# ECE 342, Final Project, Door Group 7

Benjamin Warren, Blake Garcia, Elizabeth Lindsay

March 4th, 2022

## Contents

# 1 Project Description

Our group had the goal of creating a pet door that electronically opened without the input of anybody else besides the pet being present in front of it. We oversaw the design, testing, and building the system.

Inputs:
RFID_Query input is for RFID communication from compatible tags

Outputs:
Motor_Out is a PWM output to a DC compatible motor
LED_Out is a visible light output from 3 RGB LED's

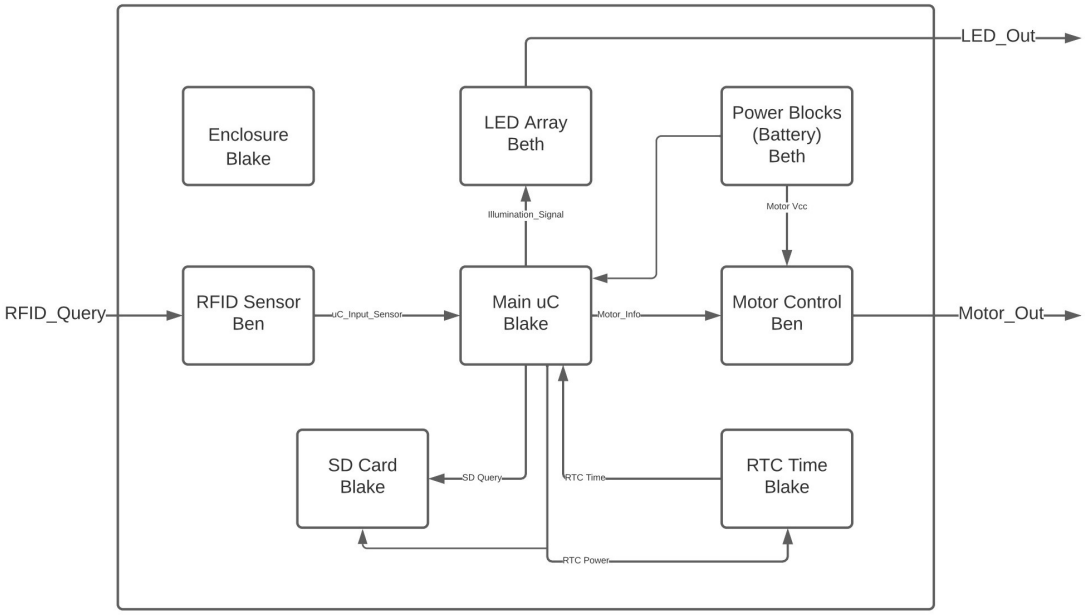



Figure 1: This is the high-level block diagram of the system as a whole. The outputs point out towards the right, and the inputs point into the system on the left.

The hardware diagram in figure 1 shows the system broken into the key components, and how they interact.

1. Each arrow an line represent an interface connection
2. The names indicate which team member was the leading contributor to that block
3. The enclosure indicates the 3D modeled and printed container for the electronics

## 1.1 Interface definitions

The interface definition table describes the type of signal and parameters that connect each block within the system. They allow the designers to have a "black box" mentality when working on their aspects of the design. For example, the LED Array Block does not "care" how or why different colors are turned on or off; it just needs to use the digital signal from the microcontroller and the 5V/GND connection to emit light.

| Signal Name: | Input System: (Input from) | Output System: (Output to) | Signal Type: | Specification(s): |
|---|---|---|---|---|
| RFID_Query | Outside World | RFID Sensor | Asynchronous Frequency Query | Operating Frequency: 13.56 MHz; Operating Voltage: 3.3 V |
| uC_Input_Sensor | RFID Sensor | Main uC | SPI Serial Communication | 10 Mbps, Baud Rate 9600, 4 MHz SPI Clock Signal |
| RTC_Power | Main uC | RTC Time | DC Power | 3.3 V uC Power @ 50 mA Max; 3.0 V Battery @ .046 mA |
| RTC_Time | RTC Time | Main uC | Serial Communication | Baud Rate 9600; 7-bit I2C Address |
| uC_Power | Power Block (Battery) | Main uC | DC Power | 20 V Max VIN; 5.5 V USB; |
| Motor_Info | Main uC | Motor Control | PWM Digital Signal | 5V Logic Operating Voltage PWM Signal; Max Logic Current 20 mA |
| Motor_Vcc | Power Block (Battery) | Motor Control | DC Power | 9-12V Drive Voltage, 2 A Max Current |
| Illumination_Signal | Main uC | LED Array | DC Power | Continuous Forward Current: 20 mA; VOLTAGE DROP: 2.2 V (RED) and 3.2 V (GREEN/BLUE) Max Power dissapation = 80mW |
| LED_Out | LED Array | Outside World | Visible Light | Luminous Intesity: 2 - 4 cd (RED/BLUE), 4 - 6 cd (GREEN); Wavelength: 625 nm (RED) / 520 nm (GREEN) / 460 (BLUE) |
| Motor_Out | Motor Control | Outside World | Physical Motion | 200 RPM Max; Torque: 2.2Kg / cm |
| SD_Query | Main uC | SD Module | SPI Serial Communication | 9600 Baud Rate, 4MHz MHz SPI Clock Signal |

Figure 2: This is a table of the interface definitions in the systems.

# 2 Electrical Schematic

## 2.1 High Level Description

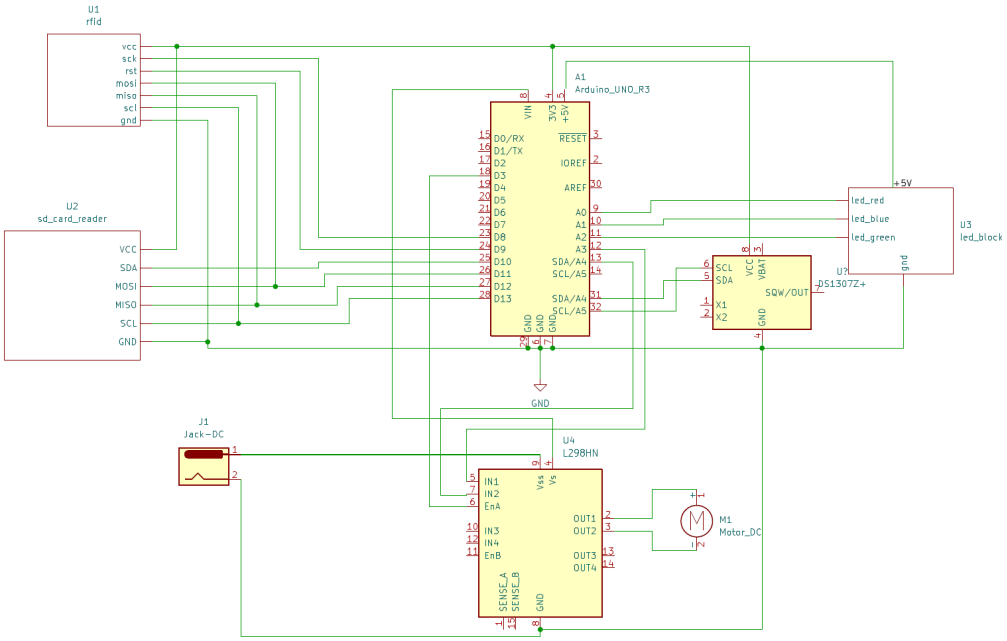
The pet door consists of several major components, an RFID reader, an SD card reader, a time clock, an Arduino Uno R3, LED block, and a motor control

If we "black-box" the system, it reads an RFID signal, and then powers a motor to operate the door, and LEDS accordingly. The inputs and output are as follows.

Inputs: This reads a RFID tag and decides if it is valid

Outputs: Runs a motor to open and close the door, and light up LEDS according to the operation state

## 2.2 Schematic Image



## 2.3 Module Descriptions

### 2.3.1 Module Part U1 - RFID

Integration/interface Specifications:

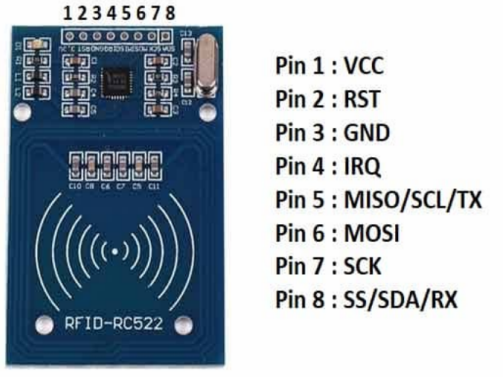
1. RFID Sensor will operate at 13.56 MHz
2. RFID Sensor will accept multiple cards and verify identity
3. RFID Sensor will operate at 3.3v supply
4. RFID will be compatible with data limit on Arduino UNO R3 (10 MBits/s max)

RC-522 RFID sensor:

1. Accepts RFID tags at 13.56 MGHz
2. Max data transfer of 10MBit/s (Compatible with Arduino Uno)
3. Has two tags, receives Hexadecimal ID
4. Runs on a MFRC522
5. 3.3v voltage supply

Pinout:

1. VCC - 3.3v input from Arduino
2. RST - Low input from Arduino powers down, High input resets
3. GND - 0v output to Arduino
4. IRQ - Interrupt Output to Arduino when RFID tag is sensed (not used)
5. MISO - Serial Data output to Arduino
6. MOSI - SPI input from Arduino
7. SCK - Serial clock input from Arduino
8. SDA - Signal Input from Arduino



