ACCESS CONTROL SYSTEM

Junior Design Project

CSE 398

M002

Chikeluba Anierobi, Raffi Mendoza, Peter Spinelli

May 2022

TABLE OF CONTENTS

INTRODUCTION

Our junior design project is the implementation of an Access Control System (basically an Attendance System with an additional layer of security for our design labs). Many times students leave the doors unlocked which leaves a possibility that anyone regardless of their authority could have the ability to enter. On top of that, we noticed that Dr. Marcy's current method of taking attendance is by doing a manual headcount every few minutes at the beginning of class. This attendance system would require only students authorized in the labs to enter and make the process of taking attendance automated by exporting all records to a csv file.

Our embedded system consists of a **Raspberry Pi Zero W, an RFID-RC522 Module, multiple RFID tags, 16x2 LCD screen. and a red/green LED**. Since we have used some of the components from previous projects which we are able to reuse, and additionally found 10 RFID cards containing numeric data with 13.56mhz frequencies needed for the RC522; All that was left was the RC522 reader itself (Purchase Link: https://www.amazon.com/%C2%AEMifare-Antenna-Proximity-13-56mhz-Arduino/dp/B00UBE1JW).

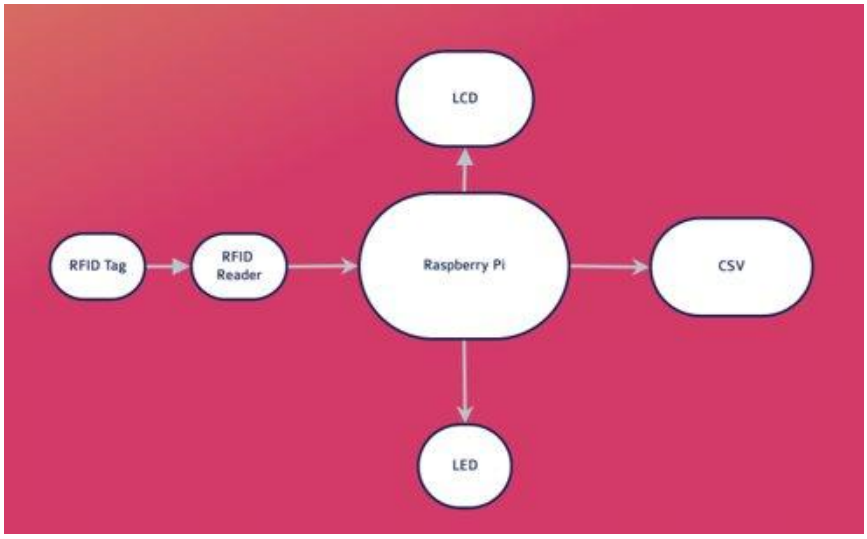


Figure: Block Diagram of our Hardware Setup

Although we have a few hardware components, our access control system would satisfy various real world applications such as: improving security, taking attendance efficiently, and reducing micromanaging by class instructors and professors.

Potential negatives that surfaced during our planning phase were that we would have to issue IDs separate from students' normal school IDs (Syracuse University IDs) which would be costly if lost or stolen. Furthermore, having the program continuously run outside of lab times to manage attendance could become tricky (the circuit could become faulty or intruders could tamper with the board easily). To mitigate these negative impacts, we would have to enforce strict health & safety policies to all students in labs and hold them accountable for any violation.

For our implementation, our first few demonstrations would be simple to form as we can reuse our previous C++ GPIO and LCD classes to enable the display and LEDs. What is left is the RFID reader which we would have to solder pins to in order to connect to our breadboard. Unlike Python, C++ does not have an RC522 library that would easily configure the device. We

managed to find a class library on github that would configure it: https://github.com/paguz/RPi-RFID. Afterwards, we created three simple string arrays that have corresponding data of an RFID based on the pointer value (names, dates of birth and ID numbers of authorized cardholders). We also added time stamps so the instructor can know when the authorized cardholder was granted access. Our final goal is to append these records into the default csv (comma separated value) file, and take attendance by updating the file name to the current date. For instance, the instructor can obtain the csv file for May2nd.csv, and rename it as that. Next class, the instructor can rename the csv file as May4th.csv and so on.

IMPLEMENTATION DESIGN

Day 1: Print “Hello World” on LCD

We plan to implement an LCD screen to print the authority level when a particular cardholder taps their card/tag on the card reader. Therefore, we sought to get this to work by interfacing the LCD screen to print the string "Hello World" on the screen.

Day 2: Make green/red LED lights blink

We would be using the LEDs (Light Emitting Diodes) to indicate when the cardholder is authorized (green) or unauthorized (red). We can do this by assigning two GPIO pins to both LEDs and setting their direction as Output. In order to make the LEDs blink, we just set them to 2 values: 0 and 1, and added a usleep (a method to delay) to make the blink quite noticeable.

Day 3: Configuring RFID-RC522 card reader to read ID # from RFID tag

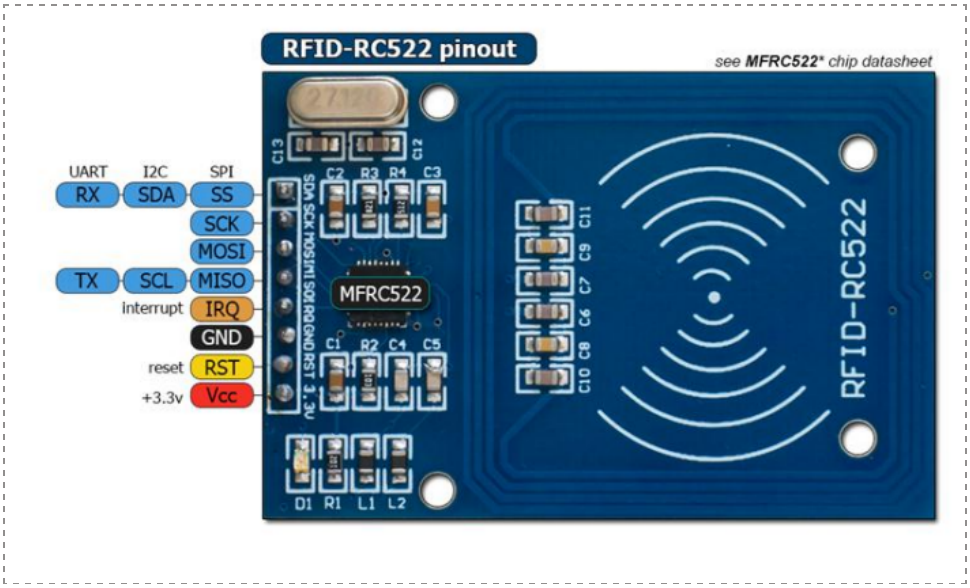


Figure: RFID-RC522 CARD READER PINOUT

In order to find out if the reader was properly configured, we tapped the tag against the reader and had Raspberry Pi print the data on the LCD screen (which would be demonstrated on the next signature).

Day 4: Print ID numbers of cardholders on LCD Screen

As follow up to the previous signature, we programmed the LCD screen to display the ID numbers of all cardholders.

Day 5: Print authority level of cardholders on LCD Screen

Much like the previous signature, we now program the LCD screen to display the authority levels of all cardholders, authorized or unauthorized.

Day 6: Export data of cardholder to “records.csv”

To carry out this signature, we would first make sure that the data of each cardholder was printed on the Raspberry Pi Terminal. It is worth noting that only authorized cardholders have existing data in the system (name and date of birth). Since unauthorized cardholders are not in the system, the only data that is allotted to them is their ID number.

