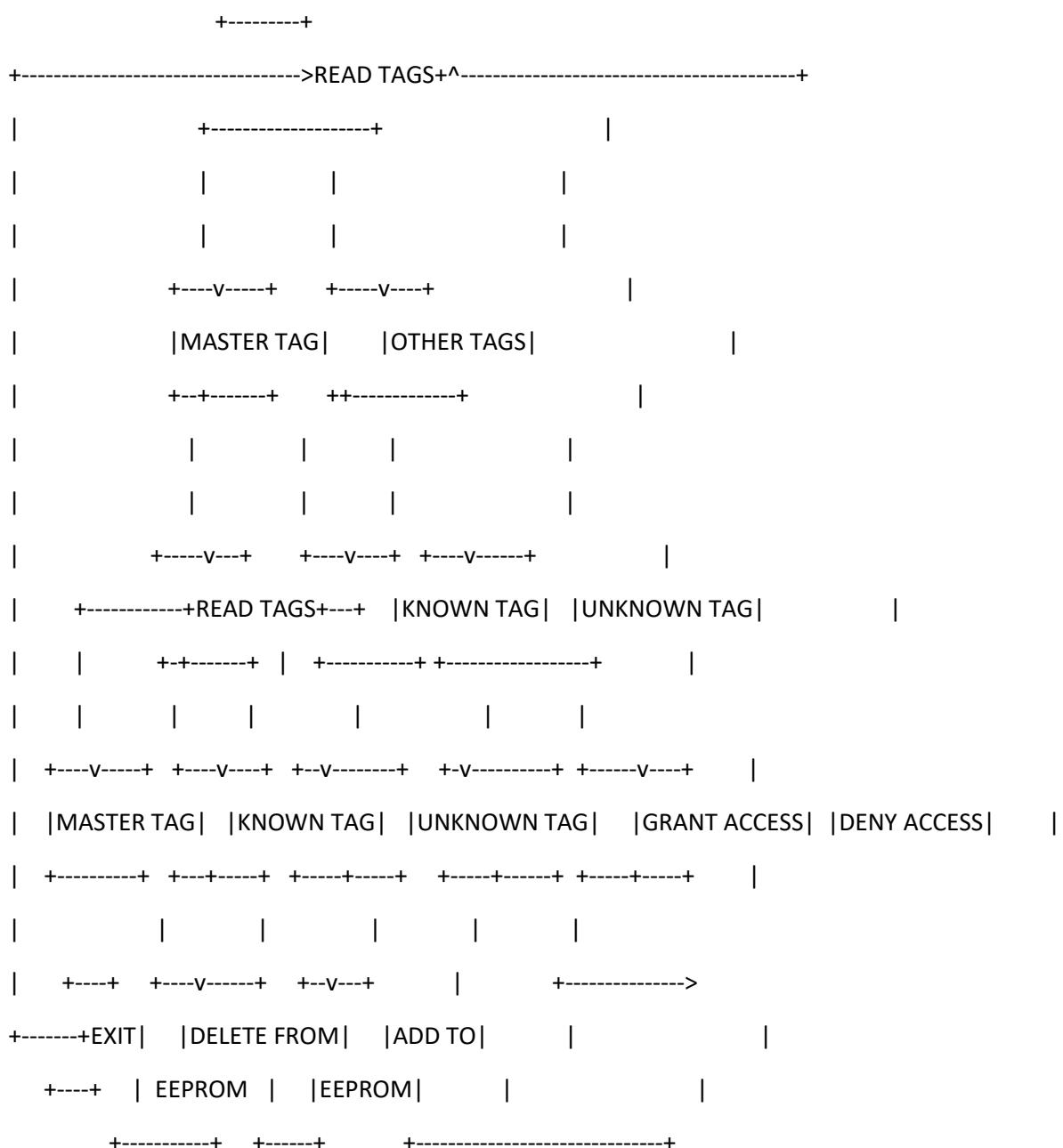


```
/*
```

```
-----  
Example sketch/program showing An Arduino Door Access Control featuring RFID, EEPROM  
-----
```

Door Access Control System

Simple Work Flow:



Use a Master Card which is act as Programmer then you can able to choose card holders who will granted access or not

* **Easy User Interface**

Just one RFID tag needed whether Delete or Add Tags. You can choose to use Leds for output or Serial LCD module to inform users.

