ECE 2242 Project Design Notebook

Daniel Xie

March 19, 2021

Contents

1 Problem

A new shed and he wants it to be ”smart”. Part of this overall project, the door lock, will be scope of this project.

1.1 Background

My dad wants to design and build a new shed this spring/summer since he can’t find his desired style, build quality and price point at places like Home Depot and Lowe’s. He wants to make it a ”smart shed”, so that accessing and working in the shed is easier and more convenient. One aspect he wants to make smart is the door lock instead of fumbling around with a key-chain when he’s trying to unlock his shed to get some tools in the winter or hoping that the neighbour wouldn’t make a copy of the key to the shed when he let’s them use it to borrow some tools. On the other hand, smart locks are not cheap. They can cost anywhere between $150 to $400 depending on the quality of the build and features. Fig. 1 below is the result of a quick search for such locks.

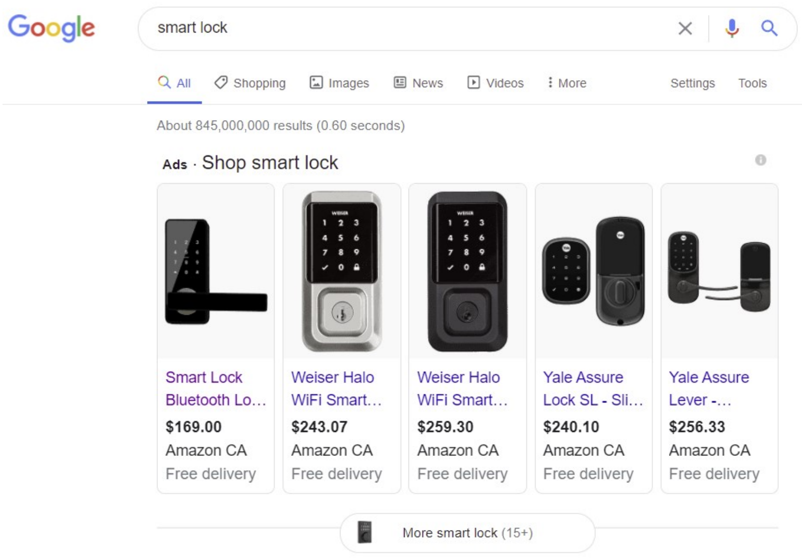

Google
smart lock
All
Shopping
Images
News
Videos
More
Settings
Tools
About 845,000,000 results (0.60 seconds)
Ads · Shop smart lock
Smart Lock Bluetooth Lo… $169.00 Amazon CA Free delivery
Weiser Halo WiFi Smart… $243.07 Amazon CA Free delivery
Weiser Halo WiFi Smart… $259.30 Amazon CA Free delivery
Yale Assure Lock SL - Sli… $240.10 Amazon CA Free delivery
Yale Assure Lever -… $256.33 Amazon CA Free delivery
More smart lock (15+)

Figure 1: Too Expensive!!

Moreover, some of the locks have design oversights that make them easy to defeat, such as the LockPickingLawyer’s demonstration where he was able to unlock a lock with a magnet[1]. With this in mind, my dad desires a smart lock with strong security.

I have an old door lock designed for outdoor use that can be purposed for this project. I simply need to turn the shaft to lock and unlock.

2 Design Parameters

2.1 Primary Criteria

- There must be at least one secure keyless authentication option (excluding a number pad).
- **Display Status.** The lock must display whether the lock is unlocked/locked. It also must display whether authentication is successful.
- **A keypad.** This is in addition to the keyless authentication option.
- **Phone Control.** The convenience of changing the settings of the lock using his phone via RFID/Bluetooth/WiFi would be very convenient. It should also allow the option to access things such as logs.

2.2 Secondary Criteria

- **1-time/temporary passwords.** Such a password that expires after a single use/temporary use (i.e. a neighbour needs to borrow/return a tool from the shed) would be convenient.
- **Log of entries and exits.** My Dad would like to keep track of any entries and exits into his shed.
- **Night Mode.** The lock should light up in the dark when motion is detected near the door to better see it.
- **User Notification.** The lock should notify the user when it's locked/unlocked, etc.

2.3 Design Constraints

- **Break-in Resistant.** The lock should not be susceptible to attacks such as using magnets or spoofs.
- **Within budget.** Total part component cost of the final lock design should not be more than CAD $50.
- **Use available components from list.** Unfortunately, components like the ESP32 microcontroller, cannot be used within the scope of this course.
- **Efficient Power Usage.** The lock should use power as efficiently as possible.
- As compact of a design as possible.
- Must meet project complexity requirement.

3 Possible Solutions

The main focus of this sectio will be to decide which keyless authentication entry solution (excluding the keypad) is the most optimal given above design criteria and constraints.

3.1 Facial Recognition

This solution would use a machine learning model to analyze a face and match it with the database as authentication. This option is greatly to use convenient as your face is the lock. There plenty of options to use to train a neural network for such a complex task to the point that trained models can operate on micro-controllers with no problems. A major problem with this method would be the cost and ability to get the components necessary for this solution. The Arduino Uno R3 is the main limiting factor, as while it is certainly possible for it to run image processing, there will be a significant performance hit. Typical micro-controllers that were used tend to be ESP32s, which have dual-core CPUs with much better clock speeds, making them better suited for such application. Furthermore, the camera module will be needed, which might not get approved by the TAs. Another problem is adding additional users will be difficult. A final problem with this solution is that it is vulnerable to spoofing attacks such as using photos of an authorized user although this would require the attacker to specifically target a user.

3.2 RFID

RFID technology can be a secure way of authenticating a user when an encryption scheme is used. This method is also extremely convenient to use for users. There are a few problems with this method, however. Furthermore, the scanner is already part of the given, which is plus in terms of cost. While many phones today have NFC technology, a subset of RFID, there is little to no official software support for neither Android nor iOS devices. It's not impossible but it will still be difficult, which makes the user experience on a phone more difficult. It also makes the possibility of creating temporary passwords extremely difficult as you would either need to rely on someone to download a potentially janky app or create a clone of the card. Another problem is that you have to touch the card near the scanner, which makes this solution extremely range limited.

3.3 Bluetooth/BLE

A Bluetooth module/BLE could allow for the ubiquitous use of phones as keys to authenticate an user. Furthermore an encrypted key could be generated and shared as a temporary password for other users with their smartphones. Moreover, Bluetooth tends to use less power than WiFi, another possible solution. IT also gives the user flexibility to control some of the lock settings such as adding temporary passwords, accesss logs, receiving alerts as long as they are within range and more. A limitation of this method is that it can only be part of a personal area network, which is a double-edged sword - on one hand it does not expose the lock to the wider internet as is the case with WiFi, on the other hand the user would not be able to receive any notifications if he/she is out of range if something were to happen.

3.4 WiFi

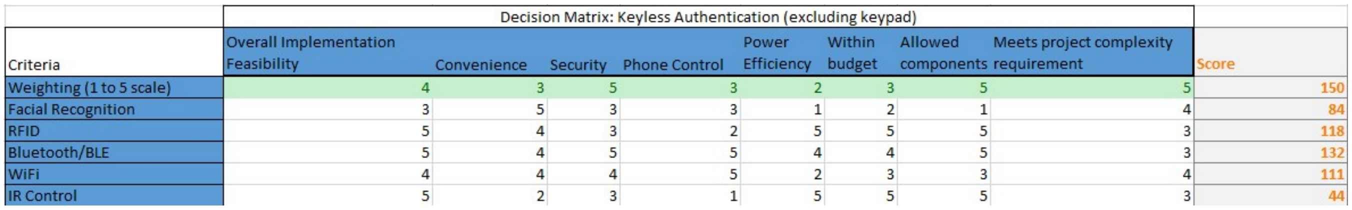
A WiFi module would have the same advantages as Bluetooth in terms of capabilities, but perhaps a bit overpowered. Another upside is that the lock would have the capability of being controlled outside the range of the device, which would give the user a greater degree of control of the device. On the other hand, this greater integration also means that the lock is exposed to the internet, which could make it less secure. Furthermore, WiFi is typically consumes more power than Bluetooth, which could make it less desirable in maximizing power efficiency.