### 2.3.2 Module Part U2 - SD Card

Integration/interface Specifications:

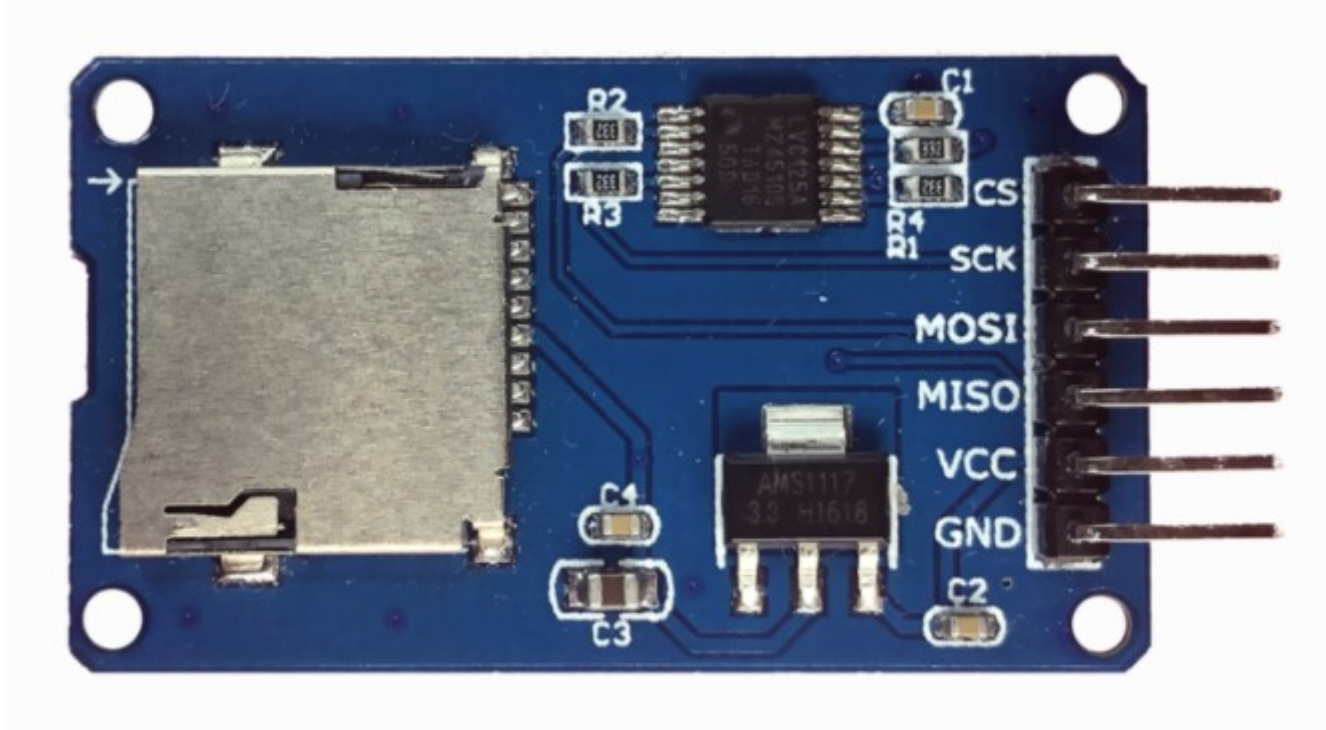
1. The SD module must be able to read and write data to an SD card.
2. The SD module must be re-accessible and compatible (able to share the SPI connection with the RFID Module specifically) within the system.

SD Card Reader:

1. Module Operates at 3V3.
2. Uses Serial Peripheral Interface of the Arduino to exchange data.

Pinout:

1. VCC - 3.3v input from Arduino
2. RST - Low input from Arduino powers down, High input resets
3. GND - 0v output to Arduino
4. MISO - Serial Data output to Arduino
5. MOSI - SPI input from Arduino
6. SCK/SCL - Serial clock input from Arduino
7. CS/SDA - Clock Select Pin



### 2.3.3 Module Part U3 - LED Block

Integration/interface Specifications:

1. LEDs run on a 5V supply.
2. Digital Signal does not draw more than 20 mA from the Arduino.
3. LEDs turn on/off as directed.

LED Block:

1. Accepts a digital signal from the Arduino to signal the On/Off state of three different LEDs.
2. Digital Signals are Active-Low.
3. Outputs light into the world.

Pinout: More information regarding the Pinout and design of the LED block can be found under the PCB section of the document.

### 2.3.4 Module Part U4 - Motor Control

Integration/interface Specifications:

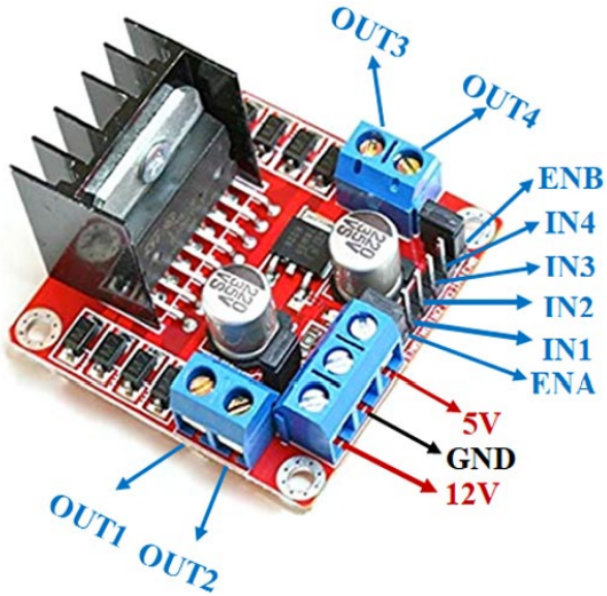
1. Motor Control must have 5V Logic Operating Voltage
2. Motor Control must operate motor with a 9v-12v Operating voltage
3. Motor Control Logic current must be within arduino capability (20 mA)

L298n Motor Controller:

1. Logic operating voltage: 5v Logic operating current: 0-36mA
2. Requires 12v DC input
3. Runs on a L298 Dual H(full)-Bridge driver(can run two motors simultaneously)
4. Outputs PWM signal for DC motor
5. Uses logic signal for spinning direction

Pinout:

1. IN1/IN2 - Motor A Input pin from Arduino to set direction of rotation
2. ENA - PWM input from Arduino to set speed of motor A
3. OUT1 - Output for motor A
4. 12V - 12v input from DC power
5. 5v - 5v output to Arduino for power
6. GND - GND for Arduino and power supply



### 2.3.5 Module Part U5 - DS1307 RTC

Integration/interface Specifications:

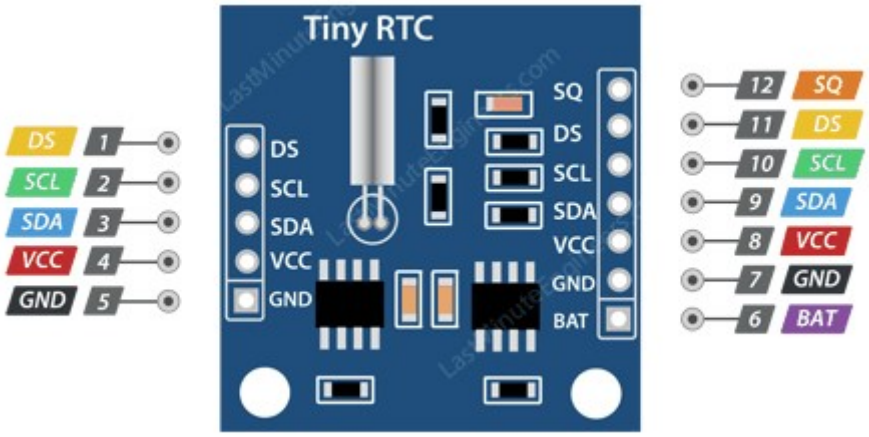
1. The module needs to operate using 3V3 connection.
2. The real-time clock needs to operate when no power is supplied to the microcontroller.
3. The real-time clock needs to be accurate within one minute of actual time.

DS1307:

1. Operates on a 3v3 supply voltage from the microcontroller.
2. Continually runs off of a CR1220 Battery.
3. Communicates with the Arduino via I2C connection on the SDA and SCL pins.

Pinout:

1. SCL - Serial Clock Interface (controls I2C connection)
2. SDA - Serai Data / Address Connection (Sends information via I2C connection)
3. VCC - Powers module when Arduino UNO is powered.
4. GND - Creates common ground connection with Arduino
5. BAT - Powers circuit when Arduino is not powered.