* **Stores Information on EEPROM**

Information stored on non volatile Arduino's EEPROM memory to preserve Users' tag and Master Card. No Information lost

if power lost. EEPROM has unlimited Read cycle but roughly 100,000 limited Write cycle.

* **Security**

To keep it simple we are going to use Tag's Unique IDs. It's simple and not hacker proof.

@license Released into the public domain.

Typical pin layout used:

MFRC522	Arduino	Arduino	Arduino	Arduino	Arduino	Arduino
Reader/PCD	Uno/101	Mega	Nano v3	Leonardo/Micro	Pro Micro	
Signal	Pin	Pin	Pin	Pin	Pin	Pin
RST/Reset	RST	9	5	D9	RESET/ICSP-5	RST
SPI SS	SDA(SS)	10	53	D10	10	10
SPI MOSI	MOSI	11 / ICSP-4	51	D11	ICSP-4	16
SPI MISO	MISO	12 / ICSP-1	50	D12	ICSP-1	14

```
SPI SCK    SCK      13 / ICSP-3  52      D13      ICSP-3      15
```

```
*/
```

```
#include <EEPROM.h> // We are going to read and write PICC's UIDs from/to EEPROM  
#include <SPI.h> // RC522 Module uses SPI protocol  
#include <MFRC522.h> // Library for Mifare RC522 Devices
```

```
/*
```

Instead of a Relay you may want to use a servo. Servos can lock and unlock door locks too
Relay will be used by default

```
*/
```

```
// #include <Servo.h>
```

```
/*
```

For visualizing whats going on hardware we need some leds and to control door lock a relay and a wipe button

(or some other hardware) Used common anode led,digitalWriting HIGH turns OFF led Mind that if you are going

to use common cathode led or just seperate leds, simply comment out #define COMMON_ANODE,
*/

```
// #define COMMON_ANODE
```

```
#ifdef COMMON_ANODE  
#define LED_ON LOW  
#define LED_OFF HIGH  
#else  
#define LED_ON HIGH  
#define LED_OFF LOW
```

```

#endif

constexpr uint8_t redLed = 7; // Set Led Pins
constexpr uint8_t greenLed = 6;
constexpr uint8_t blueLed = 8;

constexpr uint8_t relay = 4; // Set Relay Pin
constexpr uint8_t wipeB = 3; // Button pin for WipeMode

boolean match = false; // initialize card match to false
boolean programMode = false; // initialize programming mode to false
boolean replaceMaster = false;

uint8_t successRead; // Variable integer to keep if we have Successful Read from Reader

byte storedCard[4]; // Stores an ID read from EEPROM
byte readCard[4]; // Stores scanned ID read from RFID Module
byte masterCard[4]; // Stores master card's ID read from EEPROM

// Create MFRC522 instance.

constexpr uint8_t RST_PIN = 5; // Configurable, see typical pin layout above
constexpr uint8_t SS_PIN = 53; // Configurable, see typical pin layout above

MFRC522 mfrc522(SS_PIN, RST_PIN);

/////////////////////////////// Setup ///////////////////////////////
void setup() {

    //Arduino Pin Configuration

```

```

pinMode(redLed, OUTPUT);
pinMode(greenLed, OUTPUT);
pinMode(blueLed, OUTPUT);
pinMode(13, OUTPUT);
pinMode(wipeB, INPUT_PULLUP); // Enable pin's pull up resistor
pinMode(relay, OUTPUT);
pinMode(10, OUTPUT);

//Be careful how relay circuit behave on while resetting or power-cycling your Arduino
digitalWrite(relay, HIGH); // Make sure door is locked
digitalWrite(redLed, LED_OFF); // Make sure led is off
digitalWrite(greenLed, LED_OFF); // Make sure led is off
digitalWrite(blueLed, LED_OFF); // Make sure led is off

//Protocol Configuration
Serial.begin(9600); // Initialize serial communications with PC
SPI.begin(); // MFRC522 Hardware uses SPI protocol
mfrc522.PCD_Init(); // Initialize MFRC522 Hardware

//If you set Antenna Gain to Max it will increase reading distance
//mfrc522.PCD_SetAntennaGain(mfrc522.RxGain_max);

// Serial.println(F("Access Control Example v0.1")); // For debugging purposes

ShowReaderDetails(); // Show details of PCD - MFRC522 Card Reader details

//Wipe Code - If the Button (wipeB) Pressed while setup run (powered on) it wipes EEPROM
if (digitalRead(wipeB) == LOW) { // when button pressed pin should get low, button connected to ground
  digitalWrite(redLed, LED_ON); // Red Led stays on to inform user we are going to wipe
  bool buttonState = monitorWipeButton(10000); // Give user enough time to cancel operation
}