3.5 IR Control

Using an IR sensor and remote control is a cheap and easy way to remote control an Arduino. However, there are many problems with using this particular method. The IR remote would be extremely range limited considering how it is powered with a little CR2025 cell. You would also have to point the remote towards the extremely small sensor, which is much less ideal than a solution like Bluetooth which could care less about the orientation of the key. The ability to create temporary passwords would be extremely difficult as the key would be an IR remote which is somewhat difficult to clone, much less create temporary passwords.

3.6 Overall Assessment

Using a decision matrix, I found that Bluetooth was unsurprisingly the best option overall.



	Decision Matrix: Keyless Authentication (excluding keypad)								
Criteria	Overall Implementation Feasibility	Convenience	Security	Phone Control	Power Efficiency	Within budget	Allowed components	Meets project complexity requirement	Score
Weighting (1 to 5 scale)	4	3	5	3	2	3	5	5	150
Facial Recognition	3	5	3	3	1	2	1	4	84
RFID	5	4	3	2	5	5	5	3	118
Bluetooth/BLE	5	4	5	5	4	4	5	3	132
WiFi	4	4	4	5	2	3	3	4	111
IR Control	5	2	3	1	5	5	5	3	44

Figure 2: Decision Matrix

3.7 Other General Design Considerations

- This project will avoid using relays in favour of MOSFETs to be more resistant to electromagnetic interference.
- 'Night Mode' can be implemented using photo-resistors and PIR sensors to detect motion at night.
- The 28BYJ-48 step motor will likely be used to provide the necessary torque to turn the shaft of the lock.
- A 1602A LCD display will be used as part of the lock to show the user output.
- To indicate the lock is unlocked/lock, in addition to a message displaying on the screen, an active buzzer will be used to inform the user.
- The keypad will be using the 4x4 matrix of buttons.

4 Diagrams

4.1 Block Diagram

Here's a block diagram showing how the layout of the entire project will look like

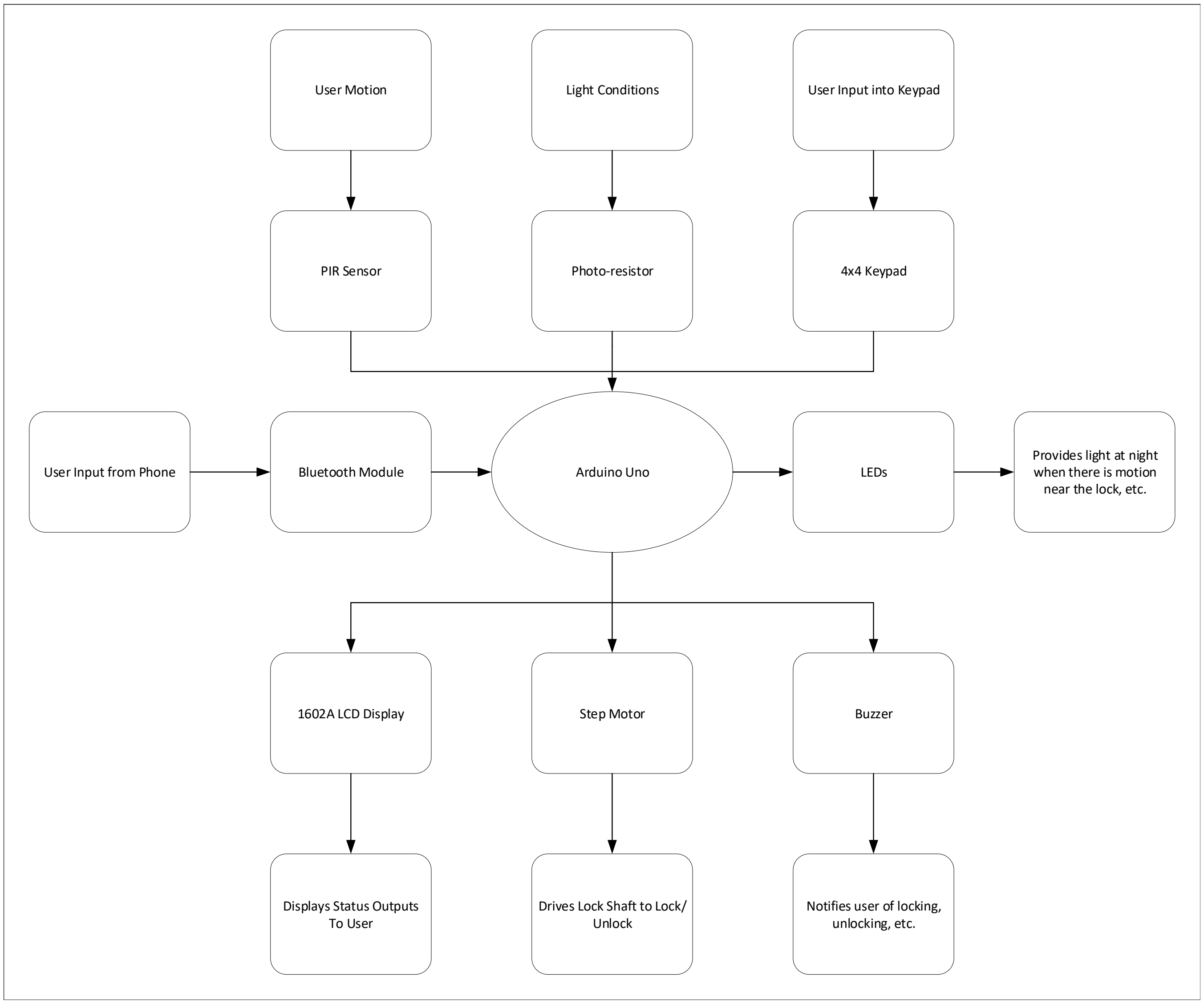

User Motion
Light Conditions
User Input into Keypad
PIR Sensor
Photo-resistor
4x4 Keypad
User Input from Phone
Bluetooth Module
Arduino Uno
LEDs
Provides light at night when there is motion near the lock, etc.
1602A LCD Display
Step Motor
Buzzer
Displays Status Outputs To User
Drives Lock Shaft to Lock/ Unlock
Notifies user of locking, unlocking, etc.

Figure 3: Block Diagram

4.2 Flowchart of Prototype Algorithm

The following flowchart diagram explains the general algorithm of how the system will function.