### 2.3.6 Module Part A1 - Arduino UNO

Integration/interface Specifications:

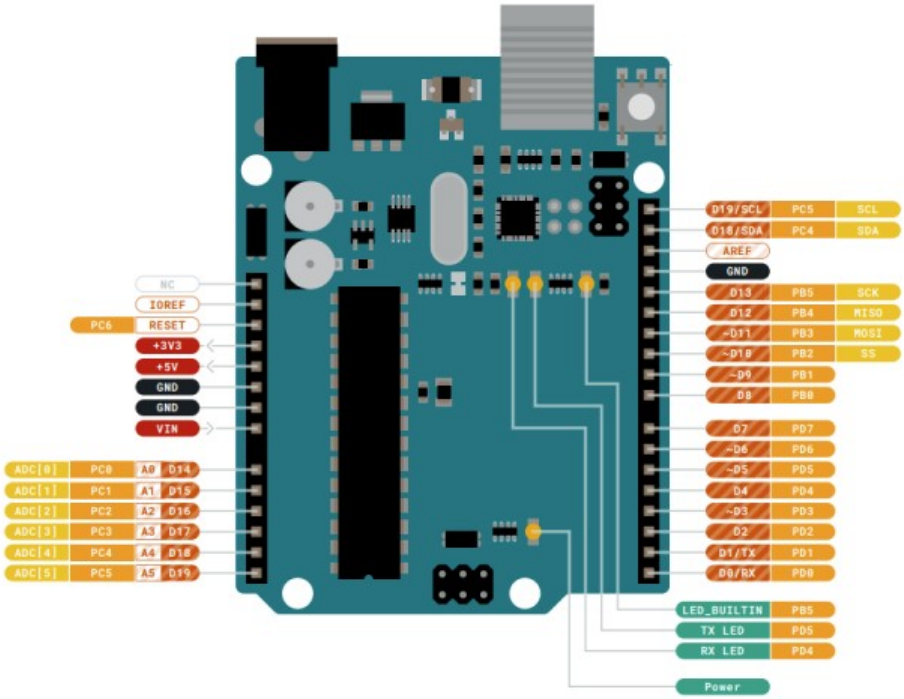
1. Send and receive 5V (High) / 0V (Low) digital signals.
2. Communicate with peripheral devices over I2C or SPI connections.
3. Powered by 12V power supply.
4. Output 5V and 3.3V VCC supply.

Arduino UNO:

1. Runs off of a 20V Max Vin power input or 5.5 V USB Connection.
2. Sends and receives 5V HIGH digital signals.
3. Uses an I2C connection to interact with the RTC module.
4. Communicates with multiple distinct modules over Serial Peripheral Interfaces (only one active at a time).

Pinout:

1. Pins D2, D3, D9 D14 - D17: Send digital logic signals to activate other modules.
2. Pins 5V, 3V3, GND, VIN: Provide power to Arduino and other modules.
3. Pins D13, D12, D11, D10, D8: Used to create two distinct SPI connections.
4. Pins D18 and D19: Communicate over I2C with RTC module.



# 3 Code

## 3.1 High Level Description

The code for this project is written in Arduino's Language, similar to C/C++ (although it is object-oriented). The system requires a microcontroller to read in an input via a Serial Peripheral Interface connection with the RFID module, interpret the input as a string of characters, terminate the connection with the RFID module so it can reuse SPI connection to interface with an SD card reader that will check the RFID string against known strings, and check if it is within the unlock window for that RFID tag. If the microcontroller does not recognize the tag or the tag is not in the unlock window, the door will remain closed. If the tag is recognized and within the unlock period, the microcontroller will open and close the door and record the encounter on a separate file within the SD card.

The code used will be broken down in four segments: the inclusion of header files and declared global variables, the initial setup routine, the main loop, and finally all of the additional functions created for ease-of-use. Comments are included in the code documents as well as a brief description of each segment. For display purposes it has been broken down into four separate files, but the code used to program the Arduino itself is contained in a single file as a combination of all four segments.

## 3.2 Headers and Global Variables

The first segment of code begins by defining the purpose of the microcontroller in comments and then including the proper header files. All header files were downloaded in the Arduino IDE Library Manager. The first one, "MsTimer2.h" uses interrupts triggered by Timer/Counter2 to run a specified functions (included later in the code). The "SPI.h" library provides the functions to interact with the Serial Peripheral Interface connections for the RFID and SD card modules. The "MFRC22.h" provides functions to interact with RFID card reader via the SPI interface. The "RTClib.h" header file allows the microcontroller to interact with the DS1307 RTC module and read/write the correct time information. Finally the "SD.h" library provides functions to interact with the SD card, so the microcontroller can read and write from the proper files.

The segment continues by declaring useful global variables. Most of them are pin definitions so the structure of the code is more readable and intuitive (instead of referring to pins by their number, they are referred to by a descriptive name). Two class variables are also declared, the MFRC522 RFID module and the DS1307 RTC module.

```
1  // This code is for the main uc
2  //
3  // It needs to:
4  // *Read in the time from the RTC module
5  //
6  // *Read in data from the SD card module
7  //
8  // *Write to the SD card
9  //
10 // *Read in from the RFID Sensor (double check the
11 //  pins needed/used for this)
12 //
13 // *Output three signals to the motor controller about
14 //  opening/closing the door when in the correct time
15 //  frames for each pet
16 //
17 // *Output three seperate signals for each of the three
18 //  RGB LED values.
19
20 // Libraries included:
21   // MsTimer2.h controls the interrupts to create
22   // the low frequency PWM signal for the blue LED
23   #include "MsTimer2.h"
24
25   // SPI library for the RFID module to communicate
26   // with the Arduino
27   #include "SPI.h"
28
```

```
29    // RFID library (https://github.com/miguelbalboa/rfid)
30    #include "MFRC522.h"
31
32    // Include the RTC library to control the RTC module
33    #include "RTClib.h"
34
35    // Include the SD card library
36    #include "SD.h"
37
38  // Define three output pins for the motor controller
39  // JUST NEEDS TO BE DIGITAL WRITE PINS
40  int enable = 3;
41  const int motor_pwm_pin = 3; // Not needed to be PWM
42  const int motor_control_fPin = 18;
43  const int motor_control_rPin = 17;
44
45
46  // Define three output pins for the LEDs
47  // Red and Green just need constant Digital Output
48  // but Blue needs low frequency PWM
49  // ACTIVE LOW
50  const int LEDredPin = 14;
51  const int LEDbluePin = 15;
52  const int LEDgreenPin = 16;
53
54  // Define the reset and sda pins
55  const int pinRST = 9;
56  const int pinSDA = 8;
57
58  // Define the SD Card SS
59  const int SD_ss = 10;
60
61  // Set up the RFID module on the Ardiuno
62  MFRC522 mfrc522(pinSDA, pinRST);
63
64  // Set up the rtcmodule
65  RTC_DS1307 rtc;
66
67
68  // A global variable to refer to an open or closed door state
69  int door_open = 0;
```

### 3.3 Setup Routine

The purpose of the setup routine is to initialize the Input/Output responsibilities of the microcontroller and prepare it for continuously running the main loop. The setup has to declare the pin mode of the digital pins controlling the LEDs and the motor, and write the correct values to them. The LEDs are active low, so the LEDpinRed is set low to turn on the Red LEDs signifying a closed door, and the other LED pins are set high to turn them off. All motor pins are set low. The SPI connection protocol is initiated and the RFID module is initially connected to the interface using the two pins (SDA and RST). The code includes informative prints to the Serial Monitor which are not necessary for door functionality but are useful for informative purposes. The MsTimer library discussed in Section 3.2.1 is initialized to have a 325 ms period (about 3/8 of a second) and call the function "pulse" (discussed in section 3.2.4) upon interrupt.

The setup routine continues by initializing the RTC module and checking if it is running. Due to the independent power source it runs off of, it should not need to reset or adjust the date/time value. An informative message is printed to the serial monitor to inform a user of the functionality of the RTC module.

```
1  void setup() {
2
3    // Assumes the default position of the door is CLOSED:
4      // LEDredPin will be HIGH (indicating closed door)
5      // LEDbluePin and LEDgreenPin will be LOW (indicating closed door)
6      // motor_control_fPin and motor_control_rPin will be LOW (indicating motor is
           off)
7
```

```
8       // Sets LED pins to output
9       pinMode(LEDredPin, OUTPUT);
10      pinMode(LEDbluePin, OUTPUT);
11      pinMode(LEDgreenPin, OUTPUT);
12
13      // Sets motor control pins to output
14      // Motor does to be on pin 3 for PWM signal
15      //pinMode(motor_control_fPin, OUTPUT);
16      //pinMode(motor_control_rPin, OUTPUT);
17      //pinMode(motor_pwm_pin, OUTPUT);
18
19      // Writes high values to all LEDs except red
20      // because they are ACTIVE LOW
21      digitalWrite(LEDredPin, LOW);
22      digitalWrite(LEDbluePin, HIGH);
23      digitalWrite(LEDgreenPin, HIGH);
24
25      // Writes low values to motor control pins
26      digitalWrite(motor_control_fPin, LOW);
27      digitalWrite(motor_control_rPin, LOW);
28      digitalWrite(motor_pwm_pin, LOW);
29
30      // Initialize the SPI connection
31      SPI.begin();
32
33      // Initialize the Proximity Coupling Device (PCD)
34      mfrc522.PCD_Init(pinSDA, pinRST);
35
36      // Initialize the serial monitor
37      Serial.begin(9600);
38      Serial.println(F("Beginning Setup Routine!"));
39
40      // Initialize the PWM signal for the blue LED
41      // to call the flash function each 325 ms
42      Serial.println(F("Beginning Setup Routine for PWM signal!"));
43      MsTimer2::set(325, pulse);
44
45      // Initialize RTC connection
46       Serial.println(F("Beginning Setup Routine for RTC connection!"));
47      if (!rtc.begin()){
48        Serial.println(F("Couldn't find RTC"));
49        Serial.flush();
50        while (1) delay(10);
51      }
52
53      // Check if RTC module is already running
54      if (!rtc.isrunning()){
55        Serial.println(F("RTC is NOT running, let's set the time!"));
56        rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
57      }
58
59      else if (rtc.isrunning()){
60        Serial.println(F("RTC is already running!"));
61      }
62
63
64
65      DateTime now = rtc.now();
66      Serial.print(F("The current time is "));
67      Serial.print(now.hour());
68
69      Serial.print(F(":"));
70      Serial.println(now.minute());
71  }
```

## 3.4 Main Routine

The main routine runs a continuous loop checking for an RFID signal from the RFID module until one is detected. If no tag is detected, it will continue to do nothing until one is found. After

it is found, it appends each byte of the tag into a string in hexadecimal format. The allowed users are hardcoded into the program, although as long as the string of the hexadecimal representation is known it is possible to re-program the microcontroller to look for seperate tags operating at the correct frequency for the RFID modules to interact with. If the string does not match either allowed user, the function returns to checking for an RFID tag.

