i);
                  pinCodeArray[edit_code][i] = new_code[i];
                }

                // Clear the memory of the new code and new code repeat arrays
                delete[] new_code;
                delete[] repeat_code;

                // CHECK
                Serial.print("\n");
                for (int i=0; i<10; i++) {
                  Serial.print(i);
                  Serial.print(": ");
                  for (int j=0;j<4; j++) {
                    Serial.print(pinCodeArray[i][j]);
                  }
```

```
                Serial.print("\n");
              }
              Serial.print("Exit");
              return 1;
            }

            // If the code wasn't valid, clear the screen and display message
            lcd.clear();
            lcd.print("Invalid code!");
            delay(1000);
            break;
          }

          lcd.clear();

          // New code entry and hidden user entry
          lcd.print("new code: ");
          for (int i=0; i<new_code_pointer; i++) {
            lcd.print("*");
          }

          // Repeat code entry and hidden user entry
          lcd.setCursor(0,1);
          lcd.print("repeat : ");
          Serial.print(repeat_code_pointer);
          for (int i=0; i<repeat_code_pointer; i++) {
            lcd.print("*");
          }

          // Get user input
          char key = keypad.getKey();

          // If the input is a number
          if (is_num(key)) {
            buzzer_tap();

            // If 4 digits haven't been entered into the new code
            if (new_code_pointer < 4) {

              // Store the user input into the new code and increment the new pointer
              new_code[new_code_pointer] = key;
              new_code_pointer++;

            // If 4 digits haven't been entered into the repeat code
            } else if (repeat_code_pointer < 4) {

              // Store the user input into the repeat code and increment the repeat pointer
              repeat_code[repeat_code_pointer] = key;
              repeat_code_pointer++;
            }
          }
          delay(200);
        }
      }
      delay(200);
    }
  }
  delay(200);
 }
}
```

```
430
431 /*==== Servo Function definition ====*/
432 // Function used to control servo to open the door
433 void open_door() {
434
435   // Attach the servo to the servo pin to allow
436   // for data to be sent to the servo
437   servo.attach(servoPin);
438
439   // Clear the lcd and write to it
440   lcd.clear();
441   lcd.print("Unlocked");
442
443   // Set the servo to the open position
444   servo.write(0);
445
446   // Wait and then detach the servo
447   delay(1000);
448   servo.detach();
449 }
450
451 // Function used to control servo to close the door
452 void close_door() {
453
454   // Attach the servo to the servo pin to allow
455   // for data to be sent to the servo
456   servo.attach(servoPin);
457
458   // Set the servo to the closed postion
459   servo.write(180);
460
461   // Wait and detach the servo
462   delay(1000);
463   servo.detach();
464
465   // Clear the lcd and write to it
466   lcd.clear();
467   lcd.print("Locked");
468   delay(1000);
469 }
470
471 /*==== Buzzer tone function definition ====*/
472 // Function to make the success tone with the buzzer
473 void buzzer_success() {
474   tone(BUZZER, 2500);
475   delay(200);
476   noTone(BUZZER);
477 }
478
479 // Function to make the fail tone with the buzzer
480 void buzzer_fail() {
481   tone(BUZZER, 450);
482   delay(200);
483   noTone(BUZZER);
484 }
485
486 // Function to make the tapping sounds of button pressing
487 // with the buzzer
488 void buzzer_tap() {
489   tone(BUZZER, 2500);
```

```
490   delay(10);
491   noTone(BUZZER);
492 }
```

C, firstline=50, lastline=58, caption=assignment_code.ino]assignment$_c$*ode.ino*

0.4 User Stories

User Story ID	User Story Description
001	The user can enter a code and open the door
002	The user can enter an invalid code and the code will be rejected
003	The user can enter an admin code to edit the other codes
004	The user can enter a non-admin code and the admin menu will not be opened
005	The user will get feedback when using the system

0.5 Test Suites

Test Reference No.	User Story ID	Description of test	Testing Process	Expected outcome	Actual outcome
001	003	The user enters a valid admin code and can edit a code.	• User enters a valid admin code • User enters number of code to edit • User enters new code • User re-enters the new code	The system accepts the new code and will then take the user back to the main "locked" screen.	This unit test was successful. The system accepted the new code entered by the user and then took the user back to the main "locked" screen.
002	004	The user enters an invalid admin code and will be rejected by the system	• User enters an invalid admin code	The system will reject the code and provide feedback to the user.	This unit test was successful. The system rejected the code and provided feedback in the form of flashing the LED and using the piezoelectric speaker.

Test Reference No.	**User Story ID**	**Description of test**	**Testing Process**	**Expected outcome**	**Actual outcome**
003	001	The user enters a valid code to open the door using the servo.	• User performs test 001 to create a valid code • User enters the valid code	When the valid code is entered the servo will open and then close.	This unit test was successful. Test 001 was done successfully. The code created was then accepted by the system when entered by the user. The servo opened and then closed, accompanied by feedback from the green LED.
004	002	The user enters an invalid code and the system will reject the code.	• User performs test 001 to create a valid code • User enters an invalid code	When the user enters the invalid code, the system rejects the code	This unit test was successful. Test 001 was done successfully. The invalid code was entered and the system rejected it and gave feedback with the red lED and piezoelectric speaker.
005	005	The user receives feedback from the system	• The user presses buttons on the keypad, producing a tone from the piezoelectric speaker • The user performs test 001 and creates a valid code • The user enters the valid code, getting feedback • The user enters an invalid code, getting feedback	When the use presses a button, the speaker will beep quickly and the red LED will flash quickly as feedback to the user. When a valid code is entered the green LED will flash and a message will be displayed on the LCD. When an invalid code is entered, the red LED will flash and a message will be displayed.	This unit test was successful. When the user presses a button on the keypad, the system provides feedback with a small beep from the piezoelectric speaker and a small flash from the red LED.

References

- Guyer, S. Helbling, C. (2016) 'Juniper: a functional reactive programming language for the Arduino', *FARM 2016: Proceedings of the International Workshop on Functional Art, Music, Modelling, and Design*, 4, p8-16
- Hernandez, J. (2016) *Arduino's LiquidCrystal Library with SPI*[ONLINE]. Available at: https://playground.arduino.cc/Main/LiquidCrystal/ (Accessed: 16^{th} February 2020)
- lastminuteengineers (no date) *How 74HC595 Shift Register work & Interface it with Arduino*[ONLINE]. Available at: https://lastminuteengineers.com/74hc595-shift-register-arduino-tutorial/ (Accessed: 4^{th} March 2020)