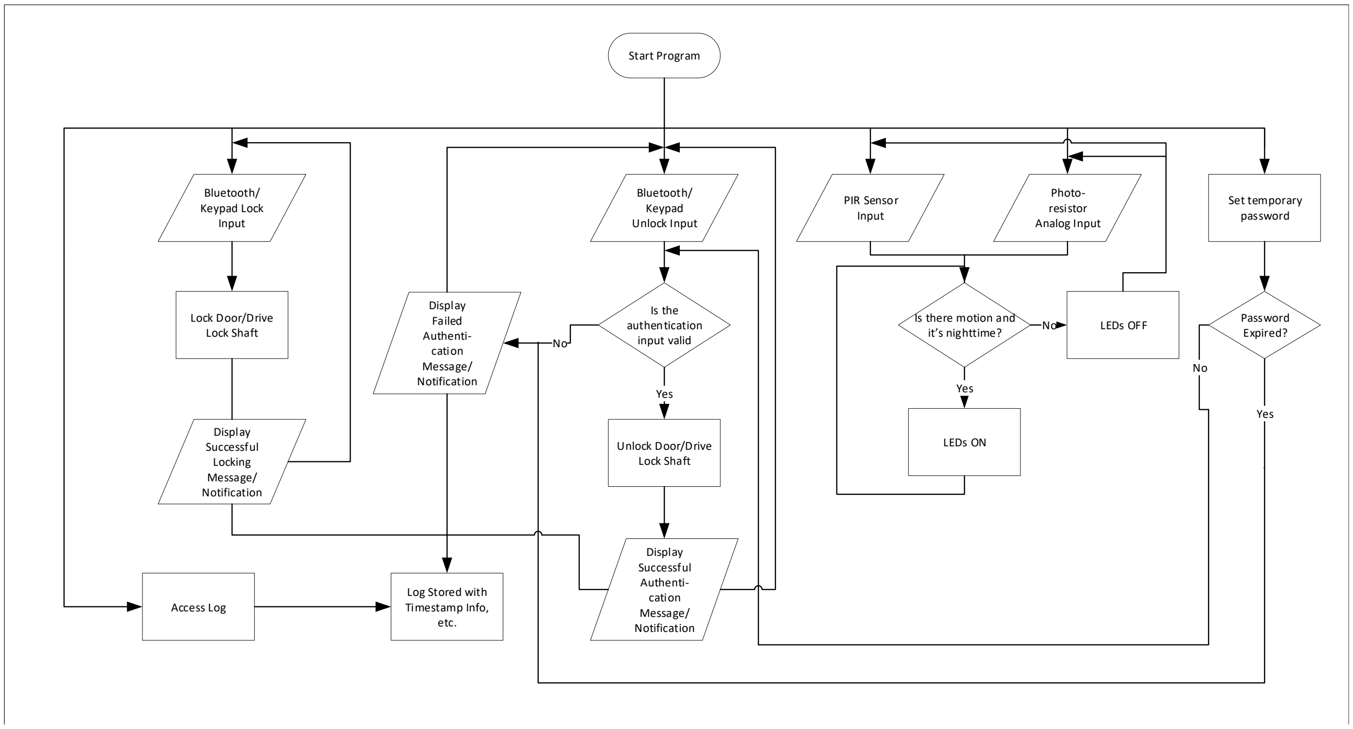

Start Program
Bluetooth/ Keypad Lock Input
Lock Door/Drive Lock Shaft
Display Successful Locking Message/ Notification
Display Failed Authentication Message/ Notification
Bluetooth/ Keypad Unlock Input
Is the authentication input valid
No
Yes
Unlock Door/Drive Lock Shaft
Display Successful Authentication Message/ Notification
PIR Sensor Input
Photo-resistor Analog Input
Is there motion and it's nighttime?
No
Yes
LEDs OFF
LEDs ON
Set temporary password
Password Expired?
No
Yes
Access Log
Log Stored with Timestamp Info, etc.

Figure 4: Flowchart Diagram

5 Component List, Availability and Cost

Table 1: Component List

Component Name	Description	Type	Cost	Link
Arduino Uno R3	Micro-controller included in kit	Micro-controller	N/A	N/A
HM-10 Bluetooth Module	Bluetooth/BLE 4.0 module	Sensor	$14.99	Amazon link
28BYJ-48 Step Motor	Step motor to drive lock shaft. Driver board is included	Actuator	N/A	N/A
Tactile Keypad 4X4 Matrix	4x4 matrix of buttons that will be used as the keypad	Sensor	N/A	N/A
PIR Sensor	Motion detection for 'night mode	Sensor	$3.30	Amazon link
Photo-resistor	Changes resistance based on amount of light	Sensor	N/A	N/A
1602A LCD Screen	16x2 character LCD screen, header pins already pre-soldered	Actuator	N/A	N/A
Active/Piezo Buzzer	Creates noise	Actuator	N/A	N/A
Miscellaneous LED's	light things ups	Actuator	N/A	N/A
Miscellaneous Resistors	Can serve a variety purposes such as a voltage divider	Other	N/A	N/A
SN74HC595 Shift Register	3 to 8 shift register	Other	N/A	N/A
MBv2 Breadboard Power Supply Module Board	Power Supply Module, takes 6.5-12Vin, Vout 3.3V, 5V	Other	N/A	N/A
2N7000 transistor	NMOS transistor	Other	N/A	N/A
Jumper Wires	Connects things together (M-M, F-M, F-F)	Other	N/A	N/A

6 Meeting Notes

6.1 March 4, 2021

- Add button to detect if the door is closed. Possibly reed switches.
- Add low-power mode to algorithm.

7 Prototyping

7.1 TinkerCAD Simulations Notes

TinkerCAD circuit setup is shown below. In this circuit simulation, the pushbutton is the proxy for a door switch sensor and four LEDs are proxies for the inputs of the stepper motor. For the stepper motor, the LED's light from left-to-right and right-to-left to indicate locking and unlocking. There is no equivalent sensor for the HM-10 Bluetooth module, but the code seen above is there. HM-10 uses pins 0 and 1 and has additional pins connected to 5V and GND. Stepper motor power pins are connected to 5V and GND. The next iteration of design will look at using shift registers to reduce the number of ports needed by each component.

7.1.1 Input Block

Below is the input block circuit diagram.
Code for the circuit is -

```
#define LDR A0 //Photoresistor pin
#define PIRdist A1 //PIR pin
#define NCSW A2 //reed switch pin (pushbutton is proxy to simulate.
#define RX_PIN 0 //RX pin for HM10 bluetooth module
#define TX_PIN 1 //TX pin for HM10 bluetooth module
#include <Keypad.h>
#include <SoftwareSerial.h>

SoftwareSerial HM10(RX_PIN, TX_PIN);
const byte ROWS = 4;
const byte COLS = 4;
const int threshold = 300;
int pir_state;
int ph_val;
int oc_state;
char appData;
String inData = "";

char hexaKeys[ROWS][COLS] = {
  {'1', '2', '3', 'A'},
  {'4', '5', '6', 'B'},
  {'7', '8', '9', 'C'},
  {'*', '0', '#', 'D'}
```

