NFC/RFID Access Control

Introduction

Table of Content

Description

On starting for the first time, the LCD screen will indicate that no master tag is defined and it will request you to define one (A blue flashing LED will indicate the waiting for master definition), once you present the first RFID tag it will be saved as your master tag.

The master tag will act as programmer and you can use it to add or remove other tags. To do this, you can scan it again to start “program mode” (Continued flashing of all 3 LEDs will indicate that the system is in program mode).

When in “program mode”, scanning the tags will add/Remove these from the system. Scan the tags that you want to use to open the door and system will store the UID’s of these tags in the EEPROM. Scan the tag again to remove it from the EEPROM. To exit the program mode, scan the master tag again.

After doing this you can scan the tags that you have added to the system to open the door (a green LED, a message on the LCD screen and the activation of the servo motor will indicate success). If the wrong tag is scanned, the door will remain closed and you'll be informed by a RED LED, a buzzer sound and a message in the LCD.

To reset the system, press the reset button of Arduino and then long-press the wipe button for 10 seconds. This will remove all the data from the EEPROM including the master tag.

General description

Block Diagram:

Hardware Design

Hardware List:

- Arduino UNO
- I2C LCD
- MFRC522 RFID Reader
- RFID tags
- SG90 micro-servo motor
- 3x LED
- 3x 220 ohm resistor

- Buzzer
- Push button
- 6 to 12V power source

Wiring Diagram:

Breadboard Schematic:

Wiring

DIGITAL

- Pin ~3 → 1st leg pair of Push button
- Pin 4 → Buzzer S pin
- Pin ~5 → Red LED Anode (long)
- Pin ~6 → Blue LED Anode (long)
- Pin 7 → Green LED Anode (long)
- Pin 8 → Signal of Servo Motor (Orange on SG-90)
- Pin ~9 → RST on RC522 RFID Module
- Pin ~10 → IRQ on RC522 RFID Module
- Pin ~11 → SCK on RC522 RFID Module
- Pin 12 → MOSI on RC522 RFID Module
- GND → 220 ohm resistors connected to Cathod (short) of all LEDs
- GND → GND of Servo Motor (Black on SG-90)
- GND → - of Buzzer
- GND → 2nd leg pair of Push button

ANALOG IN

- A5 → SCL of 1602 I2C Module
- A4 → SDA of 1602 I2C Module

POWER

- GND → GND of 1602 I2C Module
- GND → GND of RC522 RFID Module
- 5V → VCC of 1602 I2C Module
- 3V3 → VCC of RC522 RFID Module
- IOREF → VCC of Servo Motor (Red on SG-90)

Software Design

Software Used:

- Arduino IDE 1.8.19
- Notepad++
- Fritzing 0.9.9.64
- Microsoft Visio Drawing

Code

Libraries Used:

- EEPROM (https://docs.arduino.cc/learn/built-in-libraries/eeprom)

We used the EEPROM to save the tags information, this allows us to remain with the tags information even after powering off the module.

- MFRC522 (https://www.arduino.cc/reference/en/libraries/mfrc522/)

We use this library to read and write different types of Radio-Frequency Identification (RFID) cards on the Arduino using an RC522-based reader connected via the Serial Peripheral Interface (SPI) interface.

- SPI (https://www.arduino.cc/reference/en/language/functions/communication/spi/)

As mentioned above the RFID reader communicates with the Arduino through the SPI protocol, this library allows us to communicate with SPI devices, using the Arduino as the controller device.

- LiquidCrystal I2C (https://www.arduino.cc/reference/en/libraries/liquidcrystal-i2c/)

With this library, we are able to communicate with alphanumerical liquid crystal displays (LCDs) using an I2C communication interface and continue to have extremely similar functions as the main LiquidCrystal library. Due to the lack of resources on the Arduino Uno board and to simplify the connections, we use an LCD with an I2C module configured on 0x27 address.

- Wire (https://www.arduino.cc/reference/en/language/functions/communication/wire/)

For the communication with the above-mentioned LCD we use this library which allows us to communicate with I2C/TWI devices.

