Garrett Barton and Nicholas Harper

Dr. Trahan

EE 3752/003 (Tuesday)

12/01/2017

Semester Project: Automatic Door Opener with Radio Frequency Control

Introduction:

The purpose of this project it to allow students to apply the knowledge and skills learned in this course so far to a broader scope than the course may have covered. This project also allows students to gain experience with design and implementation procedures needed for Senior Design and industry work. This project is relevant to this course in that it deals with coding hardware that includes microcontrollers in order to implement a practical result.

This particular project will be focusing on using 2 factor authentication (2FA) to allow access to an automatic door opener. The two factors are a possession factor and a knowledge factor. The transmitter (clicker) only transmits using certain a frequency (315 MHz) and the receiver will read only that certain frequency (possession factor). By this it is meant that the transmitter and the receiver are paired to work together, but not with other transmitters/receivers. The knowledge factor will be the password itself. The password will consist of a series of letters A, B, C, and D; done with a four button clicker. This will allow up to $4^4 = 256$ passwords which is more than enough for the roommates plus one. The idea is to have up to five persons have access to this door remotely; four roommates and a "guest" account. Indicator LEDs will be used to indicate processing, successes, and/or failures of the attempts, and to aid in the presentation of the project.

Background:

The goal is to have an automatic door open remotely with a radio frequency controller, in the form of a keychain-fob clicker and via a password. The door has a system in place to allow persons in wheelchairs to have the door open automatically using a keypad. There is a manual open/close toggle button on the inside of the apartment. The idea is to allow the clicker holders to have to door open as they are walking up instead of having to swipe a card and input a password.

The door was found to have an issue with the locking, dual-piston solenoid. The pistons have a notch on one side of their tips to help prevent undesired forced openings. This was found to prevent the pistons from retracting when the door frame was pressing on the solenoid housing. This can be circumvented with a simple piece of rubbery cord pressed into one of the notches; which helps to prevent the notch from being fully engaged if the door is pressing on the solenoid housing.

The devices chosen were picked mostly on availability from Adafruit, an electronics distributor that caters to DIY projects. However, some were taken from various parts collections from personal tinkering:

- The transmitter and receiver pair was picked out of the paired sets offered by Adafruit (though due to stocking issues others were found to match).
- The power supply will be coming from a wall outlet.
- Servomotor was ordered to be powerful enough to flip the switch.

In total the project will cost students $31.89 (not including tax, etc.) [$47.69 in total] for the receiver, one transmitter (each of the other roommates that wants one will buy their own), servo, and AC to 5.5 x 2.1 mm DC barrel connector power cord.

Methodology:

Approach:

The system flow is as follows:

1) RF signal sent (characters sent serially to receiver) and received (indicator LED turned blue).
2) Controller checks the string against the saved passwords.
3) If the input matches (indicator LED turned green) one of the passwords then the servo is turns to a position at least 45° counterclockwise from the vertical, neutral position.
4) A wait-timer will allow the door to be held open for a predetermined time and then will turn the servo to a position at least 45° clockwise from the vertical, neutral position.
    a. In the event of another user entering a password, an interrupt signal will keep the wait-timer from signaling the servo to flip the switch to close the door.
    b. There will be a buzzer before the door opens or closes to alert to clear the area.

Edge Case:

If there is an offset entry of character, when a user enters less than the four characters required, the program will reset the attempted entry to allow the user to enter in a new attempt after 2 seconds of not inputting.

Interrupts:

1. This software interrupt will allow a new user to input their password and, provided that the password is correct, the door will stay open. This is for the situation when one user has already opened the door and a new user, not knowing about the first user, enters their password to open the door. (Functions with “wait” concatenated on the end)
2. There is a software interrupt that will clear the previous inputted characters if there is not an input attempt within two seconds of the last input. (In ReadInputs)

Functions:

The code was done in a style that broke processes down into functions, with main processes being functions calling subsequent functions. This was done to declutter the code and make the code easy to read, and debug. The control flow map will be attached at the end of this report due to its size. A list of functions is as follows:

- set_up
  - This is used to configure all the pins used, and set the baud rate used in serial communication for displaying to the Serial Window (like a UART).
- loop
  - This function acts as the main function, and will loop continuously.
- concatInputs
  - This takes the returned character from ReadInputs and concatenates them together to form a four character long string. This function will loop until the proper length is reached.
- ReadInputs
  - This is where the inputs are decoded from the pins. There is an infinite loop to continuously check for a pin to receive a high (+5V) signal, which is interpreted as that being the character being attempted to be inputted. There is a software interrupt that will clear the previous inputted characters if there is not an input attempt within two seconds of the last input. There is a half second wait for each iteration to ensure only one copy of the incoming signal is decoded, instead of a stream.
- array_checks
  - This function compares the string of inputted characters with each element of the array of saved passwords by calling pwcheck over each element. If the passwords match then the input string is cleared for a new input string.
- pwcheck
  - This function preforms a string comparison on the input string and the element of the saved passwords array passed to it from array_checks. If the passwords match then the indicator LED in turned green, OpenDoor is called, CloseDoor is called, and the indicator LED is turned off. Else, the indictor LED is turned red, and then off after a second.
- concatInputsWait
  - Does the same as concatInputs, but will break form the loop if the input string is cleared (only done if cleared in ReadInputsWait, see that function for details).
- ReadInputsWait
  - Does the same as ReadInputs, but instead of clearing the previous inputs after two seconds it will wait for the time chosen for an entry and then if there are no inputs, it will break from the input-searching loop.
- indicatorLED
  - Uses input to easily define what color the indicator LED should be.
- setColor

- Called by indicatorLED to write the corresponding voltages to the correct pins to achieve the desired color.

- OpenDoor
  - Calls Sound_Alarm(Open), signals the servo to turn to flip the switch the open position, and then have a delay to allow the servo time to move to the desired position.
- CloseDoor
  - Calls Sound_Alarm(Close), signals the servo to turn to flip the switch the close position, a delay to allow the servo time to move to the desired position, and then signal the servo to move to the neutral position, with a delay for the servo to move.
- Sound_Alarm
  - The alarms for opening and closing the door are the same except for the opening sequence having a lower frequency buzz so that the high pitch squeals of the alarm do not scary someone sitting in the living room when it is quite.
- Hardware_testing
  - This function is "wired" to the push button on the breadboard and when pressed will: cycle the indicator LED colors through green, red, and then blue; calls OpenDoor, then CloseDoor; and then sounds an alarm.

Physical Implementation:

There is already a hardware for the actual door opener (see Image 1 below), and the hardware for this project will sit on top and be attached to this hardware. This will add no obstruction to the regular function of the door, and minimal obstruction to the switch for the door opener. The servo will be attached to a metal frame that will be attached to the power supply box (brown box in the center of Image 1) with command strips. This will put the servo close to the switch, but far enough away to allow the servo to turn past the switch after the switch has been flipped. The Arduino and the breadboard will be set on top of the actuator box (silver box on the left hand side of Image 1).

Image 1

Testing:

In the testing of this project, the effective range of the transmitter/receiver was found to be 15-20 meters in distance from each other. The signal has to travel through a wall, a metal window screen, and an area that has several Wi-Fi routers nearby. The angles for the servo were experimentally found to be the full range of the servo to ensure the switch is flipped.

When decoding inputs it was found, from displaying to a serial window, that the input was being taken as a stream of a character instead of a single character as was desired. This was mitigated by adding a half second delay to each iteration. This allows the user to hold the button on the transmitter for a solid signal transmission, and not have the input be a string of just one character. This delay can of course be tuned to any sensitivity desired. However, for testing a half second allowed tester to hold the transmission long enough to see that and LED lite up showing that the signal was received.

The most time consuming and tedious testing was of the first interrupt, holding open for overlapping code entries. This is due to the Arduino microcontroller not having multiple cores. This means practically that all processes have to be serial in execution. This serial only format makes having an interrupt that is supposed interrupt a function that runs through a waiting procedure. The manner in which the waiting function is setup allows there to be a nearly parallel check for inputs and waiting. The waiting was designed to be a side effect of the checking procedure that was specifically made to act as the waiting function. Meaning that the waiting is actually controlled by the number of iterations (controlled by the conditions of checking for inputs) of the checking for inputs function. Another side effect occurs when the program is backing out of the array_checks. It was found that when multiple iterations are entered, then the door closing procedure will be called for each iteration exited. This side effect is benign and is almost unnoticeable in application, and therefore there was no attempt made to counter act it.