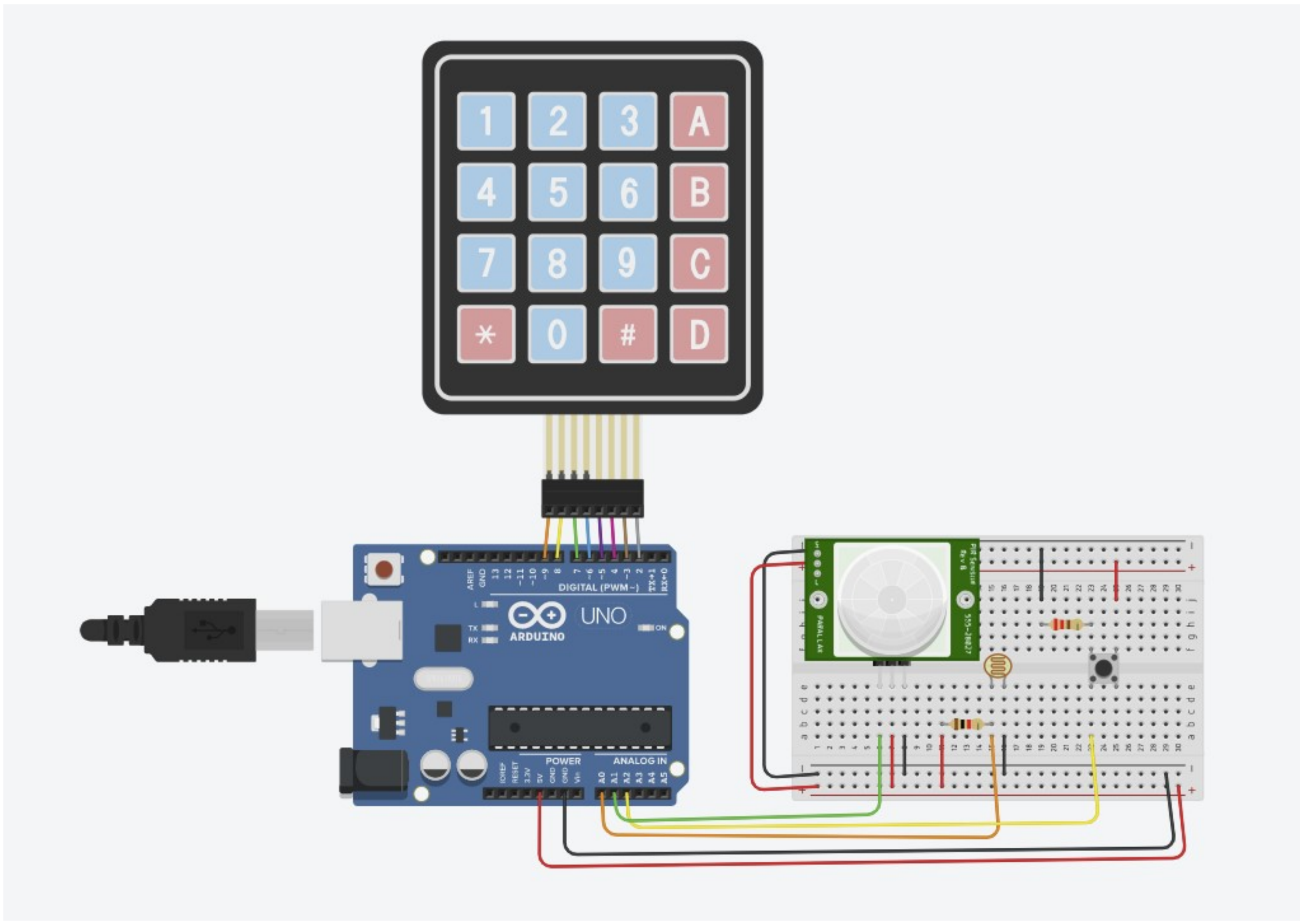


Figure 5: Input Block Circuit

```
};

byte rowPins[ROWS] = {9, 8, 7, 6};
byte colPins[COLS] = {5, 4, 3, 2};

Keypad customKeypad = Keypad(makeKeymap(hexaKeys), rowPins, colPins, ROWS, COLS);

void setup(){
  Serial.begin(9600);
  HM10.begin(19200);//bluetooth serial

  pinMode(LDR, INPUT);
  pinMode(NCSW, INPUT);
  pinMode(RX_PIN, INPUT);
  pinMode(TX_PIN, INPUT);
}

void loop(){ //output of sensors is shown in Serial Monitor
```

```
  keyPad(); //keypad

  bluetoothControl(); //bluetooth serial method
  //this code doesn't actually do anything because there are no
  //equivalent sensors to simulate a bluetooth module

  detectNight_Motion();//detect motion at night

  doorsensor(); //normally closed switch door sensor

  delay(150);
}

void keyPad() {
        char customKey = customKeypad.getKey();

          if (customKey){
      Serial.print("Key Input: ");
      Serial.println(customKey);
    }
}

void detectNight_Motion(){
        ph_val = analogRead(LDR);
          pir_state = digitalRead(PIRdist);

  if((threshold < ph_val) && (pir_state == HIGH)){
          Serial.println("Light on");
  } else{
          Serial.println("Light off");
  }
}

void doorsensor(){
        oc_state = digitalRead(NCSW);

    if (oc_state == HIGH){
      Serial.println("Door State: Opened");
    }
    else {
      Serial.println("Door State: Closed");
    }
}

void bluetoothControl() {//Bluetooth Control Method
        HM10.listen();  // listen the HM10 port
```

```
            while (HM10.available() > 0) {// if HM10 sends something then read
        appData = HM10.read();
        inData = String(appData);// save the data in string format
        Serial.write(appData);
            }

             if (Serial.available()) {// Read user input if available.
              delay(10);
              HM10.write(Serial.read());
             }
            if (inData == "L") {
        Serial.println("Door Locked");
            }
      if (inData == "U") {
        Serial.println("Door Unlocked");
      }
  }
```

7.1.2 Output Block

Below is the input block circuit diagram.
Code for the circuit is -

```
#define BUZZ 10
#define spREVOL 90
#include <Stepper.h>
#include <LiquidCrystal.h>

// initialize the library with the numbers of the interface pins
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
//Stepper myStepper(spREVOL, 6, 7, 8, 9);

int lstate;

void setup() {

  Serial.begin(9600);

  // set up the LCD's number of columns and rows:
  lcd.begin(16, 2);
}

void loop() {

  if (Serial.available() > 0)
  {
    lstate = Serial.read();
```

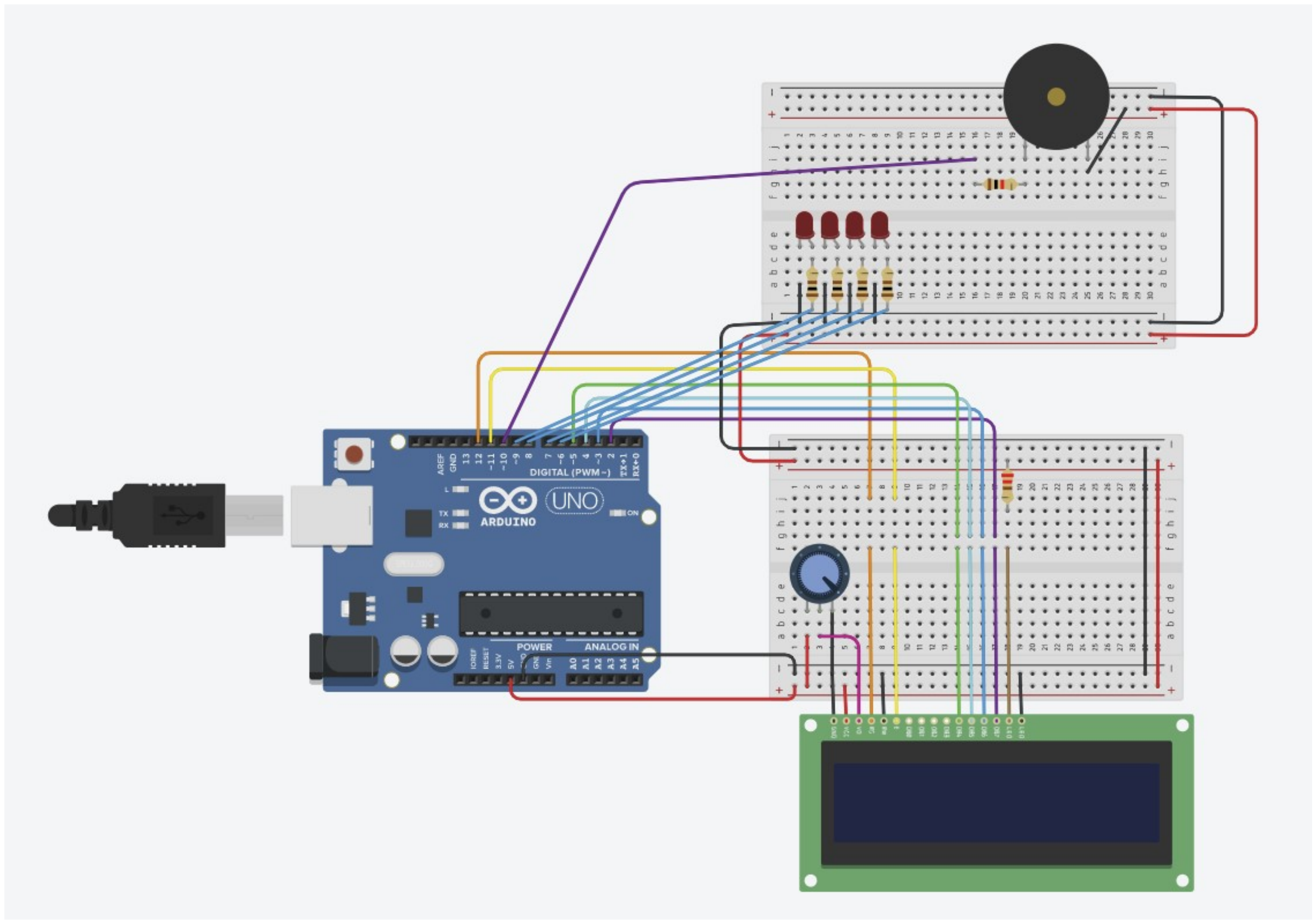


Figure 6: Output Block Circuit

```
    turnShaft(lstate); //makes stepper motor turn
    displayOut(lstate); //display output function
  }

}

int displayOut(int lstate)
{

  lcd.setCursor(0, 0);
  Serial.println(lstate);
    switch(lstate) {//4 states -> a locked successful, b unlocked successful,
      case 'a':
      lcd.print("Locked!");
      buzzer(true);
      noTone(BUZZ);
      break;
      case 'b':
      lcd.print("Unlocked!");
      buzzer(false);
```

```
        noTone(BUZZ);
        break;
        case 'c':
        lcd.print("Failed to Lock");
        break;
        case 'd':
        lcd.print("Failed to Unlock");
        break;
        default:
        lcd.print("Error");
        break;
      }
            delay(1000);
            lcd.clear();
}

void turnShaft(int lstate)
{
  if (lstate == 'a') {
    for (int count = 6; count < 10; count++) //right to left
    {
      digitalWrite(count, HIGH);
      delay(250);
      digitalWrite(count, LOW);
    }
    /*myStepper.step(stepsPerRevolution); //actual code to turn lock shaft.
    delay(500);*/
  } else if (lstate == 'b')
  {
    for (int count = 9; count >= 6; count--) //left to right
    {
      digitalWrite(count, HIGH);
      delay(250);
      digitalWrite(count, LOW);
    }
          /*myStepper.step(-stepsPerRevolution); //actual code to turn lock s
          delay(500);*/
  }
}

void buzzer(bool lock)
{
  if(lock)//locking
  {
    tone(BUZZ, 500);
    delay(500);
  } else if(!lock)//unlocking
```

```
        {
          tone(BUZZ, 500);
          delay(500);
              noTone(BUZZ);
          delay(500);
          tone(BUZZ, 500);
          delay(500);
        }
      }
```

In this scenario you will notice that LED on the board will light up when the conditions are dark enough and there is motion detected as seen in the video submitted labelled ”Night Mode Input”.