- Servo (https://www.arduino.cc/reference/en/libraries/servo/)

We use a servo motor to exemplify a locking mechanism, this library allows our Arduino board to control our SG90 micro-servo motor.

```
#include <EEPROM.h>     // We are going to read and write Tag's UIDs
from/to EEPROM
#include <MFRC522.h>
#include <LiquidCrystal_I2C.h>
#include <Servo.h>
#include <SPI.h>
#include <Wire.h>
// Create instances
MFRC522 mfrc522(10, 9); // MFRC522 mfrc522(SS_PIN, RST_PIN)
LiquidCrystal_I2C lcd(0x27, 16, 2);
Servo myServo;  // create servo object to control a servo
// Set Pins for led's, servo, buzzer and wipe button
constexpr uint8_t greenLed = 7;
constexpr uint8_t blueLed = 6;
constexpr uint8_t redLed = 5;
constexpr uint8_t ServoPin = 8;
constexpr uint8_t BuzzerPin = 4;
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
```

Setup:

- Set up of the arduino Pin Configuration
- Ensure the LED's are OFF
- Initiate the LCD, SPI, MFRC522 and attach the servo on pin8
- Set servo initial position
- Setup of Master Card reading and definition code (Master Card allows the user to add or remove RFID/NFC tags)
- Setup of Wipe code (If the reset button is pressed while the setup is running - during power on - it will wipe the EEPROM

```
/////////////////////////////////////// Setup
///////////////////////////////////
void setup() {
  //Arduino Pin Configuration
  pinMode(redLed, OUTPUT);
  pinMode(greenLed, OUTPUT);
  pinMode(blueLed, OUTPUT);
  pinMode(BuzzerPin, OUTPUT);
```

```
  pinMode(wipeB, INPUT_PULLUP);   // Enable pin's pull up resistor
  // Make sure led's are off
  digitalWrite(redLed, LOW);
  digitalWrite(greenLed, LOW);
  digitalWrite(blueLed, LOW);
  //Protocol Configuration
  lcd.init();  // initialize the LCD
  lcd.backlight();
  SPI.begin();           // MFRC522 Hardware uses SPI protocol
  mfrc522.PCD_Init();    // Initialize MFRC522 Hardware
  myServo.attach(ServoPin);   // attaches the servo on pin 8 to the servo
object
  myServo.write(10);   // Initial Position
  //If you set Antenna Gain to Max it will increase reading distance
  //mfrc522.PCD_SetAntennaGain(mfrc522.RxGain_max);
  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, HIGH); // Red Led stays on to inform user we are
going to wipe
    lcd.setCursor(0, 0);
    lcd.print("Button Pressed");
    digitalWrite(BuzzerPin, HIGH);
    delay(1000);
    digitalWrite(BuzzerPin, LOW);
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("This will remove");
    lcd.setCursor(0, 1);
    lcd.print("all records");
    delay(2000);
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("You have 10 ");
    lcd.setCursor(0, 1);
    lcd.print("secs to Cancel");
    delay(2000);
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Unpres to cancel");
    lcd.setCursor(0, 1);
    lcd.print("Counting: ");
    bool buttonState = monitorWipeButton(10000); // Give user enough time
to cancel operation
    if (buttonState == true && digitalRead(wipeB) == LOW) {    // If button
still be pressed, wipe EEPROM
      lcd.print("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
        }
      }
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print("Wiping Done");
      // visualize a successful wipe
      digitalWrite(redLed, LOW);
      digitalWrite(BuzzerPin, HIGH);
      delay(200);
      digitalWrite(redLed, HIGH);
      digitalWrite(BuzzerPin, LOW);
      delay(200);
      digitalWrite(redLed, LOW);
      digitalWrite(BuzzerPin, HIGH);
      delay(200);
      digitalWrite(redLed, HIGH);
      digitalWrite(BuzzerPin, LOW);
      delay(200);
      digitalWrite(redLed, LOW);
    }
    else {
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print("Wiping Cancelled"); // Show some feedback that the wipe
button did not pressed for 10 seconds
      digitalWrite(redLed, LOW);
    }
  }
  // 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) {
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("No Master Card ");
    lcd.setCursor(0, 1);
    lcd.print("Defined");
    delay(2000);
    lcd.setCursor(0, 0);
    lcd.print("Scan A Tag to ");
```

```
    lcd.setCursor(0, 1);
    lcd.print("Define as Master");
    do {
      successRead = getID();            // sets successRead to 1 when we
get read from reader otherwise 0
      // Visualize Master Card need to be defined
      digitalWrite(blueLed, HIGH);
      digitalWrite(BuzzerPin, HIGH);
      delay(200);
      digitalWrite(BuzzerPin, LOW);
      digitalWrite(blueLed, LOW);
      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 Tag's UID to
EEPROM, start from address 3
    }
    EEPROM.write(1, 143);                  // Write to EEPROM we defined
Master Card.
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Master Defined");
    delay(2000);
  }
  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
  }
  ShowOnLCD();    // Print data on LCD
  cycleLeds();    // Everything ready lets give user some feedback by
cycling leds
}
```