Day 7: Take attendance by renaming “records.csv” to “[CurrentDate].csv”

Given the main goal of the project, this last implementation would be to get each records.csv file containing data for that particular day and rename in accordance with the day that cardholders accessed the system.

ETHICAL JUSTIFICATION

Problem solving is at the core of engineering: As engineers, we are prompted to think critically and solve problems by implementing new technological principles and advancements. By nature, an access control system is meant to uphold and encourage the basic principle of security. By hindering unauthorized access to certain infrastructure, we are enforcing the security of those who are authorized. In many scenarios, it is likely that potential threats to an organization or educational facility such as thieves or those looking to harm the organization's members will come from outside that organization. Our system ensures that the only people allowed to access said facility are those who come from inside the organization. By assigning the information of employees or students to their RFID access cards, we can verify the identities of each individual who enters the building. Furthermore, our record of who has both been granted and unsuccessfully attempted access to the building created by our security system helps the organization identify potential threats and improve upon their security in other areas as well. All of these aspects greatly reduce the likelihood of a potential outside attack, thus ensuring the safety and wellbeing of both the company/educational body and its members.

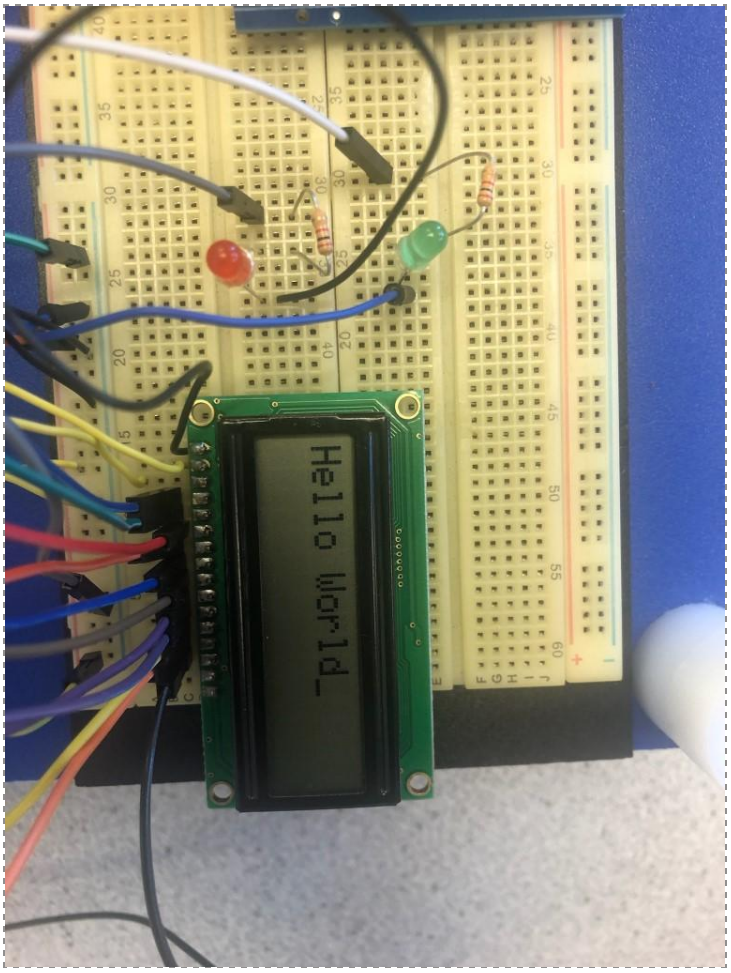
Santa Clara University's *A Framework for Ethical Decision Making* identifies in its definition of Ethics that "ethics is **not** following the law." The distinction being made here is that while a good set of laws encompasses many ethical principles, the law does not define what ethical standards are. It goes on to state that systems of law can become ethically corrupt. This means that the law cannot be relied on completely to uphold ethical standards, which implies that to some degree, corporate, educational, and individual security is partly the responsibility of the individual body itself. With the knowledge that some aspects of security lie in the hands of the citizens, our system aids individual entities in upholding their own security. We understand that

government systems of protection, like the police, cannot be relied upon for 24/7 protection. Our embedded system is more accessible in that it can be added on to any building's existing security system, and it is faster in its ability to abate potential threats than the police are.

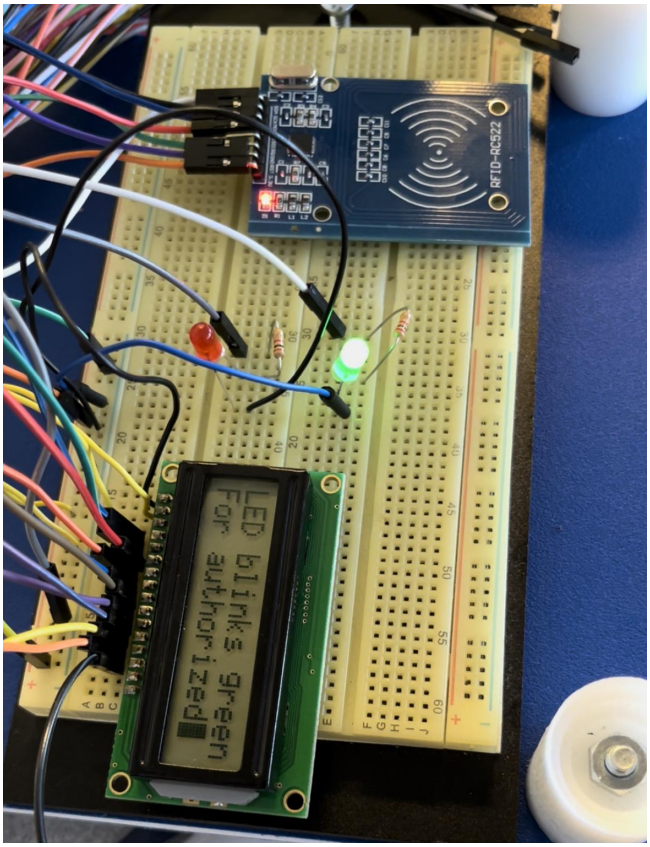
On a more general level, the IEEE Code of Ethics emphasizes in its very first section the importance of upholding public welfare; "to hold paramount the safety, health, and welfare of the public, to strive to comply with ethical design and sustainable development practices, to protect the privacy of others, and to disclose promptly factors that might endanger the public or the environment…" . At the core of the system we designed is concern for general safety and welfare of the public. By allowing those who would employ our system to take security measures on their own and protect themselves from potential threats, we are encouraging the spread of public safety and general wellbeing. Additionally, our system perpetuates the standard of privacy emphasized in the IEEE Code of Ethics. By enforcing security in the classroom and in corporations, any information intended to be confidential by our client entities would be protected by keeping out anyone with intent to steal or corrupt that information.