7.2 Complete Circuit

Unfortunately attempts to implement at least a majority of the circuit resulted in such a laggy result, the simulation was non-functional. Here's the code from such attempt

```
#define LDR A0 //Photoresistor pin
#define PIRdist A1 //PIR pin
#define NCSW A2 //reed switch pin (pushbutton is proxy to simulate.
#define RX_PIN 0 //RX pin for HM10 bluetooth module
#define TX_PIN 1 //TX pin for HM10 bluetooth module
#define BUZZ 5
#define spREVOL 90
#include <Stepper.h>
#include <LiquidCrystal.h>
#include <Keypad.h>
#include <SoftwareSerial.h>
//#include <Stepper.h>
//#include <Wire.h>
//#include <LiquidCrystal_I2C.h>

SoftwareSerial HM10(RX_PIN, TX_PIN);
//LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
//LiquidCrystal_I2C lcd(0x27, 16, 2);
//Stepper myStepper(spREVOL, 6, 7, 8, 9);
const byte ROWS = 4;
const byte COLS = 4;
const int threshold = 500;
int pir_state;
int ph_val;
int oc_state;
char appData;
const int latchPin = 3;
const int clockPin = 4;
const int dataPin = 2;
```

```
String inData = "";
byte leds = 0;
int lstate;
bool lock_state, door_state;
short a = 0, i = 0, s = 0, j = 0;

char keypressed;                 //Where the keys are stored it changes very often
char code[] = {'0'}; //The default code, you can change it or make it a 'n' digits one
char code_buff1[sizeof(code)];  //Where the new key is stored

char hexaKeys[ROWS][COLS] = {
  {'1', '2', '3', 'A'},
  {'4', '5', '6', 'B'},
  {'7', '8', '9', 'C'},
  {'*', '0', '#', 'D'}
};

byte rowPins[ROWS] = {13, 12, 11, 10};
byte colPins[COLS] = {9, 8, 7, 6};

Keypad customKeypad = Keypad(makeKeymap(hexaKeys), rowPins, colPins, ROWS, COLS);

void setup() {
  Serial.begin(9600);
  HM10.begin(9600);//bluetooth serial

  //lcd.init();
  //lcd.backlight();

  pinMode(LDR, INPUT);
  pinMode(PIRdist, INPUT);
  pinMode(NCSW, INPUT_PULLUP);
  pinMode(A3, OUTPUT);
  pinMode(A4, OUTPUT);
  pinMode(A5, OUTPUT);
  pinMode(BUZZ, OUTPUT);
  pinMode(latchPin, OUTPUT);
  pinMode(dataPin, OUTPUT);
  pinMode(clockPin, OUTPUT);
  //pinMode(RX_PIN, OUTPUT);
  //pinMode(TX_PIN, INPUT);

}

void loop() {
  keyPad();
  detectNight_Motion();
  doorsensor();
  //bluetoothControl();
}

//actuator
void displayOut(int lstate)
{
  if (lstate == 'a') {
    leds = 0;
    updateShiftRegister();
    delay(500);
    for (int i = 0; i < 6; i++)
    {
      bitSet(leds, i);
      updateShiftRegister();
      delay(500);
    }
    buzzer(true);
    noTone(BUZZ);
  } else if (lstate == 'b') {
    leds = 0;
    updateShiftRegister();
    delay(500);
    for (int i = 5; i >= 0; i--)
    {
      bitSet(leds, i);
      updateShiftRegister();
      delay(500);
    }
    buzzer(false);
    noTone(BUZZ);
  } else if (lstate == 'c') {
    leds = 0;
    updateShiftRegister();
    delay(500);
    for (int i = 2; i >= 0; i--)
    {
      bitSet(leds, i);
      updateShiftRegister();
      delay(500);
    }
    delay(500);
  } else if (lstate == 'd') {
    leds = 0;
    updateShiftRegister();
    delay(500);
    for (int i = 0; i < 3; i++)
    {
      bitSet(leds, i);
      updateShiftRegister();
      delay(500);
    }
    delay(500);
  }
  //lcd.clear();
}

void turnShaft(int lstate)
{
  int timecheck = millis();
  lock_state = true;
  if (lstate == 'a') {
    digitalWrite(A0, HIGH);
    delay(250);
    digitalWrite(A0, LOW);
    digitalWrite(A1, HIGH);
    delay(250);
    digitalWrite(A1, LOW);
    digitalWrite(A2, HIGH);
    delay(250);
    digitalWrite(A2, LOW);
  } else if (lstate == 'b')
  {
    lock_state = false;
    digitalWrite(A2, HIGH);
    delay(250);
    digitalWrite(A2, LOW);
    digitalWrite(A1, HIGH);
    delay(250);
    digitalWrite(A1, LOW);
    digitalWrite(A0, HIGH);
    delay(250);
    digitalWrite(A0, LOW);
  }
}

void buzzer(bool lock)
{
  int timecheck = millis();
  if (lock) //locking
  {
    tone(BUZZ, 500);
    delay(500);
  } else if (!lock) //unlocking
  {
    tone(BUZZ, 500);
    delay(500);
    noTone(BUZZ);
    delay(500);
    tone(BUZZ, 500);
    delay(500);
  }
}

void keyPad() {
  keypressed = customKeypad.getKey();


  if (keypressed == '*') {                    // * to open the lock
          Serial.println(keypressed);
    //lcd.clear();
    //lcd.setCursor(0, 0);
    //lcd.print("Enter code");             //Message to show
    getCode();                            //Getting code function
    if (a == sizeof(code))  {      //The GetCode function assign a value to a (it's corre
      turnShaft('b');                     //Open lock function if code is correct
      displayOut('b');
    }
    else {
      displayOut('d');
    }
  }
  if (keypressed == '#') {                  //To change the code it calls the changecode fun
    //ChangeCode();
    //lcd.clear();
    turnShaft('a');                  //When done it returns to standby mode
    displayOut('a');
  }
}

void getCode() {
  i = 0;                    //All variables set to 0
  a = 0;
  j = 0;

  while (keypressed != 'A') {                                   //The user press A to conf
    keypressed = customKeypad.getKey();
    if (keypressed != NO_KEY && keypressed != 'A' ) {      //If the char typed isn't A and
      //lcd.setCursor(j, 1);                                  //This to write "*" on the L
      //lcd.print("*");
      j++;
      if (keypressed == code[i] && i < sizeof(code)) {       //if the char typed is correc
        a++;
        i++;
      }
      else
        a--;                                                 //if the character typed is wro
    }
  }
  keypressed = NO_KEY;
}

void detectNight_Motion() {

  int timecheck = millis();

    ph_val = analogRead(LDR);
    pir_state = digitalRead(PIRdist);

  if ((threshold < ph_val) && (pir_state == HIGH)) {
    Serial.println("Light on");
    //digitalWrite(ledpin, HIGH);
    delay(500);
  }
  else {
    Serial.println("Light off");
    //digitalWrite(ledpin, LOW);
    delay(500);
  }
}

void doorsensor() {
  lstate = false;
  oc_state = digitalRead(NCSW);

  if (oc_state == HIGH) {
    //Serial.println("Door State: Opened");
    door_state = true;
  }
  else if (door_state == true) {
    //Serial.println("Door State: Locking");
    door_state = false;
    turnShaft('a');
  }
}

void updateShiftRegister()
{
   digitalWrite(latchPin, LOW);
   shiftOut(dataPin, clockPin, LSBFIRST, leds);
   digitalWrite(latchPin, HIGH);
}
```

The lack of the an I2C module to drive the LCD display was problematic as I had to use a shift register and LEDs to simulate a message being displayed to the screen. I was also again unable to simulate the bluetooth module nor a led pin as none were available. Please refer to the attached input block and output block videos to get an idea of how the tinkercad simulation could work.