Main Loop:

- Setup of Program mode code (Program mode allows to add or remove RFID/NFC tags by presenting the Master Tag)

```
///////////////////////////////////////// Main Loop
/////////////////////////////////////
void loop () {
  do {
    successRead = getID();  // sets successRead to 1 when we get read from
reader otherwise 0
```

```
    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
    }
  }
  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
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print("Exiting Program Mode");
      digitalWrite(BuzzerPin, HIGH);
      delay(1000);
      digitalWrite(BuzzerPin, LOW);
      ShowOnLCD();
      programMode = false;
      return;
    }
    else {
      if ( findID(readCard) ) { // If scanned card is known delete it
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print("Already there");
        deleteID(readCard);
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print("Tag to ADD/REM");
        lcd.setCursor(0, 1);
        lcd.print("Master to Exit");
      }
      else {                    // If scanned card is not known add it
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print("New Tag,adding...");
        writeID(readCard);
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print("Scan to ADD/REM");
        lcd.setCursor(0, 1);
        lcd.print("Master to Exit");
      }
    }
  }
  else {
    if ( isMaster(readCard)) {    // If scanned card's ID matches Master
```

```
Card's ID - enter program mode
      programMode = true;
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print("Program Mode");
      uint8_t count = EEPROM.read(0);   // Read the first Byte of EEPROM
that stores the number of ID's in EEPROM
      lcd.setCursor(0, 1);
      lcd.print("I have ");
      lcd.print(count);
      lcd.print(" records");
      digitalWrite(BuzzerPin, HIGH);
      delay(2000);
      digitalWrite(BuzzerPin, LOW);
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print("Scan a Tag to ");
      lcd.setCursor(0, 1);
      lcd.print("ADD/REMOVE");
    }
    else {
      if ( findID(readCard) ) { // If not, see if the card is in the EEPROM
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print("Access Granted");
        granted();         // Open the door lock
        myServo.write(10);
        ShowOnLCD();
      }
      else {      // If not, show that the Access is denied
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print("Access Denied");
        denied();
        ShowOnLCD();
      }
    }
  }
}
```

Access Granted:

- Setup of the code for when the access is allowed (Green LED and servo position change)

```
////////////////////////////////////  Access Granted
///////////////////////////////////
```

```
void granted () {
  digitalWrite(blueLed, LOW);   // Turn off blue LED
  digitalWrite(redLed, LOW);  // Turn off red LED
  digitalWrite(greenLed, HIGH);   // Turn on green LED
  myServo.write(90);
  delay(1000);
}
```

Access Denied:

- Setup of the code for when the access is not allowed (RED LED and buzzer sound)

```
////////////////////////////////////// Access Denied
///////////////////////////////////
void denied() {
  digitalWrite(greenLed, LOW);  // Make sure green LED is off
  digitalWrite(blueLed, LOW);   // Make sure blue LED is off
  digitalWrite(redLed, HIGH);   // Turn on red LED
  digitalWrite(BuzzerPin, HIGH);
  delay(1000);
  digitalWrite(BuzzerPin, LOW);
}
```

Getting the TAGs UID:

- Reads the UID of a RFID/NFC tag

```
////////////////////////////////////// Get Tag's UID
///////////////////////////////////
uint8_t getID() {
  // Getting ready for Reading Tags
  if ( ! mfrc522.PICC_IsNewCardPresent()) { //If a new Tag placed to RFID
reader continue
    return 0;
  }
  if ( ! mfrc522.PICC_ReadCardSerial()) {   //Since a Tag placed get Serial
and continue
    return 0;
  }
  // There are Mifare Tags which have 4 byte or 7 byte UID care if you use
7 byte Tag
```

```
  // I think we should assume every Tag as they have 4 byte UID
  // Until we support 7 byte Tags
  for ( uint8_t i = 0; i < 4; i++) {  //
    readCard[i] = mfrc522.uid.uidByte[i];
  }
  mfrc522.PICC_HaltA(); // Stop reading
  return 1;
}
```