```

```
if (buttonState == true && digitalRead(wipeB) == LOW) { // If button still be pressed, wipe EEPROM

// Serial.println(F("Starting Wiping EEPROM"));

for (uint16_t x = 0; x < EEPROM.length(); x = x + 1) { //Loop end of EEPROM address

if (EEPROM.read(x) == 0) { //If EEPROM address 0

// do nothing, already clear, go to the next address in order to save time and reduce writes to
EEPROM

}

else {

EEPROM.write(x, 0); // if not write 0 to clear, it takes 3.3mS

}

}

// Serial.println(F("EEPROM Successfully Wiped"));

digitalWrite(redLed, LED_OFF); // visualize a successful wipe
delay(200);
digitalWrite(redLed, LED_ON);
delay(200);
digitalWrite(redLed, LED_OFF);
delay(200);
digitalWrite(redLed, LED_ON);
delay(200);
digitalWrite(redLed, LED_OFF);

}

else {

// Serial.println(F("Wiping Cancelled")); // Show some feedback that the wipe button did not pressed for
15 seconds
```

```

digitalWrite(redLed, LED_OFF);
}

}

// Check if master card defined, if not let user choose a master card

// This also useful to just redefine the Master Card

// You can keep other EEPROM records just write other than 143 to EEPROM address 1

// EEPROM address 1 should hold magical number which is '143'

if (EEPROM.read(1) != 143) {

do {

    successRead = getID();      // sets successRead to 1 when we get read from reader otherwise 0

    digitalWrite(blueLed, LED_ON); // Visualize Master Card need to be defined

    delay(200);

    digitalWrite(blueLed, LED_OFF);

    delay(200);

}

while (!successRead);        // Program will not go further while you not get a successful read

for ( uint8_t j = 0; j < 4; j++ ) { // Loop 4 times

    EEPROM.write( 2 + j, readCard[j] ); // Write scanned PICC's UID to EEPROM, start from address 3

}

EEPROM.write(1, 143);        // Write to EEPROM we defined Master Card.

}

for ( uint8_t i = 0; i < 4; i++ ) { // Read Master Card's UID from EEPROM

    masterCard[i] = EEPROM.read(2 + i); // Write it to masterCard

    Serial.print(masterCard[i], HEX);

}

```

```

cycleLeds(); // Everything ready lets give user some feedback by cycling leds
Serial.begin(9600);
// put your setup code here, to run once:

}

/////////////////////////////// Main Loop /////////////////////////
void loop () {

// put your main code here, to run repeatedly:

do {
    successRead = getID(); // sets successRead to 1 when we get read from reader otherwise 0
    // When device is in use if wipe button pressed for 10 seconds initialize Master Card wiping
    if (digitalRead(wipeB) == LOW) { // Check if button is pressed
        // Visualize normal operation is iterrupted by pressing wipe button Red is like more Warning to user
        digitalWrite(redLed, LED_ON); // Make sure led is off
        digitalWrite(greenLed, LED_OFF); // Make sure led is off
        digitalWrite(blueLed, LED_OFF); // Make sure led is off
        // Give some feedback
        bool buttonState = monitorWipeButton(10000); // Give user enough time to cancel operation
        if (buttonState == true && digitalRead(wipeB) == LOW) { // If button still be pressed, wipe EEPROM
            EEPROM.write(1, 0); // Reset Magic Number.

        while (1);

    }

if (programMode) {
    cycleLeds(); // Program Mode cycles through Red Green Blue waiting to read a new card
}