If the tag is recognized, the microcontroller calls the "check unlock" function against the string to see if the current time is within the unlock time (further details in Section 3.2.4). Since there are two recognized tags, there are two possible calls to the "check unlock" function. If either returns true, then the actual door sequence begins, which follows this procedure:

1. Turn off the Red LED and begin the PWM sequence for the blue LED to flash while the door is opening.
2. Configure the motor to operate in the forward direction until the door is open. The rate motor will operate at a consistent speed, so the time it takes to open will be constant (set by the "delay(int ms)" function).
3. The PWM signal for the blue LEDs will terminate and the LEDbluePin will be manually set high to ensure correct state.
4. The pins controlling the motor will be set low to turn off the motor.
5. The LEDgreenPin will be set low to activate the green LED.
6. The system will wait a period of time for the pet to pass through the door using the "delay(int ms)" function.
7. The system will switch back to flashing the blue LED and turn off the green LED using the same process as before.
8. The motor will operate in reverse for the same period to close the door.
9. The system will revert back to the initial state once the door has closed, with the blue LEDs off and the Red LEDs activated. All motor pins will be set low to ensure the deactivation of the motor.

```
1  void loop() {
2    // String to store userID
3    String userID = "";
4    int val = 0;
5
6
7    // It will run an empty loop until it detects an RFID card
8      // PICC_IsNewCardPresent will return true if
9      // a RFID tag/card is present
10     if (mfrc522.PICC_IsNewCardPresent()) {
11       // Second loop to check for the RFID card
12
13       if(mfrc522.PICC_ReadCardSerial()) {
14         // Read the RFID tag
15         for (byte i = 0; i < mfrc522.uid.size; i++) { // read id (in parts)
16           userID.concat(String(mfrc522.uid.uidByte[i]< 0x10 ? " 0" : " "));
17           userID.concat(String(mfrc522.uid.uidByte[i], HEX));
18         }
19
20         // Change ID tag string to uppercase for readability
21         userID.toUpperCase();
22
23         // If the RFID tag is the string of the first pet
24         if (userID.substring(1) == "83 9A E8 0B"){
25           // Check to see if the time is within the time of
26           // SD card file.
27           val = check_unlock(userID.substring(1));
28         }
29
30         // If the RFID TAG is the string of the second pet
31         else if (userID.substring(1) == "63 27 64 11"){
32           // Check to see if the time is within the time of
```

```
33            // SD card file.
34            val = check_unlock(userID.substring(1));
35          }
36
37          // The RFID tag is neither pet
38          else {
39            Serial.println(F("RFID not recognized!"));
40          }
41
42          // DOOR OPENING SEQUENCE
43          if (val){
44            // Turn off RED LEDs
45            digitalWrite(LEDredPin, HIGH);
46
47            // Begin flashing the BLUE LEDs with interrupts
48            MsTimer2::start();
49
50            // Send a signal to the motor_control_fPin to open the door
51            digitalWrite(motor_control_fPin, HIGH);
52            digitalWrite(motor_control_rPin, LOW);
53            digitalWrite(motor_pwm_pin, HIGH);
54
55
56            // Wait until door is open
57              // Wait some amount of time
58              // 5 seconds for this example
59              delay(7000);
60
61            // End flashing the BLUE LEDs
62            MsTimer2::stop();
63            digitalWrite(LEDbluePin, HIGH);
64
65            // Turn off motor_control_fPin
66            digitalWrite(motor_control_fPin, LOW);
67            digitalWrite(motor_pwm_pin, LOW);
68
69            // Turn on the GREEN LEDs
70            digitalWrite(LEDgreenPin, LOW);
71
72            // Wait for some amount of time
73              // Undetermined how long to leave door open
74              // 5 seconds for now
75              delay(2000);
76
77            // Turn off the GREEN LEDs
78            digitalWrite(LEDgreenPin, HIGH);
79
80            // Resume Flashing BLUE LEDS
81            MsTimer2::start();
82
83            // Turn on the motor_control_rPin to close the door
84            digitalWrite(motor_control_rPin, HIGH);
85            digitalWrite(motor_control_fPin, LOW);
86            digitalWrite(motor_pwm_pin, HIGH);
87
88            // Wait until door is closed
89              // Undetermined how long it will take
90              delay(6650);
91
92            // Turn of motor_contrsol_rPin to leave the door closed
93            digitalWrite(motor_control_rPin, LOW);
94            digitalWrite(motor_control_fPin, LOW);
95            analogWrite(motor_pwm_pin, 0);
96
97            // Turn of the BLUE LEDs
98            MsTimer2::stop();
99            digitalWrite(LEDbluePin, HIGH);
100
101           // Turn on the RED LEDs
102           digitalWrite(LEDredPin, LOW);
103
104
105       }
```

```
106
107         else /* Door is locked */ {
108           Serial.println(F("Door is not in unlock period"));
109
110         }
111       }
112     }
113 }
```

## 3.5 Additional Functions

There are three additional support functions: pulse, check unlock, and check unlocked. The pulse function is designed to interact with the PWM signal from Timer/Counter2 interrupts, while the check unlock and check unlocked are supporting the microcontrollers ability to check the current time against the listed unlock times for different recognized strings.

### 3.5.1 Pulse

1. Return Value: None
2. Function Parameters: None
3. Purpose: Alternates the logic value of the LEDbluePin, effectively flashing the LED on and off.

### 3.5.2 Check Unlock

1. Return Value: integer
2. Function Parameters: String "petID"
3. Purpose: checks if the recognized string "petID" is within the unlock period on the SD card and records the encounter if it is. Returns one if the unlock is successful, zero if it is not.

This function begins by terminating the SPI connection to the RFID module and initializing the connection to the SD card. If no SD card is present or recognized, it will loop through this section of code until one is inserted. It will open a file "pets.txt" for reading, and run two loops (once for each registered RFID tag) to read the data from the file and check against the values read, calling the "check unlocked" function. If "check unlock" returns true at any point in the function, it will close the "pets.txt" file, open the "record.txt" and record the encounter. It will terminate the SPI connection with the SD card and reinitialize the connection with the RFID tag and return a value of one. If "check unlocked" returns false for the entire loop, then the RFID tag is not in an unlock period so the door will not open and the same process of terminating the SD connection and re-initializing the RFID connection will occur, except not record will be kept and the function will return 0.

### 3.5.3 Check Unlocked

1. Return Value: bool
2. Function Parameters: DateTime "now", DateTime "unlock", TimeSpan "constraint"
3. Purpose: checks if the current time measured by the RTC module is within the recorded times on the SD card. Returns true if within the unlock period.

This function checks the current time measured by the RTC against the unlock time stored in the SD card and determines if the current time is greater than the beginning of the unlock period and less than the beginning of the unlock period plus the constraint timespan. Returns true if it is within the unlock period and false otherwise.

```
/*
 * This function operates as expected!
 */
// Set up the low frequency PWM pulse
// Set up the low frequency PWM pulse
void pulse(){
  static boolean output = HIGH;

  digitalWrite(LEDbluePin, output);
  output = !output;
}


// Create four unlock times and an unlock time span for
// the specified pet from the SD card. Check if the current
// time is between the unlock begin period and the unlock
// begin period plus the unlock end period
int check_unlock(String petID){
    // We need to open the file

    // File variable
    File myFile;

    // Informative Message
    Serial.println(F("Initializing SD Card!"));
    SPI.end();

    // Loop to make sure SD card is written
    while (!SD.begin(10)){
      Serial.println(F("Initialization Failed!"));
      delay(500);
    }

    // Open text file with unlock windows
    myFile = SD.open("pets.txt", FILE_READ);

    for (int i = 0; i < 2; i++){
      // These are all the unlock times we need to set up
      String IDin = "";
      TimeSpan constraint;
      DateTime unlock1, unlock2, unlock3, unlock4;
      DateTime unlocks[4] = {unlock1, unlock2, unlock3, unlock4};
      DateTime curr_time = rtc.now();

      for (int j = 0; j < 11; j++){
        // Read in the petID from the text file
        // Append each character into a string
        IDin += char(myFile.read());
        Serial.println(IDin);
      }

      // Read the comma... only present in text file for readability
      char junk;
      junk = char(myFile.read());

      // Create an empty string to read the
      // unlock and constraint times from.
      String temp = "";
      int temp2 = 0;

      for (int k = 0; k < 4; k++){
        String temp = "";
        // Read in the first two characters

        temp += char(myFile.read());
        temp += char(myFile.read());

        // Print them for error-handling
        temp2 = temp.toInt();

        Serial.print(F("The first unlock time (in int format): "));
        Serial.println(temp2);