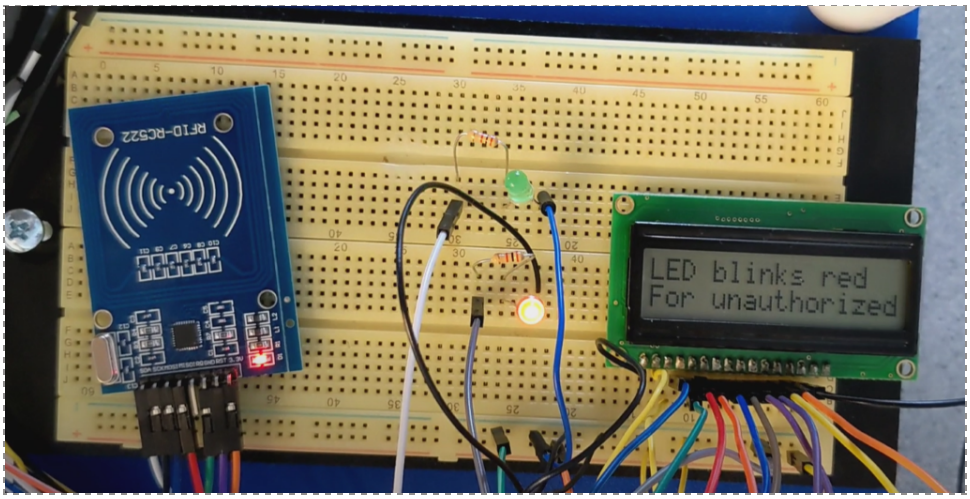
RESULTS

Day 1: Print “Hello World” on LCD

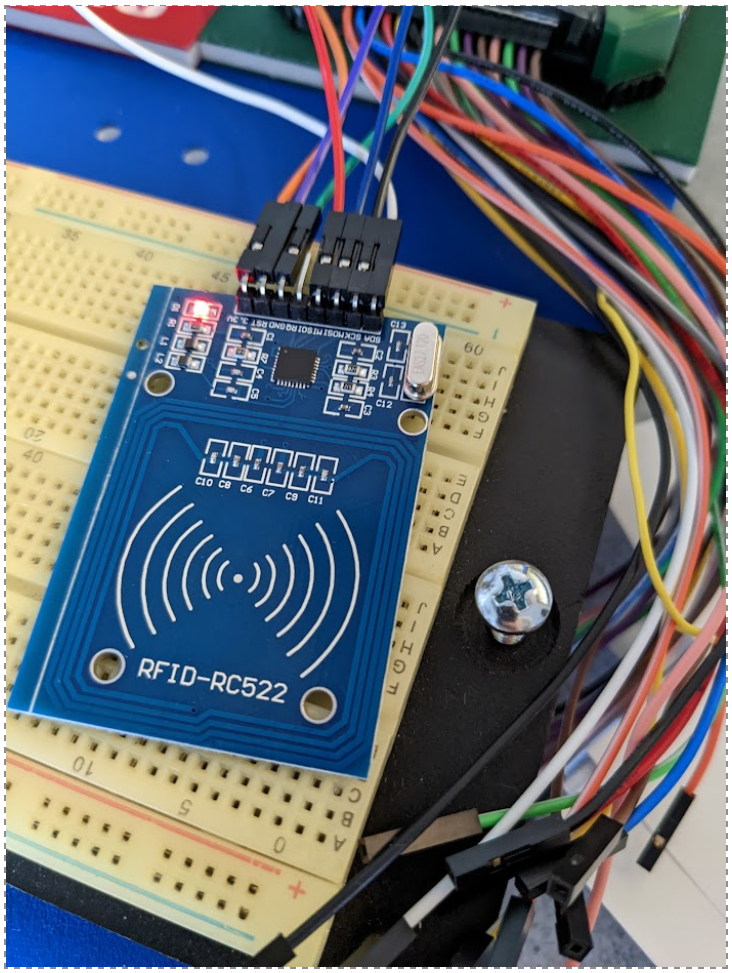


Day 2: Make green/red LED lights blink

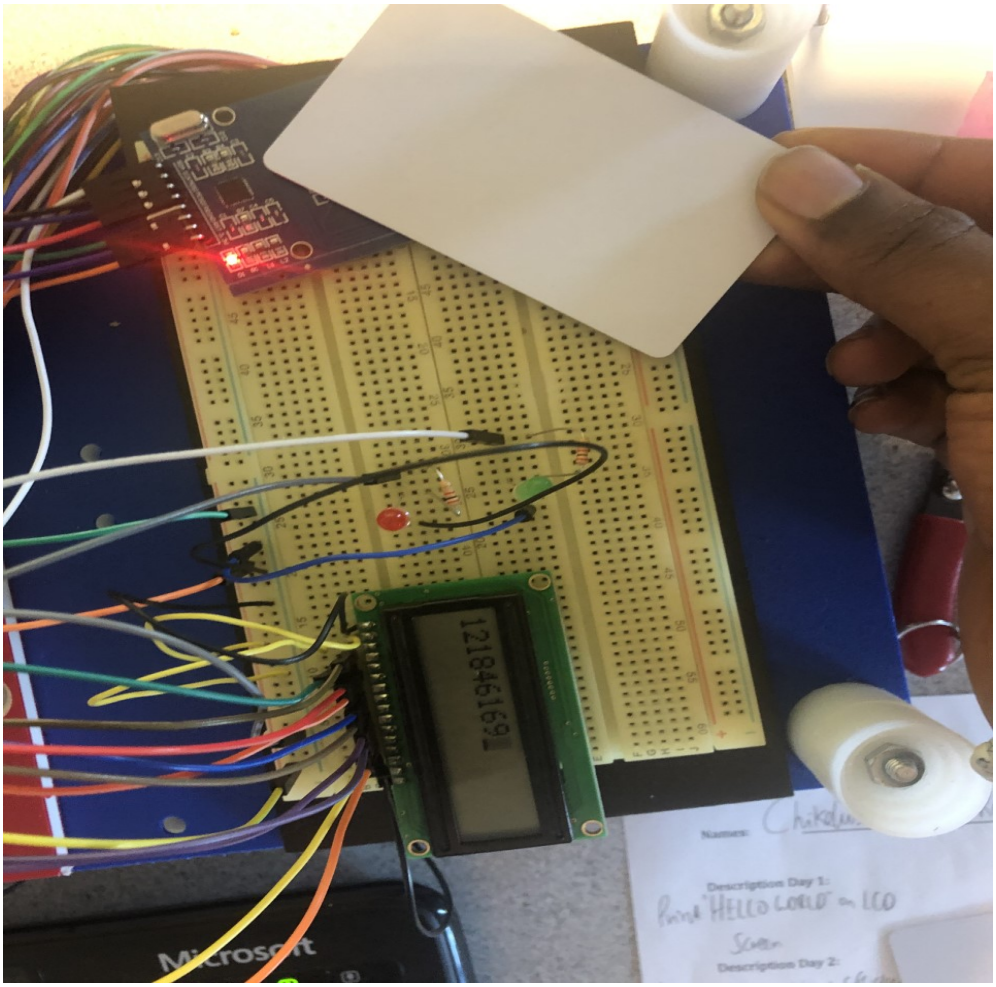




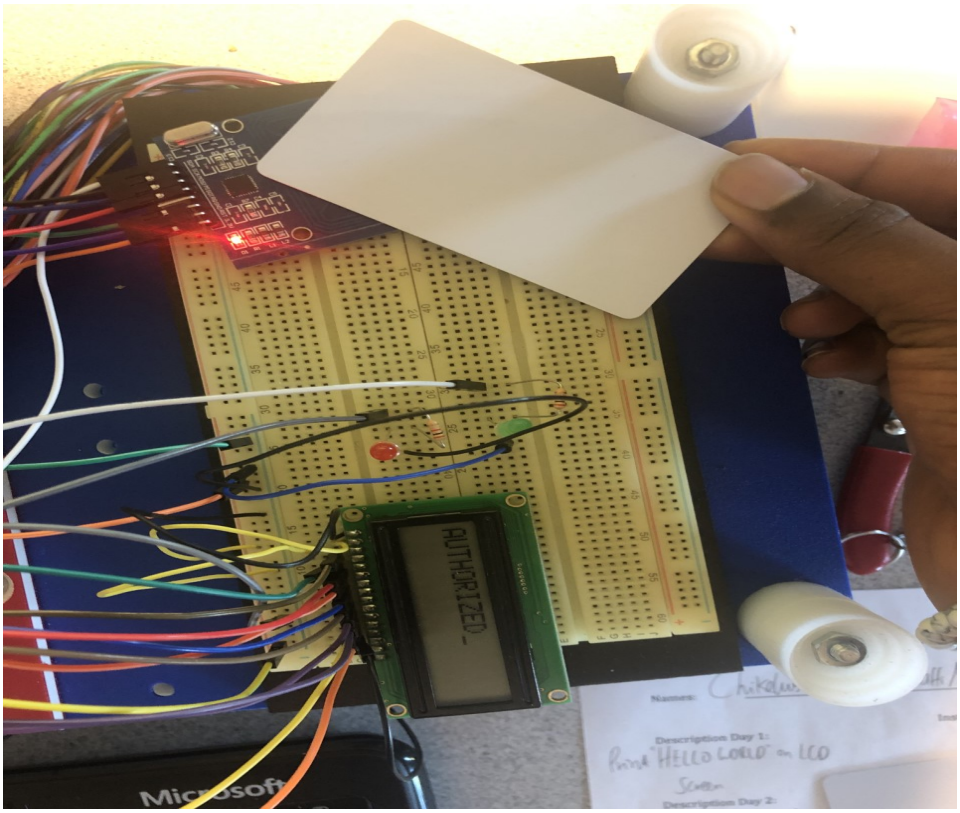
Day 3: Configuring RFID-RC522 card reader to read ID # from RFID tag



Day 4: Print ID numbers of cardholders on LCD Screen



Day 5: Print authority level of cardholders on LCD Screen



Day 6: Export data of cardholder to “records.csv”

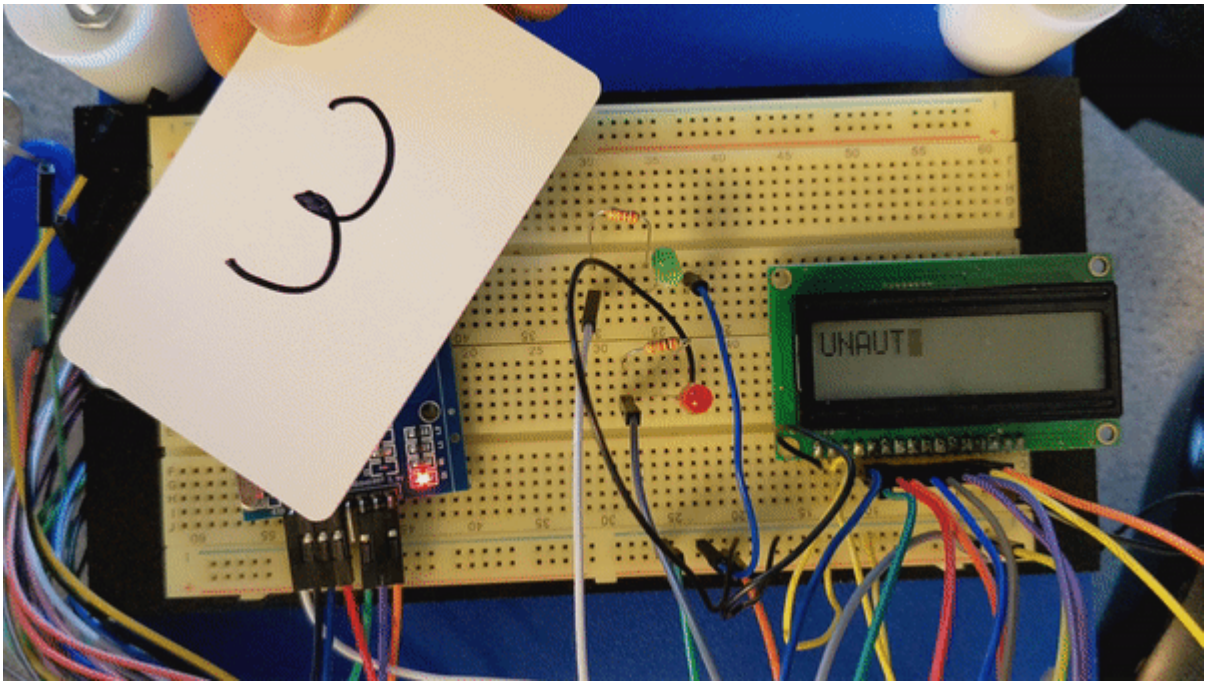
1.0818E+11	Jane Doe	4/4/00	AUTHORIZED	Mon May 2 12:51:47 2022
1.0818E+11	Jane Doe	4/4/00	AUTHORIZED	Mon May 2 13:40:17 2022
1.0818E+11	Jane Doe	4/4/00	AUTHORIZED	Mon May 2 13:40:41 2022
1.0818E+11	Jane Doe	4/4/00	AUTHORIZED	Mon May 2 13:40:46 2022
121846169	Solana Rowe	1/9/00	AUTHORIZED	Mon May 2 13:40:50 2022
121846169	Solana Rowe	1/9/00	AUTHORIZED	Mon May 2 13:40:55 2022
121175769	Hykeem Cart	11/11/99	AUTHORIZED	Mon May 2 13:41:00 2022
2810110262	N/A	N/A	UNAUTHORIZ	Mon May 2 13:41:05 2022
121846169	Solana Rowe	1/9/00	AUTHORIZED	Mon May 2 13:41:16 2022
1.0818E+11	Jane Doe	4/4/00	AUTHORIZED	Mon May 2 13:41:19 2022