```

```

}

else {

    normalModeOn(); // Normal mode, blue Power LED is on, all others are off

    int NROpen = Serial.read();

    if(NROpen == 49){

        doorSuccess();

    }

}

while (!successRead); //the program will not go further while you are not getting a successful read

if (programMode) {

    if ( isMaster(readCard) ) { //When in program mode check First If master card scanned again to exit
program mode

    programMode = false;

    return;

}

else {

    if ( findID(readCard) ) { // If scanned card is known delete it

    }

    else {          // If scanned card is not known add it

    }

}

}

else {

    if ( isMaster(readCard)) { // If scanned card's ID matches Master Card's ID - enter program mode

        programMode = true;

        uint8_t count = EEPROM.read(0); // Read the first Byte of EEPROM that

    }

    else {

```

```

if ( findID(readCard) ) { // If not, see if the card is in the EEPROM
    // Serial.println(F("1"));

    granted(300);      // Open the door lock for 300 ms

    doorSuccess();

}

else {   // If not, show that the ID was not valid

    // Serial.println(F("You shall not pass"));

    denied();

}

}

}

}

```

/////////// Access Granted ///////////

```

void granted ( uint16_t setDelay) {

digitalWrite(blueLed, LED_OFF); // Turn off blue LED
digitalWrite(redLed, LED_OFF); // Turn off red LED
digitalWrite(greenLed, LED_ON); // Turn on green LED
digitalWrite(relay, LOW); // Unlock door!
delay(setDelay); // Hold door lock open for given seconds
digitalWrite(relay, HIGH); // Relock door
delay(1000); // Hold green LED on for a second

}

```

/////////// Access Denied ///////////

```

void denied() {

digitalWrite(greenLed, LED_OFF); // Make sure green LED is off
digitalWrite(blueLed, LED_OFF); // Make sure blue LED is off
digitalWrite(redLed, LED_ON); // Turn on red LED
delay(1000);

```

```

}

/////////////////////////////// Get PICC's UID //////////////////////

uint8_t getID() {

// Getting ready for Reading PICCs

if ( ! mfrc522.PICC_IsNewCardPresent() ) { //If a new PICC placed to RFID reader continue

    return 0;

}

if ( ! mfrc522.PICC_ReadCardSerial() ) { //Since a PICC placed get Serial and continue

    return 0;

}

// There are Mifare PICCs which have 4 byte or 7 byte UID care if you use 7 byte PICC

// I think we should assume every PICC as they have 4 byte UID

// Until we support 7 byte PICCs

//Serial.println(F("Scanned PICC's UID:"));

for ( uint8_t i = 0; i < 4; i++ ) { //

    readCard[i] = mfrc522.uid.uidByte[i];

    Serial.print(readCard[i], HEX);

}

Serial.println("");


mfrc522.PICC_HaltA(); // Stop reading

return 1;

}

void ShowReaderDetails() {

// Get the MFRC522 software version

byte v = mfrc522.PCD_ReadRegister(mfrc522.VersionReg);

//Serial.print(F("MFRC522 Software Version: 0x"));

// Serial.print(v, HEX);
}

```

```

if (v == 0x91)
    Serial.print(F(" = v1.0"));

//else if (v == 0x92)
// Serial.print(F(" = v2.0"));

// else

// Serial.print(F(" (unknown),probably a chinese clone?"));

Serial.println("");

// When 0x00 or 0xFF is returned, communication probably failed

if ((v == 0x00) || (v == 0xFF)) {

    // Visualize system is halted

    digitalWrite(greenLed, LED_OFF); // Make sure green LED is off
    digitalWrite(blueLed, LED_OFF); // Make sure blue LED is off
    digitalWrite(redLed, LED_ON); // Turn on red LED
    while (true); // do not go further

}

}