Conclusion:

The design process began with the decision to use radio frequency (RF) remotes in order to trigger the door to open. When attempting to order the parts, however, the primary supplier was out of stock and would not have more until the last week available to work on the project. The parts are identical, but were ordered from different suppliers.

The original design was to use a solenoid to push the switch, but was changed to using a servo motor. The switch to a servomotor was decided due to the complications with dealing with solenoids (back-EMF, induced currents, etc.). The students were only just learning about back EMF, etc. and the students did not yet feel comfortable designing a safe system with these considerations or the power requirements of such a system. Servo motors were also a topic of discussion in one student's robotics class, and had good examples to aid in the coding for this project.

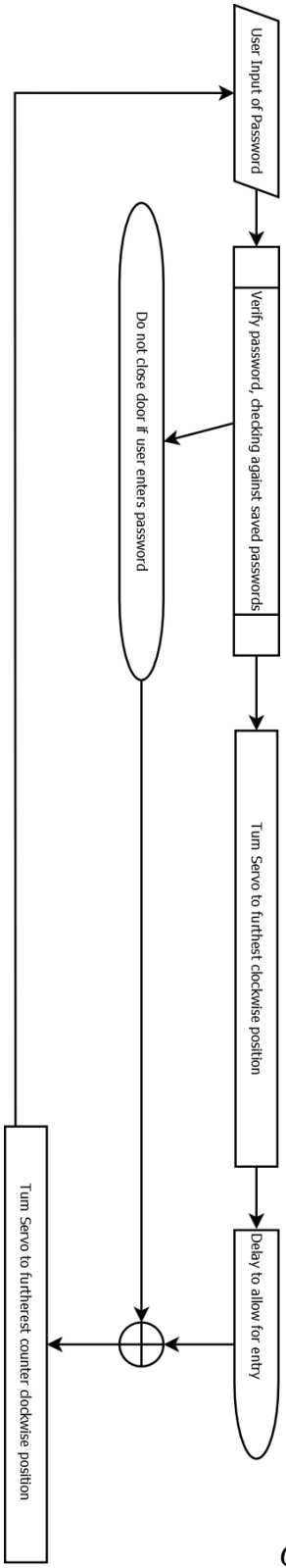
The final product preforms almost as desired when the project was originally conceived. The only short fall is the effective range of the signal. After research, the remote does not seem to

be the problem. The problem seems to be that the receiver is having trouble picking up to signal sent form the transmitter. Suggestions from professionals include a directional antenna and re-measuring/cutting the antenna provided. However, that seems too much to put in, to possibly improving this small project. The range of the process was found to be about 15-20 meters from the receiver, measured radially.

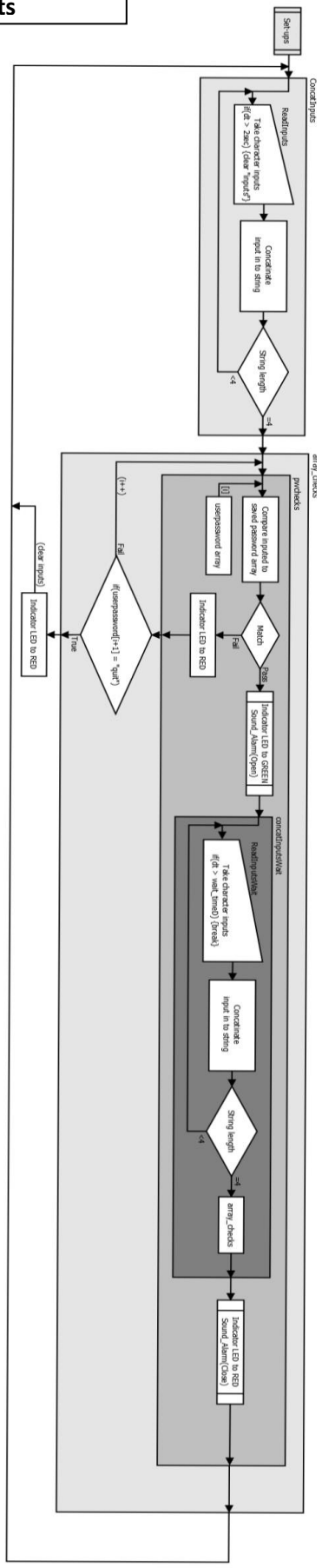
For future improvements on this project:

1) Steps might be added to counteract the side effect from backing out of multiple array_checks loops, such that the CloseDoor function is only called once for one door open.
2) The codes may be changed for ease of access (e.g. hold down one button). Having one button to hold down rather than a four character password, would help to compensate for the poor signal transmission.
3) The time that the door is held open will be optimized with user testing over the next few weeks.
4) Creating a check that will close the door if a certain letter is received while the door is open. This would be to eliminate the door hanging open until the wait function finished, or someone manually flipped the switch from the inside.
5) Improve the mounting system for the servo. Possibly add a support arm closer to the servo to reduce torque put on the frame's attachment to the power supply box.

Control Flow Charts



Overview



Full Detail

Code

The code has been copy and pasted here to ensure detailed inspection was possible for the instructor.

(Click drop-down arrow to view):

```
//Author(s): Garrett Barton and Nicholas Harper
//Course: EE 3752/sec:003 (Tuesday 1030-1230)
//Professor: Dr. J. Trahan
//Title: Semester Lab Project
/*Overview:
  ++This code will take the input from the RF receiver
    (while turning the indicator LED BLUE) and check
    that input against the saved passwords.
  ++If the code matches a password then the servo will
    flip the door switch, turn the indicator LED GREEN,
    and the buzzer will sound.
  ++Else it will flash the indicator LED RED.
  ++The wait timer will hold the door to allow entry and
    then the buzzer will sound and the servo will close
    the door

DISCLOSURE: Code for controlling hardware (LED, servo,
        etc.) will be based on examples from other
        courses, books, and other reference material.
        All logic and control flow is the design/work
        of the author(s).
Edge Case(s):
  1) Offset entry of character
       This is for when a user enters less than the
      four characters required. The program will reset
      the attempted entry to allow the user to enter in
      a new attempt after 2 seconds of not inputting.

 Interrupt(s):
  1) Hold door (software)
       There might be a time when one user opens the
      door with the clicker, and then a second user
      attempts to have the door open (unaware that
      the first user already has the door open.  This
      interrupt allows the second user to input their
      password and have the door stay open. There is
      a byproduct of this interrupt that allows the
      door to be held open for longer than the standard
      time.  This system is also recursive, allowing
      multiple entries (different attempts) while the
      door is open.
  2) "Inputs" clear (used to handle edge case #1)

 The code is in the order of flow, with some auxiliary function at the end
*/
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////// LIBRARIES ////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
 #include <Servo.h>       //functions for controlling servo (instead of using PWM)
 Servo myServo;             //names the servo for use of the servo library
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
```

```
////////////////////////////////////////////////////////// VARIABLE DEFINITIONS ////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
 int angle_open = 35;         //angle used to open door
 int angle_close = 179;       //angle used to close door
 int wait_timeS = 1000;       //allows time for servo to move
 int wait_timeD = 10000;      //allows 1/6 minute for entry

 int buzzer_pin = 3;
 int buzzer_freq1 = 100;        //in hertz [31:65535]
 int buzzer_freq2 = 500;        //in hertz [31:65535]
 int buzzer_freq3 = 1000;       //in hertz [31:65535]
 int buzzer_time = 500;       //in milliseconds

 const int redLEDpin = 11;       //pin that RED is connected to
 const int greenLEDpin = 6;      //pin that GREEN is connected to
 const int blueLEDpin = 10;     //pin that BLUE is connected to

 char R,G,B;                     //set up characters used in function indicatorLED

 int pinA = 12;                  //sets pins for decoding inputs
 int pinB = 8;
 int pinC = 7;
 int pinD = 4;

 String inputs, x;               //used for concatInputs function

 int duration;                   //used for Sound_Alarm function
 String which;

 int Htest = 0;                  //variable that controls if testing is done
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////// SET-UPS ////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
 void setup()
{
 myServo.attach(9);                      //attaches the servo on pin 9 to the servo object
 Serial.begin(9600);                     //baud rate == 9600 bits/sec

 pinMode(greenLEDpin,OUTPUT);            //sets these variables as outputs for digital pins