```

```
74          // Read the last comma / newline character
75          junk = char(myFile.read());
76
77          // Create an unlock variable with the int from text file
78          unlocks[k] = DateTime(2022, 1, 1, temp2);
79        }
80
81        // Read in final two characters and append them to the string
82        // Move the cursor to the next line
83        temp = "";
84        temp += char(myFile.read());
85        temp += char(myFile.read());
86        temp2 = temp.toInt();
87
88        constraint = TimeSpan(0, 0, temp2, 0);
89
90        junk = char(myFile.read());
91        junk = char(myFile.read());
92
93        // If this is the right pet line
94        // Compare the current time against the
95        if (IDin == petID){
96
97          // Loop through all five unlock times
98          for (int x = 0; x < 4; x++){
99            // Function returns true if the current time is within an unlock time +
                  the constraint
100           bool unlocked = check_unlocked(curr_time, unlocks[x], constraint);
101
102           // If the file is in an unlock time, it must close the text file it is
                  reading from,
103           // open the other text file to record the encounter, and then close the
                  other text file.
104           if (unlocked){
105             // Close the open file
106             myFile.close();
107
108             // Open the file meant to record the encounter
109             myFile = SD.open("record.txt", FILE_WRITE);
110
111             // Write the message "Pet with ID tag [RFID TAG NUMBER} accessed the
                    door
112             // on: [MONTH]/[DATE]/[YEAR] at [current_hour] hours and [
                    current_minute] minutes.
113             myFile.print(F("Pet with ID tag "));
114             myFile.print(petID);
115             myFile.print(F(" accessed the door on: "));
116             myFile.print(curr_time.month());
117             myFile.print(F("/"));
118             myFile.print(curr_time.day());
119             myFile.print(F("/"));
120             myFile.print(curr_time.year());
121             myFile.print(F(" at "));
122             myFile.print(curr_time.hour());
123             myFile.print(F(" hours and "));
124             myFile.print(curr_time.minute());
125             myFile.println(F(" minutes."));
126             myFile.flush();
127             myFile.close();
128
129             // Turns off the SD SPI connection
130             // and turns on the RFID Connection
131             SPI.end();
132             SPI.begin();
133             mfrc522.PCD_Init(pinSDA, pinRST);
134
135
136             return 1;
137           }
138         }
139       }
140     }
141     myFile.close();
```

```
142
143     // Turns off the SD SPI connection
144     // and turns on the RFID Connection
145     SPI.end();
146     SPI.begin();
147     mfrc522.PCD_Init(pinSDA, pinRST);
148
149     return 0;
150 }
151
152
153
154 bool check_unlocked(DateTime now, DateTime unlock, TimeSpan constraint){
155   if (now.hour() < unlock.hour()){
156     // It is before the unlock period
157     return 0;
158   }
159
160   else if (((now.hour() * 60) + now.minute()) > (((unlock.hour() + constraint.hours
        ()) * 60) + (unlock.minute() + constraint.minutes()))){
161     // It is after the unlock period
162     return 0;
163   }
164
165   else {
166     //It must be between the unlock period
167     return 1;
168   }
169
170 }
```

# 4 Mechanical Drawings

## 4.1 Electronics Enclosure

The enclosure was designed using the CAD program Fusion 360 and 3D printed on an Ender3 V2 printer with a mixture of PLA and PTEG filament. The process took about four days, which includes time spent re-printing when designs changed and restarting the process when the print failed.

The main goals of the enclosure are as follows:

1. The enclosure needs to securely store the electronics and protect them from the environment (to a reasonable extent).
2. The enclosure needs to hold the motor directly above the door and provide the necessary support to open and close it.
3. The enclosure needs to maintain small distances between electronic devices for wires to connect easily.
4. The enclosure needs to have access to the SD card on the Arduino microcontroller and have an opening for the LED lights to emit light on the outside of the enclosure.

To accomplish these requirements, the enclosure was designed with "shelves" for the electronics components (specifically the motor controller and the Printed Circuit Board) elevating their position towards the ceiling. Rectangular spaces were left open to allow for access inside and outside the enclosure. The motor was securely fastened to the wall and surrounded by a supporting enclosure.

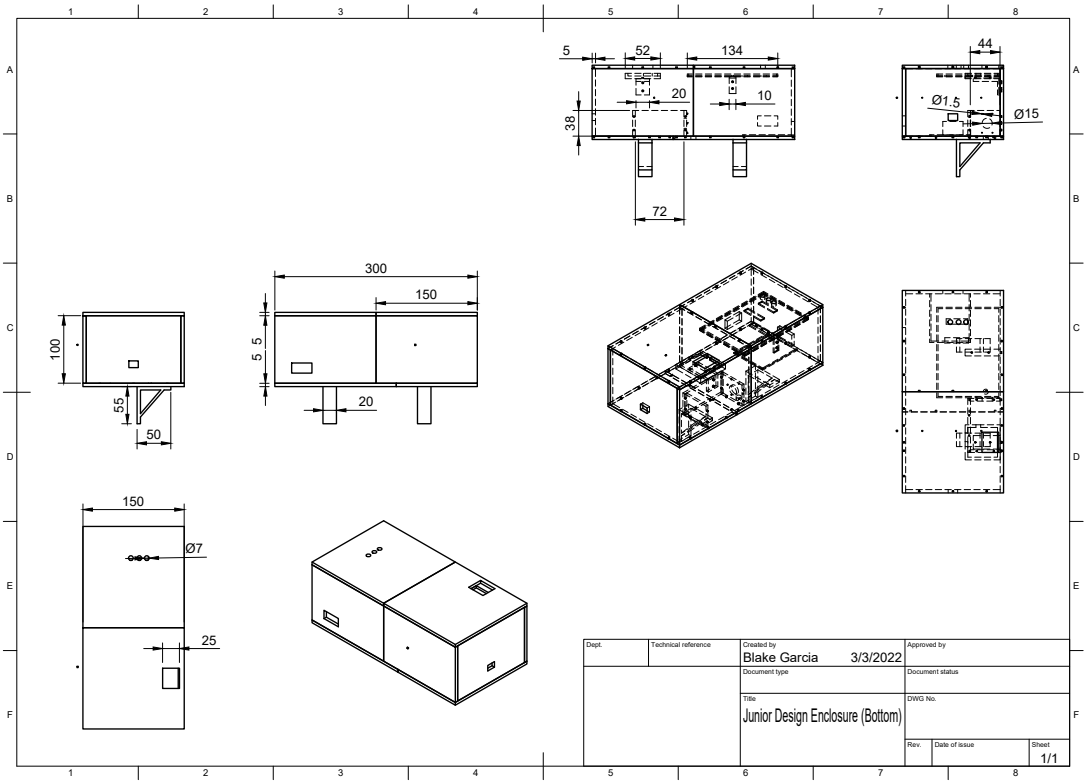



Figure 3: This is a layout with top, front, side, and isometric views of the electronics enclosure. Relevant dimensions are annotated.

## 4.2 Electronic Component Dimensions

Each subsection includes a figure of the physical dimensions of the device.

### 4.2.1 Arduino Uno

The Arduino Uno is the microcontroller used in the system. It is also attached to a shield with an RTC module and SD card module, so