7.3 BreadBoard Implementation Notes

I was able to implement every module in the breadboard part except for the bluetooth module on the input side. While the stepper motor was hooked up it did not perform as expected since the power supply was only supplying 2.5-3.2V, which was not enough to drive the motor. In the video labelled **"Output Block"** the driver board lights up but the stepper motor does not rotate. In the output block, an I2C module was used to drive the 1602A display.

7.3.1 Input Block

Input Block Implementation on BreadBoard

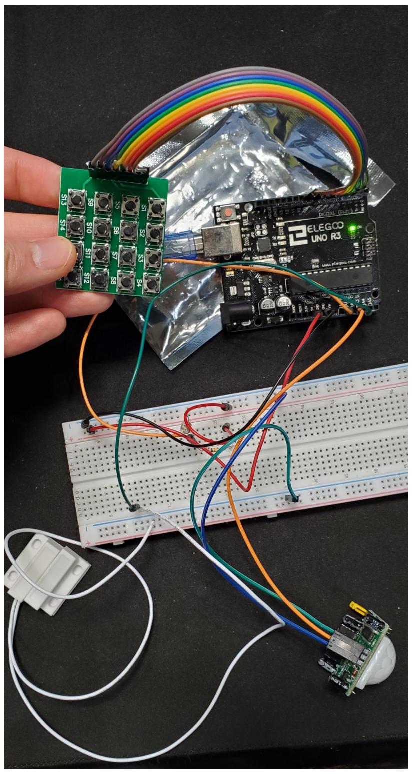


Figure 7: Input Block Circuit Implementation

Code for Input Block

```
        #define LDR A0 //Photoresistor pin
        #define PIRdist A1 //PIR pin
```

```
#define NCSW A2 //reed switch pin (pushbutton is proxy to simulate.
#define RX_PIN 10 //RX pin for HM10 bluetooth module
#define TX_PIN 11 //TX pin for HM10 bluetooth module
#include <Keypad.h>
#include <SoftwareSerial.h>

SoftwareSerial HM10(RX_PIN, TX_PIN);
const byte ROWS = 4;
const byte COLS = 4;
const int threshold = 500;
int pir_state;
int ph_val;
int oc_state;
int inData;
const int ledpin = 13;

char hexaKeys[ROWS][COLS] = {
  {'1', '2', '3', 'A'},
  {'4', '5', '6', 'B'},
  {'7', '8', '9', 'C'},
  {'*', '0', '#', 'D'}
};

byte rowPins[ROWS] = {9, 8, 7, 6};
byte colPins[COLS] = {5, 4, 3, 2};

Keypad customKeypad = Keypad(makeKeymap(hexaKeys), rowPins, colPins, ROWS, COLS);

void setup(){
  Serial.begin(9600);
  HM10.begin(9600);//bluetooth serial

  pinMode(LDR, INPUT);
  pinMode(NCSW, INPUT_PULLUP);
  pinMode(RX_PIN, INPUT);
  pinMode(TX_PIN, INPUT);
  pinMode(ledpin, OUTPUT);
}

void loop(){ //output of sensors is shown in Serial Monitor

  keyPad(); //keypad

  bluetoothControl(); //bluetooth serial method
  //this code doesn't actually do anything because there are no equivalent sensor

  detectNight_Motion();//detect motion at night
  doorsensor(); //normally closed switch door sensor

  delay(150);
}

void keyPad() {
  char customKey = customKeypad.getKey();

    if (customKey){
      Serial.print("Key Input: ");
      Serial.println(customKey);
    }
}

void detectNight_Motion(){
  ph_val = analogRead(LDR);
  pir_state = digitalRead(PIRdist);

  if((threshold < ph_val) && (pir_state == HIGH)){
    Serial.println("Light on");
    digitalWrite(ledpin, HIGH);
    delay(500);
  } else{
    Serial.println("Light off");
    digitalWrite(ledpin, LOW);
    delay(500);
  }
}

void doorsensor(){
  oc_state = digitalRead(NCSW);

    if (oc_state == HIGH){
      Serial.println("Door State: Opened");
    }
    else {
      Serial.println("Door State: Closed");
    }
}

void bluetoothControl() {//Bluetooth Control Method
    HM10.listen();  // listen the HM10 port

    while (HM10.available() > 0) {// if HM10 sends something then read
      inData = HM10.read();
      Serial.write(inData);
    }

     if (Serial.available()) {// Read user input if available.
      delay(10);
      HM10.write(Serial.read());
     }
    if (inData == '1') {
      Serial.println("Door Locked");
      digitalWrite(ledpin, HIGH);
      delay(1000);
    }
    if (inData == '0') {
      Serial.println("Door Unlocked");
      digitalWrite(ledpin, LOW);
      delay(1000);
    }
}
```

Code for Output Block

7.3.2 Output Block

Output Block Implementation on BreadBoard

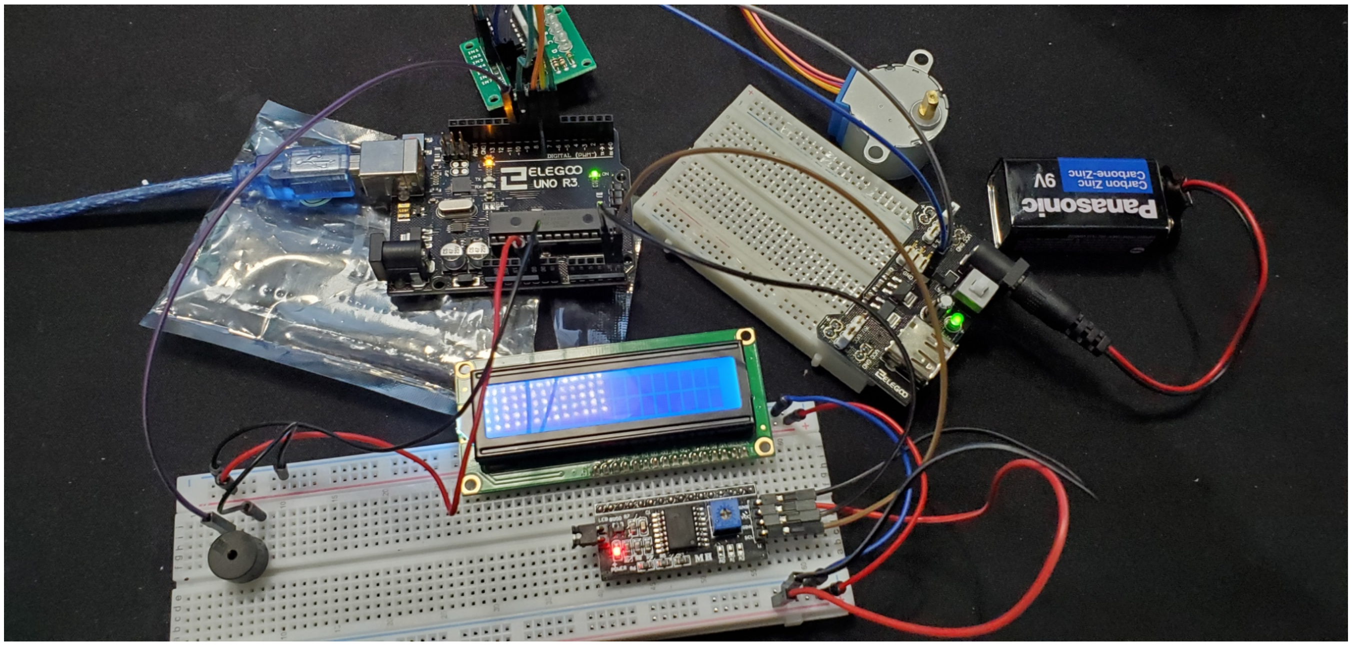


Figure 8: Output Block Circuit Implementation

```
  #define BUZZ 10
  #define spREVOL 2038
  #include <Stepper.h>
```