 pinMode(redLEDpin,OUTPUT);              //
 pinMode(blueLEDpin,OUTPUT);             //
 pinMode(2, INPUT);                      //input for Hardware_test button
 pinMode(pinA, INPUT);                   //inputs from Rx
 pinMode(pinB, INPUT);                   //
 pinMode(pinC, INPUT);                   //
 pinMode(pinD, INPUT);                   //
}
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////// PASSWORDS ////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//                  Garrett  Nick   Tyler   Patrick
 String userpasswords[] = {"ABCD", "DBCA", "BADC", "DCBA", "quit"};
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////// MAIN FUNCTION ////////////////////////////////  MAIN  //////////////
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
 void loop()
{
 Hardware_testing();                     //press button on board to test hardware

 Serial.print("\n/////////////////////////////////////////////////");
 Serial.print("\nInput password: ");
```

```
 x = concatInputs();                //ReadInputs -> concatenates inputs -> returns entry as string (HERE FOR DISPLAY ORDER ONLY)
 Serial.print("\nInputs: ");
 Serial.print(x);
 array_checks();
}
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////// CONCATINATE INPUTS ////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
String concatInputs()
 {
  int j = 0;                                //initializes incriminator
//   for(j=0; j<=3; j++)                    //iterates 4 times to get 4 character for password entry attempt
  int pwattemptL = inputs.length();
  while(pwattemptL < 4)
  {
   x = ReadInputs();                        //gets input character by character
   inputs = String(inputs + x);             //concatenates the current input with the previous inputs
   pwattemptL = inputs.length();
  }
  return inputs;                            //returns the string that is the inputted 4-character password attempt
 }
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////// READ INPUTS ///////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
String ReadInputs()
{
 String rc;
 char a,b,c,d;
 int time1 = millis();
 while(1)
 {
  a = digitalRead(pinA);                    //reads value from pin assigned to A
  if(a == HIGH)  {rc = "A"; indicatorLED('B'); break;}      //if the pin assigned to A is high -> store A in rc (received character)

  b = digitalRead(pinB);                    //reads value from pin assigned to B
  if(b == HIGH)  {rc = "B"; indicatorLED('B'); break;}      //if the pin assigned to B is high -> store B in rc (received character)

  c = digitalRead(pinC);                    //reads value from pin assigned to C
  if(c == HIGH)  {rc = "C"; indicatorLED('B'); break;}      //if the pin assigned to C is high -> store C in rc (received character)

  d = digitalRead(pinD);                    //reads value from pin assigned to D
  if(d == HIGH)  {rc = "D"; indicatorLED('B'); break;}      //if the pin assigned to D is high -> store D in rc (received character)

  Htest = digitalRead(2);                   //always allow for hardware testing
  if(Htest == 1)  {Hardware_testing();}

  int time2 = millis();
  if((time2-time1) > 2000)                  //if no other input for 2 seconds, reset
   {
    inputs = "";
    indicatorLED('R');                      //turn the indicator LED RED
    delay(1000);
    time1 = millis();                       //resets time1 for Exception 1
    indicatorLED('F');                      //turn the indicator LED OFF
   }
 }
 delay(500);                                //delay added to prevent multiple values from being read too quickly
 return rc;                                 //   for the user to let off the button
}
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////// ARRAY_CHECK ///////////////////////////////////////////////////////
```

```
/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
 void array_checks()
 {
   String quit = "";                                          //initializes password array terminator variable
   int j = 0;                                                 //initializes array indexer

                                                              //ReadInputs -> concatenates inputs -> returns entry as string into x
   while(quit != "quit")                                      //loop through saved password array until hit terminator string
    {                                                          //  (allows for easily scalable password lists)
     quit = userpasswords[j+1];                               //gets next element in saved password array(for test for next loop)
     char z = pwcheck(x, userpasswords[j]);                   //checks password with whole array
     if(z == 'P')  {break;}                                   //exits while loop if find matching, saved password