121175769	Hykeem Cart	11/11/99	AUTHORIZED	Mon May 2 13:41:24 2022
2810110262	N/A	N/A	UNAUTHORIZ	Mon May 2 13:41:29 2022
2810110262	N/A	N/A	UNAUTHORIZ	Mon May 2 13:41:32 2022
2810110262	N/A	N/A	UNAUTHORIZ	Mon May 2 13:41:35 2022
2810110262	N/A	N/A	UNAUTHORIZ	Mon May 2 13:43:01 2022
2810110262	N/A	N/A	UNAUTHORIZ	Mon May 2 13:43:12 2022
2810110262	N/A	N/A	UNAUTHORIZ	Mon May 2 13:43:28 2022
2810110262	N/A	N/A	UNAUTHORIZ	Mon May 2 13:44:02 2022
2810110262	N/A	N/A	UNAUTHORIZ	Mon May 2 13:44:15 2022
121175769	Hykeem Cart	11/11/99	AUTHORIZED	Mon May 2 13:44:24 2022
1.0818E+11	Jane Doe	4/4/00	AUTHORIZED	Mon May 2 13:45:04 2022

Day 7: Take attendance by renaming “records.csv” to “[CurrentDate].csv”

5/02/2022.csv

1.08175E+11	Jane Doe	4/4/00	AUTHORIZED	Mon	May	2	12:51:47	2022
1.08175E+11	Jane Doe	4/4/00	AUTHORIZED	Mon	May	2	13:40:17	2022
1.08175E+11	Jane Doe	4/4/00	AUTHORIZED	Mon	May	2	13:40:41	2022
1.08175E+11	Jane Doe	4/4/00	AUTHORIZED	Mon	May	2	13:40:46	2022
121846169	Solana Rowe	1/9/00	AUTHORIZED	Mon	May	2	13:40:50	2022
121846169	Solana Rowe	1/9/00	AUTHORIZED	Mon	May	2	13:40:55	2022
121175769	Hykeem Carter	11/11/99	AUTHORIZED	Mon	May	2	13:41:00	2022
2810110262	N/A	N/A	UNAUTHORIZED	Mon	May	2	13:41:05	2022