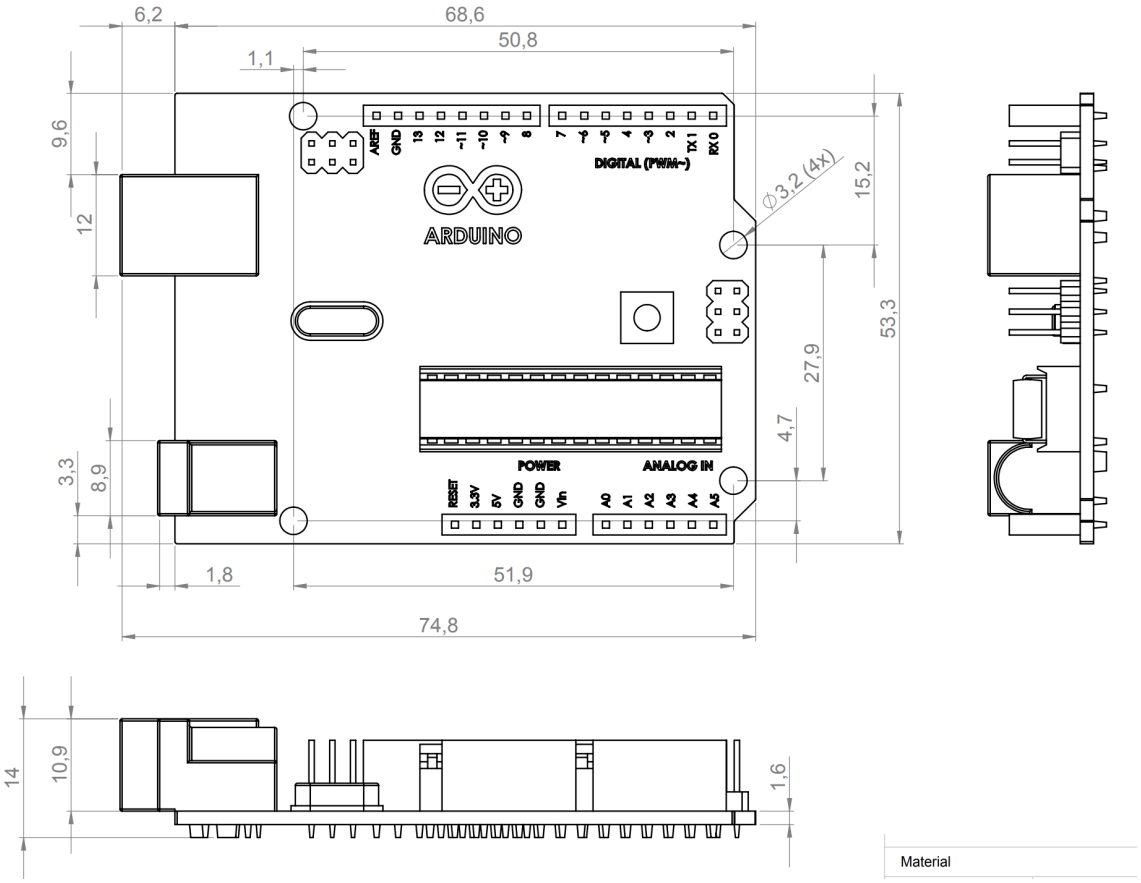



Figure 4: This is a physical representation of the Arduino Uno.

### 4.2.2 Motor Controller

There is no 3D model of the motor controller integrated circuit on the data sheet, however we measured it as a square with edges of 45 mm and a height of 34 mm at the highest point (the heat sink).

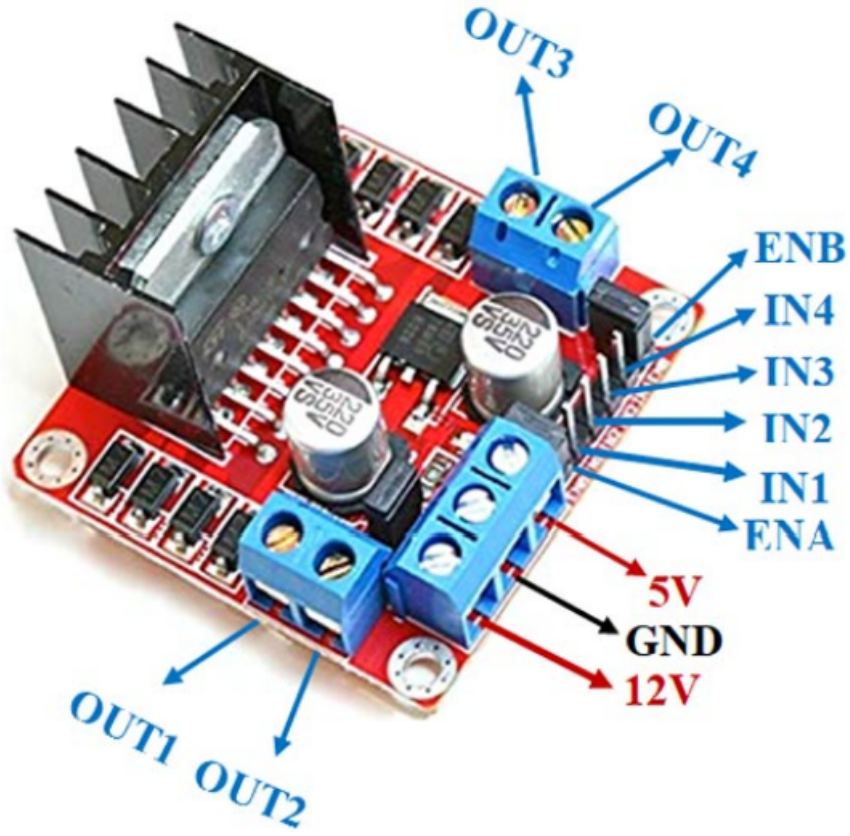



Figure 5: This is a physical representation of the motor controller.

### 4.2.3 Motor

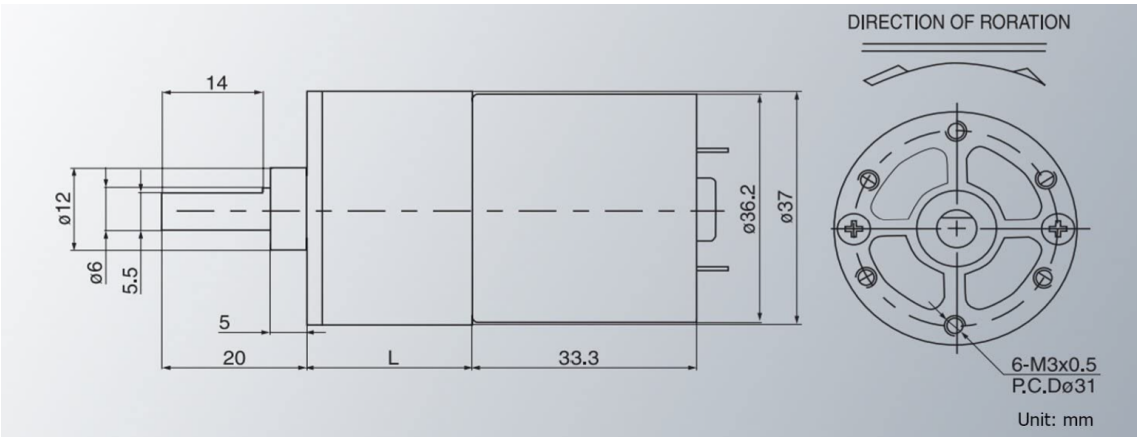



Figure 6: This is a physical representation of the motor.

### 4.2.4 Printed Circuit Board

This is included in the "Printed Circuit Board" section of the document.

# 5 Printed Circuit Board

The PCB was designed with for the connections between the LED block and the micro controller block. Created on KiCad, based on an electrical schematic, it was printed using OSH Park. The PCB will create a solid connection method for the LED's, resistors, transistors, batteries, and the Arduino Uno used to create a functioning LED system.

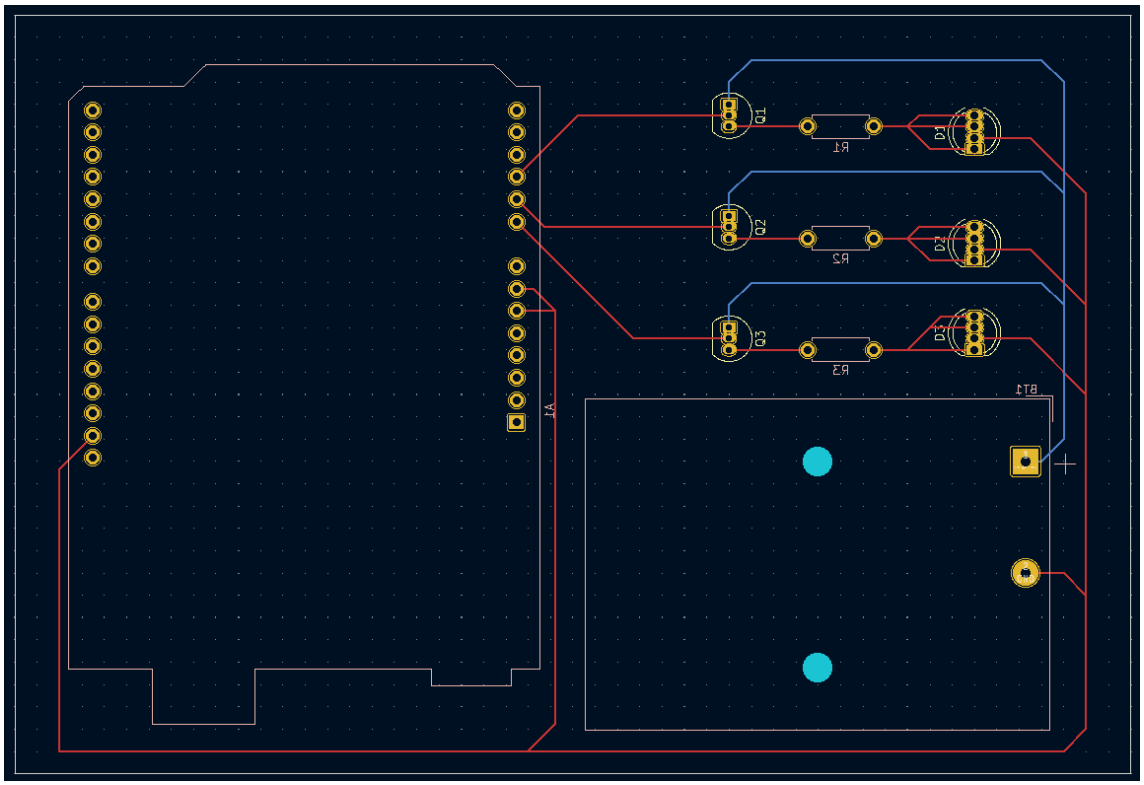



Figure 7: This is the final entire design of the PCB including all layers.