     j++;                                                      //increments through saved password array
    }
   inputs = "";                                               //reset the string holding the inputs for the next iteration
 }
/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////// PASSWORD CHECKS  ///////////////////////////////////////////////////
/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
 char pwcheck(String input, String saved)
 {
    if(input == saved)
     {
      inputs = "";
      x = "";
      Serial.print("\t\tOpen!");
      indicatorLED('G');                                      //if input matches password indicate success
      OpenDoor();                                             //then open the door
//^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
^^
      x = concatInputsWait();
      array_checks();
//^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
^^
      CloseDoor();                                            //then close the door
      indicatorLED('F');                                      //turn off the indicator LED
      return 'P';
     }
    else
     {
      Serial.print("\t\tNope!");
      indicatorLED('R');                                       //if input NOT matches password indicate failure
      delay(1000);                                            //wait long enough to show
      indicatorLED('F');                                      //turn off the indicator LED
      return 'F';
     }
 }
/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////// CONCATINATE INPUTS_WAIT /////////////////////////////////////////////
/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
 String concatInputsWait()                                     //if input character just before time limit then the door could stay
 {                                                             //  open for about 4(T-1) seconds, the standard being T (wait_timeD)
   int j = 0;                                                  //initializes incriminator
   int pwattemptL = inputs.length();                          //gets length of the attempt
   while(pwattemptL < 4)                                      //iterates 4 times to get 4 character for password entry attempt
   {
    x = ReadInputsWait();                                     //gets input character by character
    inputs = String(inputs + x);                              //concatenates the current input with the previous inputs
    pwattemptL = inputs.length();

    if(x == "")  {break;}                                     //if no inputs for 2 seconds break
```

```
   }
   return inputs;                                  //returns the string that is the inputted 4-character password attempt
  }
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////// READ INPUTS WAIT ////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
 String ReadInputsWait()
{
  String rc;
  char a,b,c,d;
  int time1 = millis();
  while(1)
  {
    a = digitalRead(pinA);                         //reads value from pin assigned to A
    if(a == HIGH)  {rc = "A"; indicatorLED('B'); break;}     //if the pin assigned to A is high -> store A in rc (received character)

    b = digitalRead(pinB);                         //reads value from pin assigned to B
    if(b == HIGH)  {rc = "B"; indicatorLED('B'); break;}     //if the pin assigned to B is high -> store B in rc (received character)

    c = digitalRead(pinC);                         //reads value from pin assigned to C
    if(c == HIGH)  {rc = "C"; indicatorLED('B'); break;}     //if the pin assigned to C is high -> store C in rc (received character)

    d = digitalRead(pinD);                         //reads value from pin assigned to D
    if(d == HIGH)  {rc = "D"; indicatorLED('B'); break;}     //if the pin assigned to D is high -> store D in rc (received character)

    Htest = digitalRead(2);                        //always allow for hardware testing
    if(Htest == 1)  {Hardware_testing();}

    int time2 = millis();
    if((time2-time1) > wait_timeD)                 //if no other input for wait_timeD seconds, move out of function
     {
       break;
     }
  }
  delay(500);                                      //delay added to prevent multiple values from being read too quickly
  return rc;                                       //   for the user to let off the button
}
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////// indicatorLED ////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
 void indicatorLED(char color)
 {
    if(color == 'R')                               //indicatorLED to RED
     {
       setColor(255,0,0);                          //R...G...B
     }
    if(color == 'B')                               //indicatorLED to BLUE
     {
       setColor(0,0,255);                          //R...G...B
     }