Final Demonstration



CONCLUSION

After completing the project, we found that implementing both the hardware and software were smooth and intuitive. Since we were able to reuse existing code, we were basically just adding features that satisfy each of our demonstrations. After our final demonstration, a good question asked by the audience is what we could do in regards to running the system outside of class hours. As mentioned in our introduction, this would require all students to have a contract agreement, making them liable for any property damage or theft. Our current program is also limited to updating attendance manually: the record stored for the day has to be done by an instructor, which could become inefficient since multiple instructors use the lab. We could mitigate this by automating the filename change by the end of each day (the system could be running from 6am-9pm daily) and storing them in their respective month directory. We also found that our program's runtime might be slower for a larger database of students, since we are only doing a linear search on an array. Maybe in the future we could implement a SQL Database (DB) or we could just transition to a HashMap (since our database is not too complex) which would reduce search to constant time. Although we had the foundational aspects for our access control system working accurately, there is always room for additional features and scalability.

APPENDIX

Day 1: Print “Hello World” on LCD

gpio.cpp

```
#include "gpio.h" //we include the header file to the cpp
#include <fstream> //this library will allow us to open files
#include <string>
#include <iostream>

using namespace std;

gpio::gpio(int a_pin_num){
   pin_num = a_pin_num;
   ofstream pin_file;
   string location = "/sys/class/gpio/export"; //set the pin number for the GPIO we are going to use
   pin_file.open(location);
   pin_file << pin_num;
   pin_file.close(); //close the file
}

gpio::~gpio(){
   set_val(0);
   set_dir("in");
   string location = "/sys/class/gpio/unexport";
   ofstream pin_file; //open file unexport
   pin_file.open(location);
   pin_file << pin_num;
   pin_file.close();
}


void gpio::set_dir(string dir){
   ofstream pin_file;
   string location = "/sys/class/gpio/gpio" + to_string(pin_num) + "/direction";
   pin_file.open(location);
   pin_file << dir; //set the direction of the corresponding GPIO (in or out)
   pin_file.close(); //close the file
}

void gpio::set_val(int val){
   ofstream pin_file;
   string location = "/sys/class/gpio/gpio" + to_string(pin_num) + "/value";
```

```
    pin_file.open(location);
    pin_file << val; //set the value of the corresponding GPIO (0 or 1)
    pin_file.close(); //close the file
}

int gpio::get_val(){
    int val;
    ifstream pin_file;
    string location = "/sys/class/gpio/gpio" + to_string(pin_num) + "/value";
    pin_file.open(location);
    pin_file >> val; //assign the value of the GPIO to the variable int val
    pin_file.close(); //close the file
    return val; //return the value of the GPIO
}
```

gpio.h

```
#ifndef GPIO_H
#define GPIO_H
#include <iostream>
using namespace std;

class gpio{
private:
    int pin_num; //private variable that can only be accessed by a GPIO object

public:
    gpio(int a_pin_num); //constructor that creates the GPIO pin necessary
    ~gpio(); //destructor
    int get_val(); //getter that returns the value of the GPIO pin
    void set_val(int val); //setter that set the value of the GPIO pin
    void set_dir(string dir); //setter that sets the direction of the GPIO pin (in or out)
};
#endif
```

LCD.cpp

```
#include "LCD.h"

#include "gpio.h"

#include <iostream>
#include <string>
#include <unistd.h> //this library will allow us to place time delayes

using namespace std;
```

```
 LCD::LCD(int a_E, int a_RS, int a_RW, int a_D0, int a_D1, int a_D2, int a_D3, int a_D4, int a_D5, int
a_D6, int a_D7){

    //Allocating the space for each pointer object
    E = new gpio(a_E);
    RS= new gpio(a_RS);
    RW= new gpio(a_RW);
    D0= new gpio(a_D0);
    D1= new gpio(a_D1);
    D2= new gpio(a_D2);
    D3= new gpio(a_D3);
    D4= new gpio(a_D4);
    D5= new gpio(a_D5);
    D6= new gpio(a_D6);
    D7= new gpio(a_D7);

    //Setting the direction for each gpio each object is pointing to
    E->set_dir("out");
    RS->set_dir("out");
    RW->set_dir("out");
    D0->set_dir("out");
    D1->set_dir("out");
    D2->set_dir("out");
    D3->set_dir("out");
    D4->set_dir("out");
    D5->set_dir("out");
    D6->set_dir("out");
    D7->set_dir("out");
    //Setting the default value for each gpio each object is pointing to
    E->set_val(0);
    RS->set_val(0);
    RW->set_val(0);
    D0->set_val(0);
    D1->set_val(0);
    D2->set_val(0);
    D3->set_val(0);
    D4->set_val(0);
    D5->set_val(0);
    D6->set_val(0);
    D7->set_val(0);
}
```

```
LCD::~LCD(){
   delete E;
   delete RS;
   delete RW;
   delete D0;
   delete D1;
   delete D2;
   delete D3;
   delete D4;
   delete D5;
   delete D6;
   delete D7;
}



void LCD::command(int command){
   //setting the values for each gpio the objects are pointing to
   RS->set_val(0);
   RW->set_val(0);
   //setting a bitwise function for each gpio and then moving the bit to the leftmost place
   D0->set_val(command & 1);
   D1->set_val((command & 2) >> 1);
   D2->set_val((command & 4) >> 2);
   D3->set_val((command & 8) >> 3);
   D4->set_val((command & 16) >> 4);
   D5->set_val((command & 32) >> 5);
   D6->set_val((command & 64) >> 6);
   D7->set_val((command & 128) >> 7);
   usleep(100);
   E->set_val(1); //turning on enable
   usleep(100);
   E->set_val(0); //turning off enable
   usleep(10000); //putting a delay so there is enough time for the next instructions to run.
}



void LCD::print(string str){
//in order to iterate through the string that has been passed and get the value for each character, to later
pass that value to the corresponding gpio pin, we used a for loop
   for(int i = 0; i < str.length(); i++){
      char l_char = str.at(i);
      int n = (int)l_char; //casting the character into int
```

```
        usleep(40000);
        RS->set_val(1); //for  printing into the LCD screen, RS value has to be 1
        RW->set_val(0);
        //setting a bitwise function for each gpio and then moving the bit to the leftmost place
        D0->set_val(n & 1);
        D1->set_val((n & 2) >> 1);
        D2->set_val((n & 4) >> 2);
        D3->set_val((n & 8) >> 3);
        D4->set_val((n & 16) >> 4);
        D5->set_val((n & 32) >> 5);
        D6->set_val((n & 64) >> 6);
        D7->set_val((n & 128) >> 7);
        usleep(10000);
        E->set_val(1); //turning on enable
        usleep(10000);
        E->set_val(0); //turning off enable
        usleep(10000); //putting a delay so there is enough time for the next instructions to run.
    }
}

void LCD::initialize(){ //This functions is passing the commands to initialize the LCD screen
    cout << "Initialized" << endl;
    usleep(40000);
    command(0x38); //Function set: 8 bits, 2 lines, and 5x8 font size
    command(0x38); //Function set: 8 bits, 2 lines, and 5x8 font size
    command(0x0F); //Display ON
    command(0x01); //Clear the display
    command(0x06); //Entry mode: cursor/blink moves to the right and DDRAM address is increased by
1, and no shift display
}
```