```

```

/////////////////////////////// Cycle Leds (Program Mode) //////////////////////

void cycleLeds() {

    digitalWrite(redLed, LED_OFF); // Make sure red LED is off
    digitalWrite(greenLed, LED_ON); // Make sure green LED is on
    digitalWrite(blueLed, LED_OFF); // Make sure blue LED is off
    delay(200);

    digitalWrite(redLed, LED_OFF); // Make sure red LED is off
    digitalWrite(greenLed, LED_OFF); // Make sure green LED is off
    digitalWrite(blueLed, LED_ON); // Make sure blue LED is on
    delay(200);

    digitalWrite(redLed, LED_ON); // Make sure red LED is on
    digitalWrite(greenLed, LED_OFF); // Make sure green LED is off
    digitalWrite(blueLed, LED_OFF); // Make sure blue LED is off
}
```

```
delay(200);  
}  
  
////////////////////////////// Normal Mode Led //////////////////////
```

```
void normalModeOn () {  
    digitalWrite(blueLed, LED_ON); // Blue LED ON and ready to read card  
    digitalWrite(redLed, LED_OFF); // Make sure Red LED is off  
    digitalWrite(greenLed, LED_OFF); // Make sure Green LED is off  
    digitalWrite(relay, HIGH); // Make sure Door is Locked  
}
```

```
////////////////////////////// Read an ID from EEPROM //////////////////////
```

```
void readID( uint8_t number ) {  
    uint8_t start = (number * 4) + 2; // Figure out starting position  
    for ( uint8_t i = 0; i < 4; i++ ) { // Loop 4 times to get the 4 Bytes  
        storedCard[i] = EEPROM.read(start + i); // Assign values read from EEPROM to array  
    }  
}
```

```
////////////////////////////// Add ID to EEPROM //////////////////////
```

```
void writeID( byte a[] ) {  
    if ( !findID( a ) ) { // Before we write to the EEPROM, check to see if we have seen this card before!  
        uint8_t num = EEPROM.read(0); // Get the number of used spaces, position 0 stores the number of ID  
        cards  
        uint8_t start = ( num * 4 ) + 6; // Figure out where the next slot starts  
        num++; // Increment the counter by one  
        EEPROM.write( 0, num ); // Write the new count to the counter  
        for ( uint8_t j = 0; j < 4; j++ ) { // Loop 4 times  
            EEPROM.write( start + j, a[j] ); // Write the array values to EEPROM in the right position  
        }  
    }  
}
```

```

successWrite();

Serial.println(F("Successfully added ID record to EEPROM"));

}

else {

failedWrite();

Serial.println(F("Failed! There is something wrong with ID or bad EEPROM"));

}

}

//////////////////////////// Remove ID from EEPROM //////////////////////

void deleteID( byte a[] ) {

if ( !findID( a ) ) { // Before we delete from the EEPROM, check to see if we have this card!

failedWrite(); // If not

Serial.println(F("Failed! There is something wrong with ID or bad EEPROM"));

}

else {

uint8_t num = EEPROM.read(0); // Get the number of used spaces, position 0 stores the number of ID cards

uint8_t slot; // Figure out the slot number of the card

uint8_t start; // = ( num * 4 ) + 6; // Figure out where the next slot starts

uint8_t looping; // The number of times the loop repeats

uint8_t j;

uint8_t count = EEPROM.read(0); // Read the first Byte of EEPROM that stores number of cards

slot = findIDSLOT( a ); // Figure out the slot number of the card to delete

start = (slot * 4) + 2;

looping = ((num - slot) * 4);

num--; // Decrement the counter by one

EEPROM.write( 0, num ); // Write the new count to the counter

for ( j = 0; j < looping; j++ ) { // Loop the card shift times
}
}
}

```

```

    EEPROM.write( start + j, EEPROM.read(start + 4 + j)); // Shift the array values to 4 places earlier in the
EEPROM
}

for ( uint8_t k = 0; k < 4; k++ ) { // Shifting loop

    EEPROM.write( start + j + k, 0);

}

successDelete();

Serial.println(F("Successfully removed ID record from EEPROM"));

}

}

```

```

////////////////////////////// Check Bytes //////////////////////

boolean checkTwo ( byte a[], byte b[] ) {

if ( a[0] != 0 ) // Make sure there is something in the array first

    match = true; // Assume they match at first

for ( uint8_t k = 0; k < 4; k++ ) { // Loop 4 times

    if ( a[k] != b[k] ) // IF a != b then set match = false, one fails, all fail

        match = false;

}

if ( match ) { // Check to see if if match is still true

    return true; // Return true

}

else {

    return false; // Return false

}

}

```

```

////////////////////////////// Find Slot //////////////////////

uint8_t findIDSLOT( byte find[] ) {

uint8_t count = EEPROM.read(0); // Read the first Byte of EEPROM that

```

```

for ( uint8_t i = 1; i <= count; i++ ) { // Loop once for each EEPROM entry
    readID(i); // Read an ID from EEPROM, it is stored in storedCard[4]
    if ( checkTwo( find, storedCard ) ) { // Check to see if the storedCard read from EEPROM
        // is the same as the find[] ID card passed
        return i; // The slot number of the card
        break; // Stop looking we found it
    }
}
}
}