    if(color == 'G')                               //indicatorLED to GREEN
     {
       setColor(0,255,0);                          //R...G...B
     }
    if(color == 'F')                               //indicatorLED to OFF
     {
       setColor(0,0,0);                            //R...G...B
     }
 }
////////////////////////////////////////////////////////////////////////
 void setColor(int redVal, int greenVal, int blueVal)
```

```
 {
   analogWrite(redLEDpin, redVal);
   analogWrite(greenLEDpin, greenVal);
   analogWrite(blueLEDpin, blueVal);
 }
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////// OPEN DOOR  ////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
 void OpenDoor()
 {
   Sound_Alarm(1,"open");                    //int duration (number of cycles), String which ("open" or "close")
   myServo.write(angle_open);                //turns servo to specified angle
   delay(wait_timeS);                        //waits for servo to move to angle
 }
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////// CLOSE DOOR  ///////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
 void CloseDoor()
  {
   Sound_Alarm(1,"close");                   //int duration (number of cycles), String which ("open" or "close")
   myServo.write(angle_close);               //turns servo to specified angle
   delay(wait_timeS);                        //waits for servo to move to angle
   myServo.write(85);                        //turns servo to center
   delay(wait_timeS);                        //waits for servo to move to angle
  }
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////// SOUND_ALARM ///////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
 void Sound_Alarm(int duration, String which)
 {
  if(which == "open")
   {
    tone(buzzer_pin, buzzer_freq1, buzzer_time);       //pin, frequency, duration in ms
    delay(750);
    tone(buzzer_pin, buzzer_freq1, buzzer_time);       //pin, frequency, duration in ms
    delay(750);

    for (int i = 0; i < duration; i++)                 //alarm for before opening door
     {
      tone(buzzer_pin, buzzer_freq2, buzzer_time);     //pin, frequency, duration in ms
      delay(200);

      tone(buzzer_pin, buzzer_freq3, buzzer_time);     //pin, frequency, duration in ms
      delay(200);

      tone(buzzer_pin, buzzer_freq2, buzzer_time);     //pin, frequency, duration in ms
      delay(200);

      tone(buzzer_pin, buzzer_freq3, buzzer_time);     //pin, frequency, duration in ms
      delay(200);

      i++;
     }
   }
  if(which == "close")
   {
    for (int i = 0; i < duration; i++)                 //alarm for before closing door
     {
      tone(buzzer_pin, buzzer_freq2, buzzer_time);     //pin, frequency, duration in ms
      delay(200);
```

```
        tone(buzzer_pin, buzzer_freq3, buzzer_time);     //pin, frequency, duration in ms
        delay(200);

        tone(buzzer_pin, buzzer_freq2, buzzer_time);     //pin, frequency, duration in ms
        delay(200);

        tone(buzzer_pin, buzzer_freq3, buzzer_time);     //pin, frequency, duration in ms
        delay(200);

        i++;
       }
    }
 }
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////// Hardware_TESTING  ////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//cycles through colors on multiLED, turns servo, and sounds buzzer at end of cycle whenever button pressed
 void Hardware_testing()
 {
   Htest = digitalRead(2);
   if(Htest == HIGH)                                 //button in pressed
   {
    indicatorLED('G');                               //tests multicolored LED
    delay(1000);

    indicatorLED('R');
    delay(1000);

    indicatorLED('B');
    delay(1000);

    tone(buzzer_pin, buzzer_freq1, buzzer_time);     //pin, frequency, duration in ms
    delay(1000);

    myServo.write(85);                               //turns servo to center
    delay(wait_timeS);                               //waits for servo to move to angle
    OpenDoor();                                      //Test servo
    delay(2000);                                     //
    CloseDoor();                                     //
    myServo.write(85);                               //turns servo to center
    delay(wait_timeS);                               //waits for servo to move to angle

   for (int i = 0; i < 6; i++)                       // prototyping for alarms
   {
    tone(buzzer_pin, buzzer_freq2, buzzer_time);     //pin, frequency, duration in ms
    delay(200);

    tone(buzzer_pin, buzzer_freq3, buzzer_time);     //pin, frequency, duration in ms
    delay(200);

    tone(buzzer_pin, buzzer_freq2, buzzer_time);     //pin, frequency, duration in ms
    delay(200);

    tone(buzzer_pin, buzzer_freq3, buzzer_time);     //pin, frequency, duration in ms
    delay(200);

    i++;
   }
    indicatorLED('F');                               //turn off the indicator LED
   }
 }
```