LCD.h

```
#ifndef LCD_H
#define LCD_H
#include <string>
#include "gpio.h" //the gpio header file needs to be addes in order to create gpio objects and access to
gpio functions.
#include <iostream>
class LCD{
    private:
        gpio * E, * RS, * RW, * D0, * D1, * D2, * D3, * D4, * D5, * D6, * D7; //private gpio pointer objects

    public:
```

```
	LCD(int a_E, int a_RS, int a_RW, int a_D0, int a_D1, int a_D2, int a_D3, int a_D4, int a_D5, int a_D6, int a_D7); //constructor with int parameters
	~LCD(); //destructor

	void command(int command); //command function with a int parameter
	void print(string str); //print function with string parameter
	void initialize(); //initialize function with no parameters
};
#endif
```

main.cpp

```
#include <iostream>
#include "gpio.h" //inlcude the  gpio header file to the main function to create objects from its gpio class
#include "LCD.h"  //include the LCD header file to create objects from its LCD class
#include <unistd.h>
#include <fcntl.h>
#include <linux/i2c-dev.h>
#include <unistd.h>
#include <ostream>

using namespace std;
int main(){
LCD LCD_1(22, 17, 27, 24, 25, 5, 6, 13, 12, 16, 26);

LCD_1.initialize();
LCD_1.print("Hello World");
LCD_1.command(0X01); //clears the screen
LCD_1.command(0X80); //brings cursor back to original position
while (1) {}
return 0;
}
```

Day 2: Make green/red LED lights blink

gpio* LCD*

main.cpp

```
#include <iostream>
#include "gpio.h" //inlcude the  gpio header file to the main function to create objects from its gpio class
#include "LCD.h"  //include the LCD header file to create objects from its LCD class
#include <unistd.h>
#include <fcntl.h>
#include <linux/i2c-dev.h>
#include <unistd.h>
```

```
#include <ostream>

using namespace std;
int main(){
LCD LCD_1(22, 17, 27, 24, 25, 5, 6, 13, 12, 16, 26);
gpio green(0);  //GREEN LED
gpio red(1);  //RED LED

green.set_dir("out");
red.set_dir("out");

LCD_1.print("LED blinks green");
LCD_1.command(0XC0); //shifts cursor to next line
LCD_1.print("For authorized");
usleep(700000);
green.set_val(1); //prompts the GREEN LED to light up
usleep(100000);
green.set_val(0); //prompts the GREEN LED to turn off

usleep(100000);

LCD_1.command(0X01);
LCD_1.command(0X80);
LCD_1.print("LED blinks red");
LCD_1.command(0XC0);
LCD_1.print("For unauthorized");
usleep(700000);
red.set_val(1); //prompts the RED LED to light up
usleep(100000);
red.set_val(0); //prompts the RED LED to turn off
usleep(100000);

LCD_1.command(0X01); //clears the screen
LCD_1.command(0X80); //brings cursor back to original position
while (1) {}
return 0;
}
```

Day 3: Configuring RFID-RC522 card reader to read ID # from RFID tag

gpio* LCD* MFRC522.cpp* bcm2835.h*

main.cpp

```
#include <iostream>
#include "gpio.h" //inlcude the  gpio header file to the main function to create objects from its gpio class
#include "LCD.h"  //include the LCD header file to create objects from its LCD class
#include <unistd.h>
#include <fcntl.h>
#include <linux/i2c-dev.h>
#include <errno.h>
#include <fstream>
#include <stdio.h>
#ifdef WIN32
#include <windows.h>
#else
#include <unistd.h>
#endif
#include <ostream>
#include "MFRC522.h"
#include <chrono>
#include <ctime>

using namespace std;

void delay(int ms){
#ifdef WIN32
  Sleep(ms);
#else
  usleep(ms*1000);
#endif
}


int main(){

LCD LCD_1(22, 17, 27, 24, 25, 5, 6, 13, 12, 16, 26);

gpio green(0);  //GREEN LED
gpio red(1);  //RED LED



green.set_dir("out");
red.set_dir("out");
LCD_1.initialize();
```

```
LCD_1.print("Place Card");
LCD_1.command(0XC0); //Shifts cursor to next line
LCD_1.print("For attendance");

usleep(700000);

LCD_1.command(0X01); //clears the screen
LCD_1.command(0X80); //brings cursor back to original position
LCD_1.print("LED blinks green");
LCD_1.command(0XC0); //shifts cursor to next line
LCD_1.print("For authorized");
usleep(700000);
green.set_val(1); //prompts the GREEN LED to light up
usleep(100000);
green.set_val(0); //prompts the GREEN LED to turn off

usleep(100000);

LCD_1.command(0X01);
LCD_1.command(0X80);
LCD_1.print("LED blinks red");
LCD_1.command(0XC0);
LCD_1.print("For unauthorized");
usleep(700000);
red.set_val(1); //prompts the RED LED to light up
usleep(100000);
red.set_val(0); //prompts the RED LED to turn off

usleep(100000);

LCD_1.command(0X01); //clears the screen
LCD_1.command(0X80); //brings cursor back to original position
LCD_1.print("Please proceed");

usleep(700000);

LCD_1.command(0X01); //clears the screen
LCD_1.command(0X80); //brings cursor back to original position
MFRC522 mfrc; //instantiates an object "mfrc" from the MFRC522 class, which allows us to setup the
card reader
mfrc.PCD_Init();
cout << "Place Tag" << endl;
while(1){
```

```
        // Look for a card
   string userID = "";
   if(!mfrc.PICC_IsNewCardPresent())
    continue;

   if( !mfrc.PICC_ReadCardSerial())
    continue;
   //cout << "CARD DATA: ";
   for(byte i = 0; i < mfrc.uid.size; ++i){
    if(mfrc.uid.uidByte[i] < 0x10){
     //printf("%d",mfrc.uid.uidByte[i]);
     userID+=to_string(mfrc.uid.uidByte[i]);
    }
    else{

     //printf("%d", mfrc.uid.uidByte[i]);
     userID+=to_string(mfrc.uid.uidByte[i]);
    }

   }
        // Demo 3
cout << userID << endl;

   cout << endl;
   bool found = false;
   int userIndex = 0;
   for (int i = 0; i < 5; i++){
            if (userID == authorized[i]){
                found = true;
                userIndex = i;
                break;
            }
        }
  }
return 0;
}
```