```

////////// Find ID From EEPROM //

```

boolean findID( byte find[] ) {
    uint8_t count = EEPROM.read(0); // Read the first Byte of EEPROM that
    for ( uint8_t i = 1; i <= count; i++ ) { // Loop once for each EEPROM entry
        readID(i); // Read an ID from EEPROM, it is stored in storedCard[4]
        if ( checkTwo( find, storedCard ) ) { // Check to see if the storedCard read from EEPROM
            return true;
            break; // Stop looking we found it
        }
    }
    else { // If not, return false
    }
}
return false;
}

```

////////// Write Success to EEPROM //

```

// Flashes the green LED 3 times to indicate a successful write to EEPROM
void successWrite() {
    digitalWrite(blueLed, LED_OFF); // Make sure blue LED is off
    digitalWrite(redLed, LED_OFF); // Make sure red LED is off
}

```

```
digitalWrite(greenLed, LED_OFF); // Make sure green LED is on
delay(200);

digitalWrite(greenLed, LED_ON); // Make sure green LED is on
delay(200);

digitalWrite(greenLed, LED_OFF); // Make sure green LED is off
delay(200);

digitalWrite(greenLed, LED_ON); // Make sure green LED is on
delay(200);

digitalWrite(greenLed, LED_OFF); // Make sure green LED is off
delay(200);

digitalWrite(greenLed, LED_ON); // Make sure green LED is on
delay(200);

}

//////////////////////////////////////////////////////////////////////// Write Failed to EEPROM ///////////////////////////////
// Flashes the red LED 3 times to indicate a failed write to EEPROM
```

```
void failedWrite() {

digitalWrite(blueLed, LED_OFF); // Make sure blue LED is off
digitalWrite(redLed, LED_OFF); // Make sure red LED is off
digitalWrite(greenLed, LED_OFF); // Make sure green LED is off
delay(200);

digitalWrite(redLed, LED_ON); // Make sure red LED is on
delay(200);

digitalWrite(redLed, LED_OFF); // Make sure red LED is off
delay(200);

digitalWrite(redLed, LED_ON); // Make sure red LED is on
delay(200);

digitalWrite(redLed, LED_OFF); // Make sure red LED is off
delay(200);

digitalWrite(redLed, LED_ON); // Make sure red LED is on
```

```
delay(200);
}

/////////////////////////////// Success Remove UID From EEPROM
///////////////////////////////

// Flashes the blue LED 3 times to indicate a success delete to EEPROM

void successDelete() {

    digitalWrite(blueLed, LED_OFF); // Make sure blue LED is off
    digitalWrite(redLed, LED_OFF); // Make sure red LED is off
    digitalWrite(greenLed, LED_OFF); // Make sure green LED is off
    delay(200);

    digitalWrite(blueLed, LED_ON); // Make sure blue LED is on
    delay(200);

    digitalWrite(blueLed, LED_OFF); // Make sure blue LED is off
    delay(200);

    digitalWrite(blueLed, LED_ON); // Make sure blue LED is on
    delay(200);

    digitalWrite(blueLed, LED_OFF); // Make sure blue LED is off
    delay(200);

    digitalWrite(blueLed, LED_ON); // Make sure blue LED is on
    delay(200);

}

/////////////////////////////// Successfull door opening /////////////////////
// Flashes the green LED 3 times to indicate a successful door opening

void doorSuccess() {

    digitalWrite(blueLed, LED_OFF); // Make sure blue LED is off
    digitalWrite(redLed, LED_OFF); // Make sure red LED is off
    digitalWrite(greenLed, LED_ON); // Make sure green LED is on
```

```

delay(3000);

digitalWrite(greenLed, LED_OFF); // Make sure green LED is off

delay(200);

}

////////////////// Check readCard IF is masterCard //////////////////

// Check to see if the ID passed is the master programing card

boolean isMaster( byte test[] ) {

if ( checkTwo( test, masterCard ) )

    return true;

else

    return false;

}

bool monitorWipeButton(uint32_t interval) {

    uint32_t now = (uint32_t)millis();

    while ((uint32_t)millis() - now < interval) {

        // check on every half a second

        if (((uint32_t)millis() % 500) == 0) {

            if (digitalRead(wipeB) != LOW)

                return false;

        }

    }

    return true;

}

```