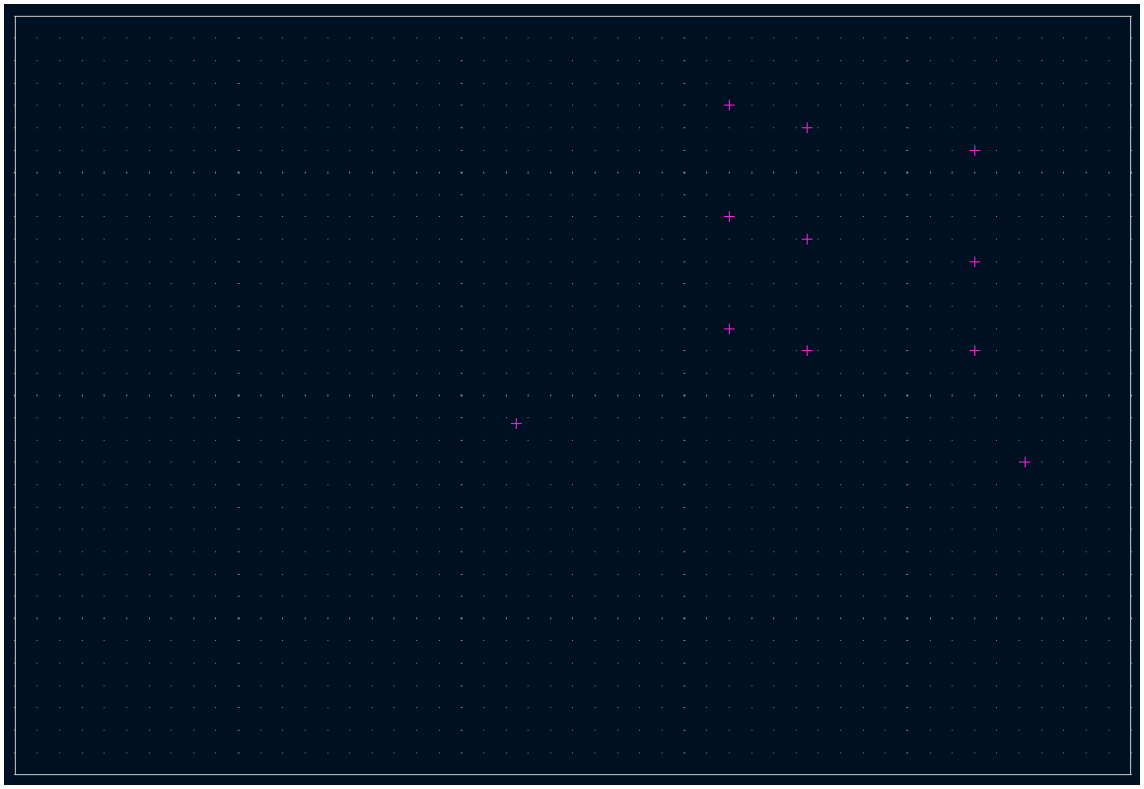


Figure 8: This is Edge Cuts layer of the PCB, which marks the outline of the board.

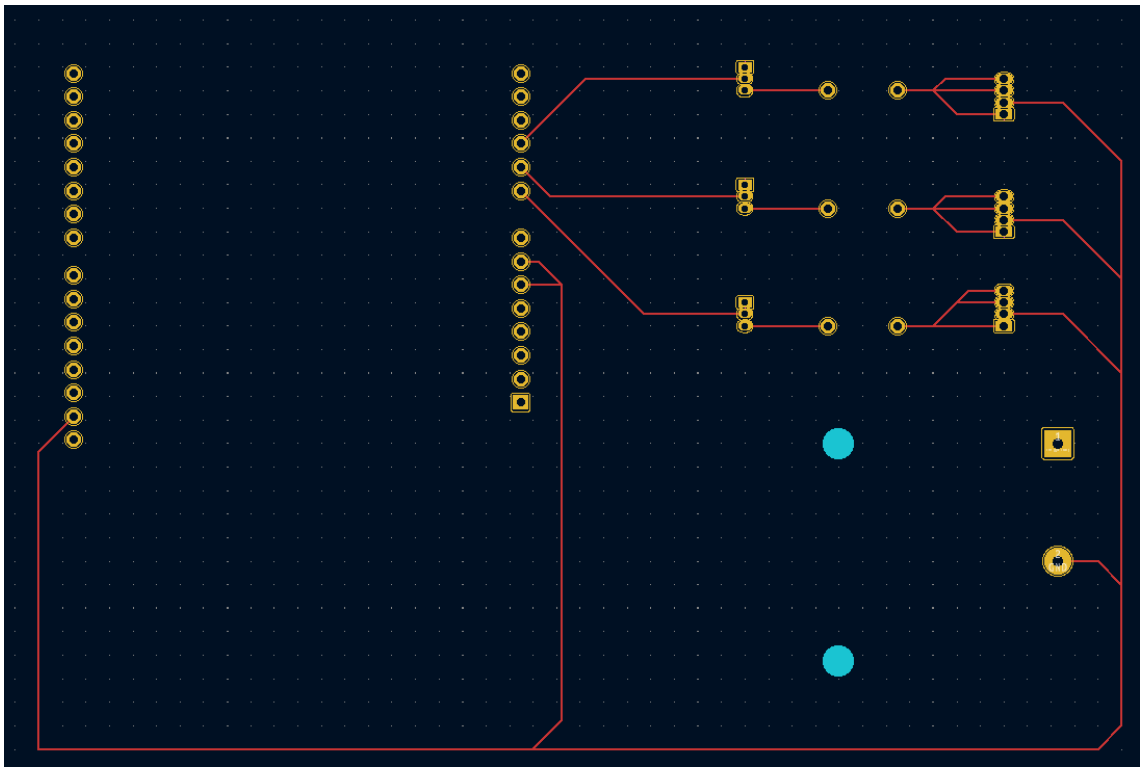


Figure 9: This is the front copper layer of the PCB, which creates all the connections on the front side of the board.

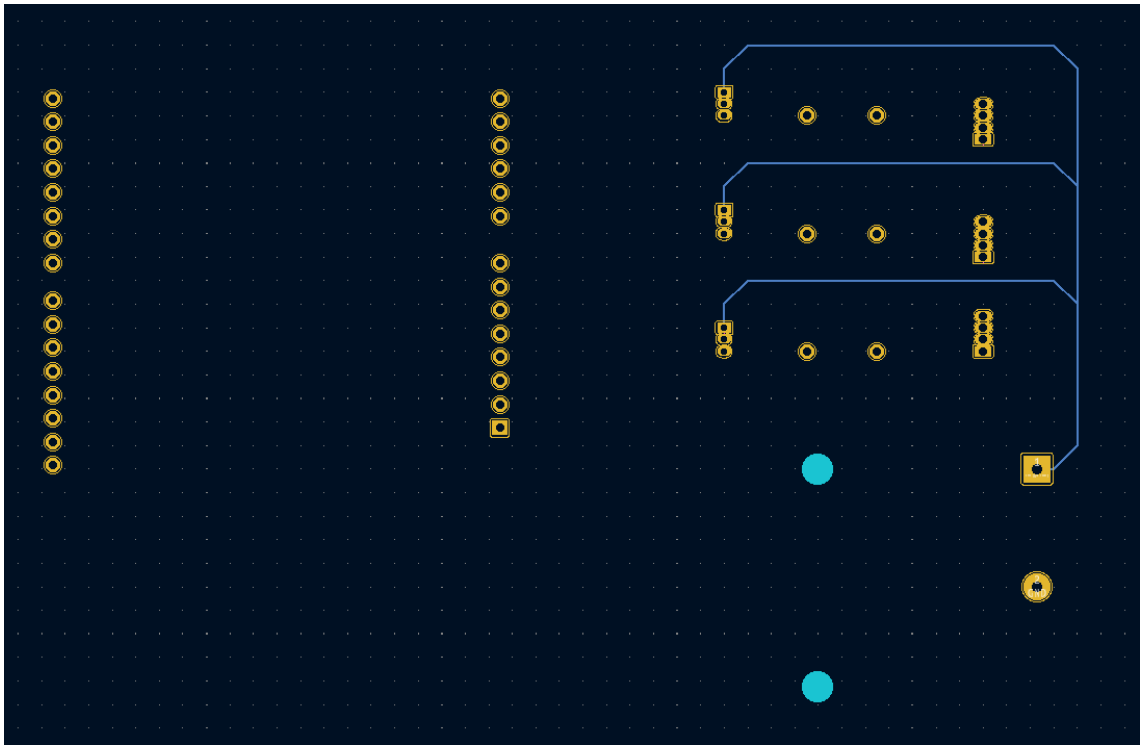


Figure 10: This is the back copper layer of the PCB, which creates all the connections that are required on the backside of the board, to prevent overlaps with the connections on the front side.