Day 4: Print ID numbers of cardholders on LCD Screen

gpio* LCD*

main.cpp

```
#include <iostream>
#include "gpio.h" //inlcude the  gpio header file to the main function to create objects from its gpio class
#include "LCD.h"  //include the LCD header file to create objects from its LCD class
#include <unistd.h>
#include <fcntl.h>
#include <linux/i2c-dev.h>
#include <errno.h>
#include <fstream>
#include <stdio.h>
#ifdef WIN32
#include <windows.h>
#else
#include <unistd.h>
#endif
#include <ostream>
#include "MFRC522.h"
#include <chrono>
#include <ctime>

using namespace std;

void delay(int ms){
#ifdef WIN32
  Sleep(ms);
#else
  usleep(ms*1000);
#endif
}


int main(){

LCD LCD_1(22, 17, 27, 24, 25, 5, 6, 13, 12, 16, 26);

gpio green(0);  //GREEN LED
gpio red(1);  //RED LED



green.set_dir("out");
red.set_dir("out");
LCD_1.initialize();
```

```
LCD_1.print("Place Card");
LCD_1.command(0XC0); //Shifts cursor to next line
LCD_1.print("For attendance");

usleep(700000);

LCD_1.command(0X01); //clears the screen
LCD_1.command(0X80); //brings cursor back to original position
LCD_1.print("LED blinks green");
LCD_1.command(0XC0); //shifts cursor to next line
LCD_1.print("For authorized");
usleep(700000);
green.set_val(1); //prompts the GREEN LED to light up
usleep(100000);
green.set_val(0); //prompts the GREEN LED to turn off

usleep(100000);

LCD_1.command(0X01);
LCD_1.command(0X80);
LCD_1.print("LED blinks red");
LCD_1.command(0XC0);
LCD_1.print("For unauthorized");
usleep(700000);
red.set_val(1); //prompts the RED LED to light up
usleep(100000);
red.set_val(0); //prompts the RED LED to turn off

usleep(100000);

LCD_1.command(0X01); //clears the screen
LCD_1.command(0X80); //brings cursor back to original position
LCD_1.print("Please proceed");

usleep(700000);

LCD_1.command(0X01); //clears the screen
LCD_1.command(0X80); //brings cursor back to original position
MFRC522 mfrc; //instantiates an object "mfrc" from the MFRC522 class, which allows us to setup the
card reader
mfrc.PCD_Init();
cout << "Place Tag" << endl;
while(1){
```

```
        // Look for a card
  string userID = "";
  if(!mfrc.PICC_IsNewCardPresent())
    continue;

  if( !mfrc.PICC_ReadCardSerial())
    continue;
  //cout << "CARD DATA: ";
  for(byte i = 0; i < mfrc.uid.size; ++i){
    if(mfrc.uid.uidByte[i] < 0x10){
      //printf("%d",mfrc.uid.uidByte[i]);
      userID+=to_string(mfrc.uid.uidByte[i]);
    }
    else{

      //printf("%d", mfrc.uid.uidByte[i]);
      userID+=to_string(mfrc.uid.uidByte[i]);
    }

  }

  // Demo 4
  LCD_1.print(userID);
  LCD_1.command(0X01);
  LCD_1.command(0X80);
 usleep(1000000);

  cout << endl;
  bool found = false;
  int userIndex = 0;
  for (int i = 0; i < 5; i++){
                if (userID == authorized[i]){
                        found = true;
                        userIndex = i;
                        break;
                }
        }
  }
return 0;
}
```

Day 5: Print authority level of cardholders on LCD Screen

gpio* LCD* MFRC522.cpp* bcm2835.h*

main.cpp

```
#include <iostream>
#include "gpio.h" //inlcude the  gpio header file to the main function to create objects from its gpio class
#include "LCD.h"  //include the LCD header file to create objects from its LCD class
#include <unistd.h>
#include <fcntl.h>
#include <linux/i2c-dev.h>
#include <errno.h>
#include <fstream>
#include <stdio.h>
#ifdef WIN32
#include <windows.h>
#else
#include <unistd.h>
#endif
#include <ostream>
#include "MFRC522.h"
#include <chrono>
#include <ctime>

using namespace std;

void delay(int ms){
#ifdef WIN32
  Sleep(ms);
#else
  usleep(ms*1000);
#endif
}


int main(){

LCD LCD_1(22, 17, 27, 24, 25, 5, 6, 13, 12, 16, 26);

gpio green(0);  //GREEN LED
gpio red(1);  //RED LED



green.set_dir("out");
red.set_dir("out");
LCD_1.initialize();
```

```
LCD_1.print("Place Card");
LCD_1.command(0XC0); //Shifts cursor to next line
LCD_1.print("For attendance");

usleep(700000);

LCD_1.command(0X01); //clears the screen
LCD_1.command(0X80); //brings cursor back to original position
LCD_1.print("LED blinks green");
LCD_1.command(0XC0); //shifts cursor to next line
LCD_1.print("For authorized");
usleep(700000);
green.set_val(1); //prompts the GREEN LED to light up
usleep(100000);
green.set_val(0); //prompts the GREEN LED to turn off

usleep(100000);

LCD_1.command(0X01);
LCD_1.command(0X80);
LCD_1.print("LED blinks red");
LCD_1.command(0XC0);
LCD_1.print("For unauthorized");
usleep(700000);
red.set_val(1); //prompts the RED LED to light up
usleep(100000);
red.set_val(0); //prompts the RED LED to turn off

usleep(100000);

LCD_1.command(0X01); //clears the screen
LCD_1.command(0X80); //brings cursor back to original position
LCD_1.print("Please proceed");


usleep(700000);

LCD_1.command(0X01); //clears the screen
LCD_1.command(0X80); //brings cursor back to original position
MFRC522 mfrc; //instantiates an object "mfrc" from the MFRC522 class, which allows us to setup the
card reader
mfrc.PCD_Init();
cout << "Place Tag" << endl;
```

```
string authorized[5] = {"121846169","921635669","121175769","108175194192","3592123167"};
//array consisting of ID data of authorized cardholders
string names[5] = {"Solana Rowe","Christopher Smith", "Hykeem Carter", "Jane Doe", "Jourdan
Riane"}; //array consisting of names of authorized cardholders
string DOB[5] = {"01/09/2000", "05/12/2001", "11/11/1999" , "04/04/2000" , "12/07/1999"}; //array
consisting of date of birth of authorized cardholders
while(1){

        // Look for a card
  string userID = "";
  if(!mfrc.PICC_IsNewCardPresent())
    continue;

  if( !mfrc.PICC_ReadCardSerial())
    continue;
  //cout << "CARD DATA: ";
  for(byte i = 0; i < mfrc.uid.size; ++i){
    if(mfrc.uid.uidByte[i] < 0x10){
      //printf("%d",mfrc.uid.uidByte[i]);
      userID+=to_string(mfrc.uid.uidByte[i]);
    }
    else{

      //printf("%d", mfrc.uid.uidByte[i]);
      userID+=to_string(mfrc.uid.uidByte[i]);
    }

  }

  cout << endl;
  bool found = false;
  int userIndex = 0;
  for (int i = 0; i < 5; i++){
                if (userID == authorized[i]){
                        found = true;
                        userIndex = i;
                        break;
                }
        }
        if (found == true){
                LCD_1.print("AUTHORIZED");
                usleep(700000);
                green.set_val(1);
                usleep(100000);
```

```
                green.set_val(0);
                LCD_1.command(0X01);
                LCD_1.command(0X80);
        }else if (found == false){
                LCD_1.print("UNAUTHORIZED");
                usleep(700000);
                red.set_val(1);
                usleep(100000);
                red.set_val(0);
                LCD_1.command(0X01);
                LCD_1.command(0X80);
        }
   cout << "**************************" << endl;
   cout << "Place Tag" << endl;
  }
return 0;
}
```