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

// initialize the library with the numbers of the interface pins
LiquidCrystal_I2C lcd(0x27, 16, 2);
Stepper myStepper(spREVOL, 6, 7, 8, 9);

int lstate;

void setup() {

  Serial.begin(9600);

  // set up the LCD's number of columns and rows:
  lcd.init();
  lcd.backlight();
}

void loop() {

  if (Serial.available() > 0)
  {
    lstate = Serial.read();
    turnShaft(lstate); //makes stepper motor turn
    displayOut(lstate); //display output function
  }

}

int displayOut(int lstate)
{
  Serial.println(lstate); //4 states -> a locked successful, b unlocked successfu
      if (lstate == 'a'){
        lcd.setCursor(0, 0);
        lcd.print("Locked!");
        buzzer(true);
        noTone(BUZZ);
        delay(1000);
      } else if (lstate == 'b') {
        lcd.setCursor(0, 0);
        lcd.print("Unlocked!");
        buzzer(false);
        noTone(BUZZ);
        delay(1000);
      } else if (lstate == 'c') {
        lcd.print("Failed to Lock");
        delay(1000);
      }else if (lstate == 'd') {
        lcd.setCursor(0, 0);
        lcd.print("Failed to Unlock");
        delay(1000);
      }
    lcd.clear();
}

void turnShaft(int lstate)
{
  if (lstate == 'a') {
    myStepper.step(spREVOL); //actual code to turn lock shaft.
    delay(500);
  } else if (lstate == 'b')
  {
    myStepper.step(-spREVOL); //actual code to turn lock shaft.
    delay(500);
  }
}

void buzzer(bool lock)
{
  if(lock)//locking
  {
    tone(BUZZ, 500);
    delay(500);
  } else if(!lock)//unlocking
  {
    tone(BUZZ, 500);
    delay(500);
  noTone(BUZZ);
    delay(500);
    tone(BUZZ, 500);
    delay(500);
  }
}
```

7.4 Complete Circuit

The following breadboard setup was used to test (mostly) the whole circuit. As seen in this circuit, the bluetooth module and the LED was not implmented as there were not enough pins. Furthermore, for whatever reason, while the stepper motor driver board would light up, the stepper motor itself would not function. Again, the bluetooth module and the LED circuit is not included, because of the lack of pins.

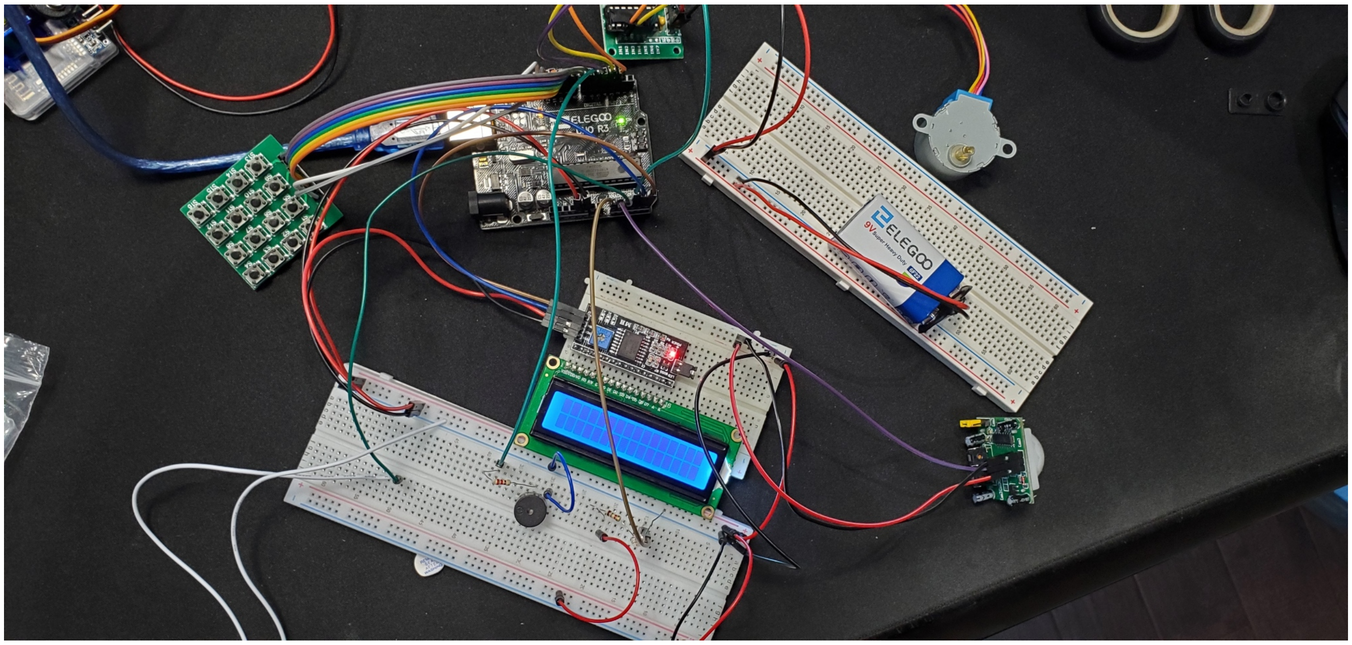


Figure 9: Complete Breadboard Setup

```
#define LDR A0 //Photoresistor pin
#define PIRdist A1 //PIR pin
#define NCSW A2 //reed switch pin (pushbutton is proxy to simulate.
#define RX_PIN 0 //RX pin for HM10 bluetooth module
#define TX_PIN 1 //TX pin for HM10 bluetooth module
#define BUZZ 5
#define spREVOL 90
#define statusTime 1000
#define soundTime 500
#define mTime 1500
#define waitLock 5000
#include <Stepper.h>
#include <Keypad.h>
#include <SoftwareSerial.h>
#include <Stepper.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

SoftwareSerial HM10(RX_PIN, TX_PIN);
LiquidCrystal_I2C lcd(0x27, 16, 2);
Stepper myStepper(spREVOL, 2, 3, 4, A3);
const byte ROWS = 4;
const byte COLS = 4;
const int threshold = 500, ledpin = 1;
int pir_state, ph_val, oc_state, lstate;
char appData;
String inData = "";
```

```
bool lock_state, door_state;
short a = 0, i = 0, s = 0, j = 0;

char keypressed;                 //Where the keys are stored it changes very often
char code[] = {'0'}; //The default code, you can change it or make it a 'n' digits one
char code_buff1[sizeof(code)];  //Where the new key is stored


char hexaKeys[ROWS][COLS] = {
  {'1', '2', '3', 'A'},
  {'4', '5', '6', 'B'},
  {'7', '8', '9', 'C'},
  {'*', '0', '#', 'D'}
};

byte rowPins[ROWS] = {13, 12, 11, 10};
byte colPins[COLS] = {9, 8, 7, 6};

//keypad pins - 6,7,8,9,10,11,12,13
//photoresistor pin A0
//pir pin A1
//ncsw pin A2
//buzzer pin 5
//i2c lcd A4,A5
//stepper motor - 2,3,4,A3

Keypad customKeypad = Keypad(makeKeymap(hexaKeys), rowPins, colPins, ROWS, COLS);

void setup() {
  Serial.begin(9600);
  HM10.begin(9600);//bluetooth serial

  lcd.init();
  lcd.backlight();

  pinMode(LDR, INPUT);
  pinMode(PIRdist, INPUT);
  pinMode(NCSW, INPUT_PULLUP);
  pinMode(BUZZ, OUTPUT);
  //pinMode(RX_PIN, OUTPUT);
  //pinMode(TX_PIN, INPUT);

}

void loop() {
  keyPad();
  detectNight_Motion();
  doorsensor();
  //bluetoothControl();
}