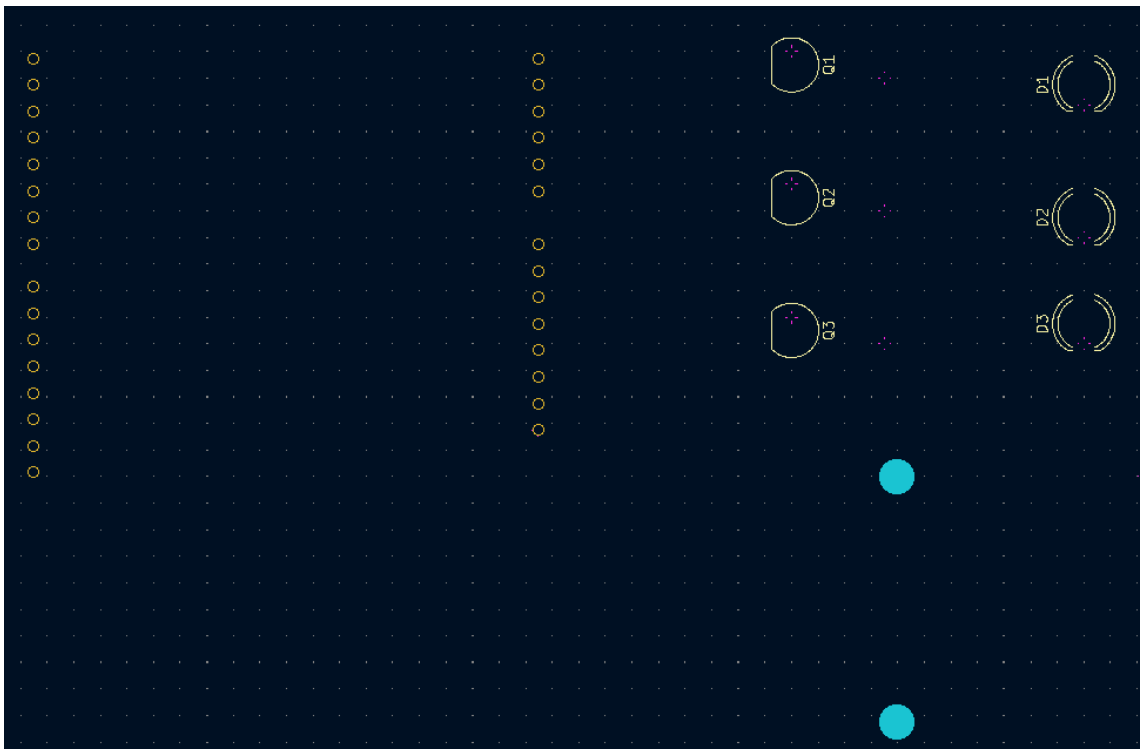



Figure 11: This is the front silk screen layer of the PCB, which shows where all the components on the front side will be placed.

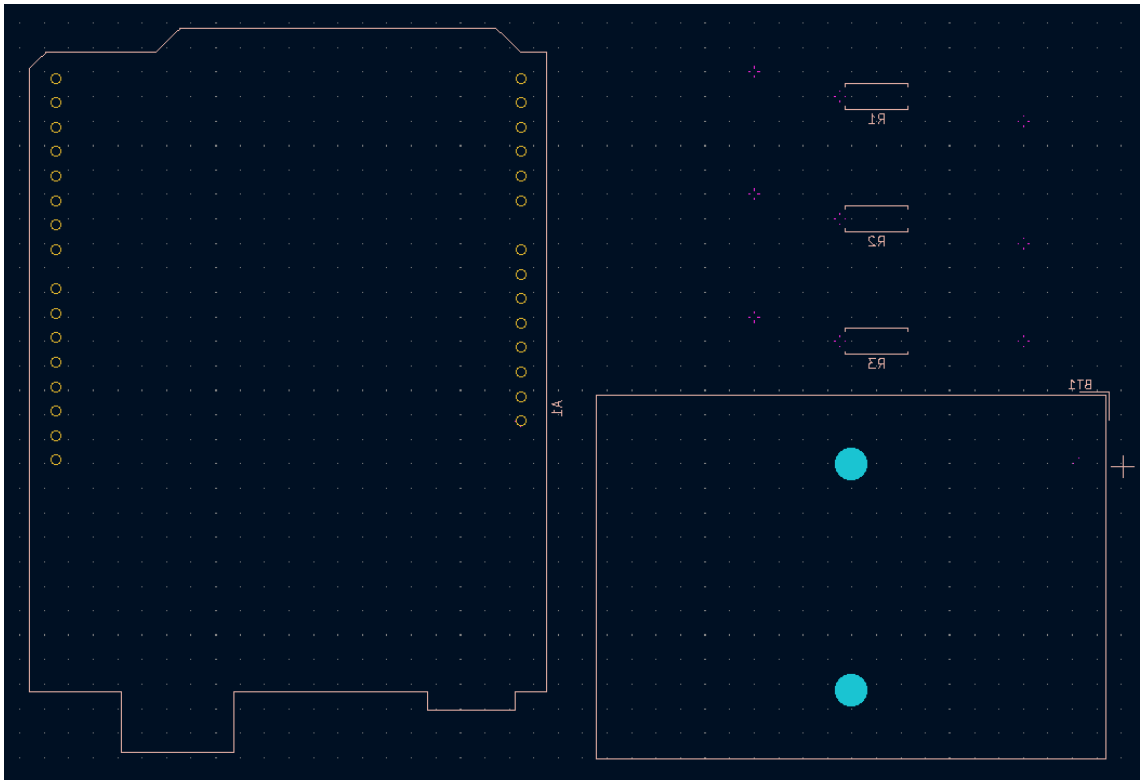



Figure 12: This is the back silk screen layer of the PCB which shows where all the components on the back side will be placed.

# 6 Bill of Materials

This is the Bill Of Materials for the entire system. The bill includes all of items used, their ID's, price, and relevant data sheets.

| Item name | ID | Qty | Price Per Unit | Place of Purchase | Manufacturer | Data Sheet | Subtotal | Total |
|---|---|---|---|---|---|---|---|---|
| Arduino UNO | A1 | 1 | $23.00 | Amazon | Arduino CC | Arduino Uno Rev3 | $23.00 | |
| RFID-RC522/RFID tags | U1 | 1 | $7.99 | Amazon | SunFounder | MFRC522 | $7.99 | |
| HiLetGo Data Logger Arduino Shield | U2/U5 | 1 | $7.39 | Amazon | HiLetGo | DS1307 | $7.39 | |
| DC 12v 100RPM Motor | M1 | 1 | $14.99 | Amazon | Greartisan | | $14.99 | |
| CR1220 Battery | U5 | 1 | $4.99 | Fred Meyer | Energizer | CR1220 | $4.99 | |
| L298N Dual Full Bridge Driver | U4 | 1 | $6.99 | Amazon | Qunqi | L298n | $6.99 | |
| 2 GB SD Flash Memory Card | U2 | 1 | $8.39 | Amazon | Transcend | | $8.39 | |
| RGB LED | U3 | 3 | $1.50 | Resistore | NTE Electronics | NTE30115 | $4.50 | |
| PNP Channel | U3 | 3 | $1.04 | Mouser | Infineon | IPAN70R750P7S | $3.12 | |
| 1/4 W Resistors | U3 | 1 | $12.99 | Amazon | Aukenien | AKN-1/4WRK | $12.99 | |
| PCB | U3 | 1 | $85 | OSHPark | OSHPark | | $85.00 | $179.35 |

# ECE 342 Electronic Pet Door Group 7

Oregon State University ECE
Beth, Blake, and Ben

Project Start: Thu, 12/30/2021
Display Week: 1

| TASK | ASSIGNED TO | TIME TAKEN TO COMPLETE | PROGRESS | START | END |
|---|---|---|---|---|---|
| **Design Phase** | | 11.25 hrs | | | |
| Organize meeting schedule | All | .25 hrs | 100% | 12/30/21 | 1/9/22 |
| Create Code Flow Chart | Ben | 1.5 hrs | 100% | 1/10/22 | 1/15/22 |
| Initial block diagram | All | 1.5 hrs | 100% | 1/9/22 | 1/10/22 |
| Research Components to use | All | 2.5 hrs | 100% | 1/10/22 | 1/15/22 |
| Decide on components and order | All | 1.5 hrs | 100% | 1/14/22 | 1/19/22 |
| Design interface connections | Blake | 2 hrs | 100% | 1/14/22 | 1/24/22 |
| Power supply design | Beth | 1 hrs | 100% | 1/10/22 | 1/20/22 |
| **Create Phase** | | 44 hrs | | | |
| uC Detects RFID Tags and Correct Time | Blake | 4 hrs | 100% | 1/21/22 | 1/28/22 |
| LEDs Light Up According to uC | Beth | 6 hrs | 100% | 1/28/22 | 2/4/22 |
| Motor Controller Lifts Door | Ben | 5 hrs | 100% | 1/28/22 | 2/4/22 |
| uC Talks to Motor Controller | Ben, Blake | 4 hrs | 100% | 2/4/22 | 2/11/22 |
| uC Receives input from RFID Tags, Time module and mo | Everyone | 10 hrs | 100% | 2/11/22 | 2/18/22 |
| System Runs off of independent power source (no comp | Beth | 3 hrs | 100% | 2/18/22 | 2/25/22 |
| Each engineering requirement is met | Everyone | 12 hrs | 100% | 2/28/22 | 3/4/22 |
| **Present Phase** | | 14 hrs | | | |
| System Verification | Everyone | 10 hrs | 50% | 2/25/22 | 3/4/22 |
| Create presentation board | Everyone | 1.5 hrs | 100% | 3/4/22 | 3/8/22 |
| Engineering Expo /Project Demonstration Preparation | Everyone | 2.5 hrs | 0% | 3/4/22 | 3/11/22 |

*Insert new rows ABOVE this one*