Day 6: Export data of cardholder to “records.csv”

gpio* LCD* MFRC522.cpp* bcm2835.h*

main.cpp

```
#include <iostream>
#include "gpio.h" //inlcude the  gpio header file to the main function to create objects from its gpio class
#include "LCD.h"  //include the LCD header file to create objects from its LCD class
#include <unistd.h>
#include <fcntl.h>
#include <linux/i2c-dev.h>
#include <errno.h>
#include <fstream>
#include <stdio.h>
#ifdef WIN32
#include <windows.h>
#else
#include <unistd.h>
#endif
#include <ostream>
#include "MFRC522.h"
#include <chrono>
#include <ctime>

using namespace std;
```

```
void delay(int ms){
#ifdef WIN32
  Sleep(ms);
#else
  usleep(ms*1000);
#endif
}


int main(){

LCD LCD_1(22, 17, 27, 24, 25, 5, 6, 13, 12, 16, 26);

gpio green(0);  //GREEN LED
gpio red(1);  //RED LED



green.set_dir("out");
red.set_dir("out");
LCD_1.initialize();

LCD_1.print("Place Card");
LCD_1.command(0XC0); //Shifts cursor to next line
LCD_1.print("For attendance");

usleep(700000);

LCD_1.command(0X01); //clears the screen
LCD_1.command(0X80); //brings cursor back to original position
LCD_1.print("LED blinks green");
LCD_1.command(0XC0); //shifts cursor to next line
LCD_1.print("For authorized");
usleep(700000);
green.set_val(1); //prompts the GREEN LED to light up
usleep(100000);
green.set_val(0); //prompts the GREEN LED to turn off

usleep(100000);

LCD_1.command(0X01);
LCD_1.command(0X80);
LCD_1.print("LED blinks red");
LCD_1.command(0XC0);
```

```
LCD_1.print("For unauthorized");
usleep(700000);
red.set_val(1); //prompts the RED LED to light up
usleep(100000);
red.set_val(0); //prompts the RED LED to turn off

usleep(100000);

LCD_1.command(0X01); //clears the screen
LCD_1.command(0X80); //brings cursor back to original position
LCD_1.print("Please proceed");


usleep(700000);

LCD_1.command(0X01); //clears the screen
LCD_1.command(0X80); //brings cursor back to original position
MFRC522 mfrc; //instantiates an object "mfrc" from the MFRC522 class, which allows us to setup the
card reader
mfrc.PCD_Init();
cout << "Place Tag" << endl;
string authorized[5] = {"121846169","921635669","121175769","108175194192","3592123167"};
//array consisting of ID data of authorized cardholders
string names[5] = {"Solana Rowe","Christopher Smith", "Hykeem Carter", "Jane Doe", "Jourdan
Riane"}; //array consisting of names of authorized cardholders
string DOB[5] = {"01/09/2000", "05/12/2001", "11/11/1999" , "04/04/2000" , "12/07/1999"}; //array
consisting of date of birth of authorized cardholders
while(1){

        // Look for a card
  string userID = "";
  if(!mfrc.PICC_IsNewCardPresent())
    continue;

  if( !mfrc.PICC_ReadCardSerial())
    continue;
  //cout << "CARD DATA: ";
  for(byte i = 0; i < mfrc.uid.size; ++i){
    if(mfrc.uid.uidByte[i] < 0x10){
      //printf("%d",mfrc.uid.uidByte[i]);
      userID+=to_string(mfrc.uid.uidByte[i]);
    }
    else{
```

```
      //printf("%d", mfrc.uid.uidByte[i]);
      userID+=to_string(mfrc.uid.uidByte[i]);
     }

   }
         // Demo 4
   LCD_1.print(userID);
         LCD_1.command(0X01);
         LCD_1.command(0X80);
         usleep(1000000);

   time_t now = time(0);
         string currTime = ctime(&now); //Gets the time stamp of when a cardholder taps their card against the reader

   cout << endl;
   bool found = false;
   int userIndex = 0;
   for (int i = 0; i < 5; i++){
                  if (userID == authorized[i]){
                        found = true;
                        userIndex = i;
                        break;
                  }
         }
         if (found == true){
                  ofstream recordsFile("records.csv", ios_base::app); //Allows us to append data, rather than overwrite
                  LCD_1.print("AUTHORIZED");
                  usleep(700000);
                  green.set_val(1);
                  usleep(100000);
                  green.set_val(0);
                  LCD_1.command(0X01);
                  LCD_1.command(0X80);
                  recordsFile << authorized[userIndex] << ",";
                  cout << "ID: " << authorized[userIndex] << endl;
                  recordsFile << names[userIndex] << ",";
                  cout << "AUTHORITY LEVEL: AUTHORIZED" << endl;
                  cout << "NAME: " << names[userIndex] << endl;
                  recordsFile << DOB[userIndex] << ",";
                  cout << "DOB: " << DOB[userIndex] << endl;
                  cout << "CURRENT TIME: " << currTime << endl;
                  recordsFile << "AUTHORIZED" << ",";
```

```
                recordsFile << currTime;
                recordsFile << "\n";
                recordsFile.close(); //closes the file stream
        }else if (found == false){
                ofstream recordsFile("records.csv", ios_base::app); //Allows us to append data, rather
than overwrite
                LCD_1.print("UNAUTHORIZED");
                usleep(700000);
                red.set_val(1);
                usleep(100000);
                red.set_val(0);
                LCD_1.command(0X01);
                LCD_1.command(0X80);
                cout << "ID: " << userID << endl;
                recordsFile << userID << ",";
                cout << "AUTHORITY LEVEL: UNAUTHORIZED" << endl;
                cout << "NAME: N/A" << endl;
                recordsFile << "N/A" << ",";
                cout << "DOB: N/A" << endl;
                cout << "CURRENT TIME: " << currTime << endl;
                recordsFile << "N/A" << ",";
                recordsFile << "UNAUTHORIZED" << ",";
                recordsFile << currTime;
                recordsFile << "\n";
                recordsFile.close(); //closes the file stream
        }
   cout << "**************************" << endl;
   cout << "Place Tag" << endl;
  }
return 0;
}
```

Day 7: Take attendance by renaming “records.csv” to “[CurrentDate].csv”

5/02/2022.csv

1.08175E+11	Jane Doe	4/4/00	AUTHORIZED	Mon	May	2	12:51:47	2022
1.08175E+11	Jane Doe	4/4/00	AUTHORIZED	Mon	May	2	13:40:17	2022
1.08175E+11	Jane Doe	4/4/00	AUTHORIZED	Mon	May	2	13:40:41	2022
1.08175E+11	Jane Doe	4/4/00	AUTHORIZED	Mon	May	2	13:40:46	2022
121846169	Solana Rowe	1/9/00	AUTHORIZED	Mon	May	2	13:40:50	2022
121846169	Solana Rowe	1/9/00	AUTHORIZED	Mon	May	2	13:40:55	2022
121175769	Hykeem Carter	11/11/99	AUTHORIZED	Mon	May	2	13:41:00	2022
2810110262	N/A	N/A	UNAUTHORIZED	Mon	May	2	13:41:05	2022

121846169	Solana Rowe		1/9/00	AUTHORIZED	Mon May 2 13:41:16 2022
1.08175E+11	Jane Doe		4/4/00	AUTHORIZED	Mon May 2 13:41:19 2022
121175769	Hykeem Carter		11/11/99	AUTHORIZED	Mon May 2 13:41:24 2022
2810110262	N/A	N/A	UNAUTHORIZED		Mon May 2 13:41:29 2022
2810110262	N/A	N/A	UNAUTHORIZED		Mon May 2 13:41:32 2022
2810110262	N/A	N/A	UNAUTHORIZED		Mon May 2 13:41:35 2022
2810110262	N/A	N/A	UNAUTHORIZED		Mon May 2 13:43:01 2022
2810110262	N/A	N/A	UNAUTHORIZED		Mon May 2 13:43:12 2022
2810110262	N/A	N/A	UNAUTHORIZED		Mon May 2 13:43:28 2022
2810110262	N/A	N/A	UNAUTHORIZED		Mon May 2 13:44:02 2022
2810110262	N/A	N/A	UNAUTHORIZED		Mon May 2 13:44:15 2022
121175769	Hykeem Carter		11/11/99	AUTHORIZED	Mon May 2 13:44:24 2022
1.08175E+11	Jane Doe		4/4/00	AUTHORIZED	Mon May 2 13:45:04 2022