//actuator
void displayOut(int lstate)
{
  if (lstate == 'a') {
    lcd.setCursor(0, 0);
    lcdPrintOut("Locked!");
    buzzer(true);
    noTone(BUZZ);
  } else if (lstate == 'b') {
    lcd.setCursor(0, 0);
    lcdPrintOut("Unlocked!");
    buzzer(false);
    noTone(BUZZ);
  } else if (lstate == 'c') {
    lcd.setCursor(0, 0);
    lcdPrintOut("Failed to Lock");
    delay(500);
  } else if (lstate == 'd') {
    lcd.setCursor(0, 0);
    lcdPrintOut("Failed to Unlock");
    delay(500);
  }
  lcd.clear();
}


void lcdPrintOut(String message) {
  int timecheck = millis();
  int prev = 0;
  lcd.clear();
  lcd.print(message);
  prev = millis();
}

void turnShaft(int lstate)
{
  int timecheck = millis();
  lock_state = true;
  if (lstate == 'a') {
    while (millis() < timecheck + mTime) {
      myStepper.step(spREVOL); //actual code to turn lock shaft.
    }
  } else if (lstate == 'b')
  {
    lock_state = false;
    while (millis() < timecheck + mTime) {
      myStepper.step(-spREVOL); //actual code to turn lock shaft.
    }

  }
}

void buzzer(bool lock)
{
  int timecheck = millis();
  if (lock) //locking
  {
    tone(BUZZ, 500);
    delay(500);
  } else if (!lock) //unlocking
  {
    tone(BUZZ, 500);
    delay(500);
    noTone(BUZZ);
    delay(500);
    tone(BUZZ, 500);
    delay(500);
  }
}

void keyPad() {
  keypressed = customKeypad.getKey();

  if (keypressed == '*') {                      // * to open the lock
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Enter code");            //Message to show
    getCode();                          //Getting code function
    if (a == sizeof(code))  {      //The GetCode function assign a value to a (it's corre
      turnShaft('b');                   //Open lock function if code is correct
      displayOut('b');
    }
    else {
      displayOut('d');
    }
  }
  if (keypressed == '#') {                  //To change the code it calls the changecode fu
    //ChangeCode();
    lcd.clear();
    turnShaft('a');                 //When done it returns to standby mode
    displayOut('a');
  }
}

void getCode() {
  i = 0;                         //All variables set to 0
  a = 0;
  j = 0;

  while (keypressed != 'A') {                                    //The user press A to conf
    keypressed = customKeypad.getKey();
    if (keypressed != NO_KEY && keypressed != 'A' ) {      //If the char typed isn't A and
      lcd.setCursor(j, 1);                                 //This to write "*" on the LCD
      lcd.print("*");
      j++;
      if (keypressed == code[i] && i < sizeof(code)) {        //if the char typed is correc
        a++;
        i++;
      }
      else
        a--;                                               //if the character typed is wro
    }
  }
  keypressed = NO_KEY;
}

void detectNight_Motion() {

  int timecheck = millis();

    ph_val = analogRead(LDR);
    pir_state = digitalRead(PIRdist);

  if ((threshold < ph_val) && (pir_state == HIGH)) {
    Serial.println("Light on");
    digitalWrite(ledpin, HIGH);
    delay(500);
  }
  else {
    Serial.println("Light off");
    digitalWrite(ledpin, LOW);
    delay(500);
  }
}

void doorsensor() {
  lstate = false;
  oc_state = digitalRead(NCSW);

  if (oc_state == HIGH) {
    //Serial.println("Door State: Opened");
    door_state = true;
  }
  else if (door_state == true) {
    //Serial.println("Door State: Locking");
    door_state = false;
    turnShaft('a');
  }
}

void bluetoothControl() {//Bluetooth Control Method
  HM10.listen();  // listen the HM10 port

  while (HM10.available() > 0) {// if HM10 sends something then read
    inData = HM10.read();
  }

  if (Serial.available()) {// Read user input if available.
    delay(10);
    HM10.write(Serial.read());
  }
  if (inData == '1') {
    turnShaft('a');
  }
  if (inData == '0') {
    turnShaft('b');
  }
  if (iunData == 'Set') }
    setCodes() {

  }
}

/*void setCodes() {
  //set codes
}*/
```

More software features will be added such as setting codes and other features, etc.

8 Eagle Design

8.1 Schematic

In the the Eagle design, a 74HC165 PISO shift register will be used to reduce the number of pins need by the keypad from 8 to 4. The photoresistor, the MC-38 door sensor, the PIR sensor, the I2C module + LCD Display, the LED, the keypad, and the stepper motor.

References

[1] LockPickingLawyer. (2020). “”[1130] avoiding the trap, and bypassing keypad with magnet (hfeng)” accessed march 1, 2020,” Youtube, [Online]. Available: https://youtu.be/d-hBDjYtE1U?t=96.

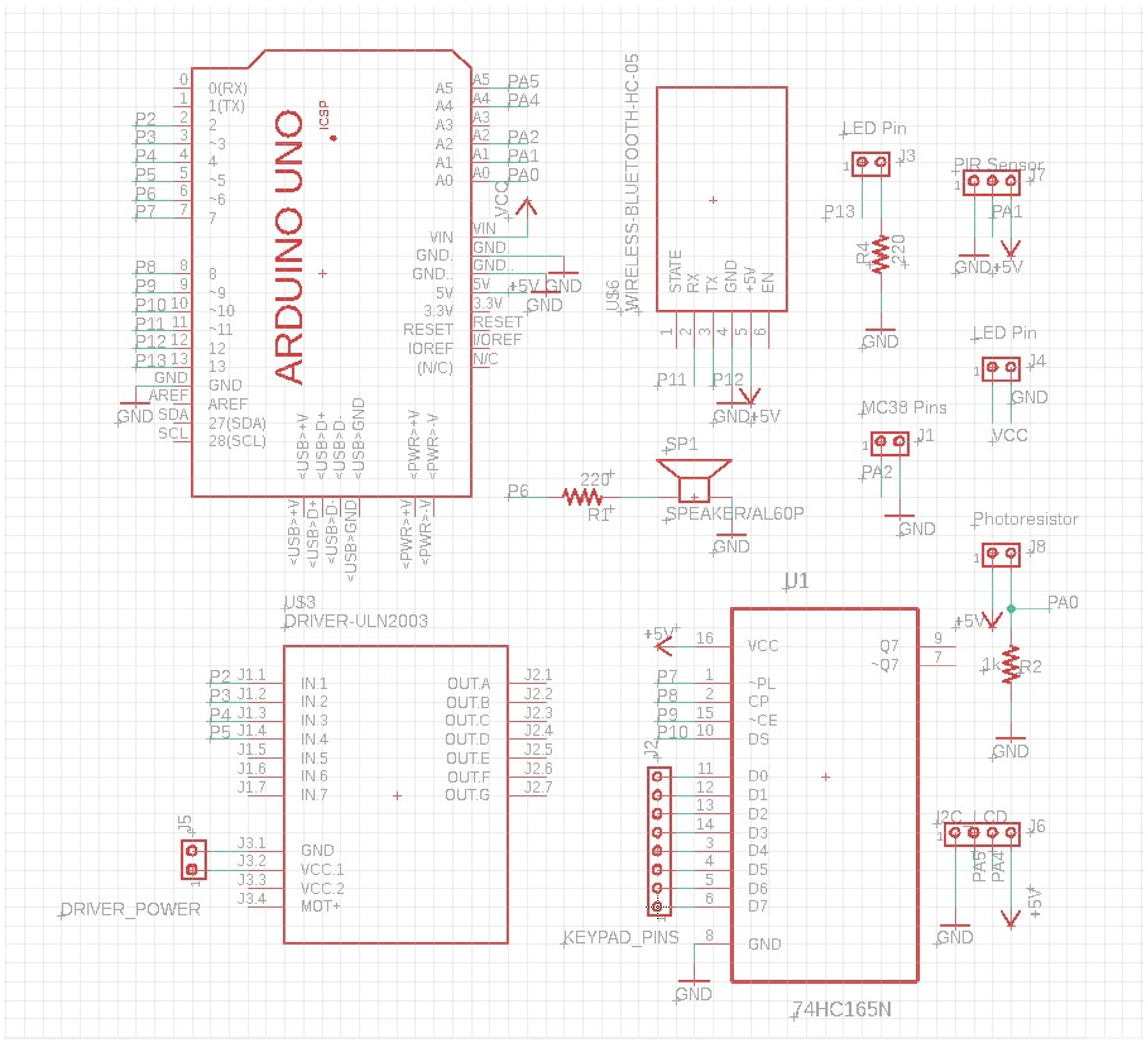


Figure 10: Complete Schematic