RFID initialization:

- Verifies that the RFID is correctly initialized
- If not, presents a message, makes a buzzer sound and turns on Red LED

```
////////////////////// Check if RFID Reader is correctly initialized or
not ////////////////////
void ShowReaderDetails() {
  // Get the MFRC522 software version
  byte v = mfrc522.PCD_ReadRegister(mfrc522.VersionReg);
  // When 0x00 or 0xFF is returned, communication probably failed
  if ((v == 0x00) || (v == 0xFF)) {
    lcd.setCursor(0, 0);
    lcd.print("Communication Failure");
    lcd.setCursor(0, 1);
    lcd.print("Check Connections");
    digitalWrite(BuzzerPin, HIGH);
    delay(2000);
    // Visualize system is halted
    digitalWrite(greenLed, LOW);  // Make sure green LED is off
    digitalWrite(blueLed, LOW);   // Make sure blue LED is off
    digitalWrite(redLed, HIGH);   // Turn on red LED
    digitalWrite(BuzzerPin, LOW);
    while (true); // do not go further
  }
}
```

Program Mode LED configuration:

- Cycles through LEDs during Program Mode

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

```
///////////////////////////////////
void cycleLeds() {
  digitalWrite(redLed, LOW);  // Make sure red LED is off
  digitalWrite(greenLed, HIGH);   // Make sure green LED is on
  digitalWrite(blueLed, LOW);   // Make sure blue LED is off
  delay(200);
  digitalWrite(redLed, LOW);  // Make sure red LED is off
  digitalWrite(greenLed, LOW);  // Make sure green LED is off
  digitalWrite(blueLed, HIGH);  // Make sure blue LED is on
  delay(200);
  digitalWrite(redLed, HIGH);   // Make sure red LED is on
  digitalWrite(greenLed, LOW);  // Make sure green LED is off
  digitalWrite(blueLed, LOW);   // Make sure blue LED is off
  delay(200);
}
```

Normal Mode LED configuration:

- Turns Blue LED on Normal Mode
- Disables Red and Green LEDs

```
//////////////////////////////////////// Normal Mode Led
///////////////////////////////////
void normalModeOn () {
  digitalWrite(blueLed, HIGH);  // Blue LED ON and ready to read card
  digitalWrite(redLed, LOW);  // Make sure Red LED is off
  digitalWrite(greenLed, LOW);  // Make sure Green LED is off
}
```

Read UID from EEPROM:

- Code setup for reading 4 bytes out of the EEPROM and assign values to an array

```
//////////////////////////////////////// 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
```

```
  }
}
```

Adding UID to EEPROM:

- Checks the EEPROM for used spaced and uses the next available slot to write 4 bytes of the UID
- Blinks the Green LED
- Prints message

```
//////////////////////////////////////// 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 numer 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
    }
    BlinkLEDS(greenLed);
    lcd.setCursor(0, 1);
    lcd.print("Added");
    delay(1000);
  }
  else {
    BlinkLEDS(redLed);
    lcd.setCursor(0, 0);
    lcd.print("Failed!");
    lcd.setCursor(0, 1);
    lcd.print("wrong ID or bad EEPROM");
    delay(2000);
  }
}
```

Removing UID from EEPROM:

- Checks the EEPROM for the location of the presented UID

- removes the 4 bytes related to the tag we want to delete
- Blinks the Blue LED
- Prints message

```
////////////////////////////////////// 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!
    BlinkLEDS(redLed);      // If not
    lcd.setCursor(0, 0);
    lcd.print("Failed!");
    lcd.setCursor(0, 1);
    lcd.print("wrong ID or bad EEPROM");
    delay(2000);
  }
  else {
    uint8_t num = EEPROM.read(0);   // Get the numer 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);
    }
    BlinkLEDS(blueLed);
    lcd.setCursor(0, 1);
    lcd.print("Removed");
    delay(1000);
  }
}
```

Other loop functions necessary:

- Check Bytes
- Find EEPROM slots
- Find IDs from EEPROM
- Blinking of LEDs
- Check Master Tag
- Counter for wipe button pressing
- Normal mode LCD print

```
///////////////////////////////////////// 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;
}
///////////////////////////////////////// Blink LED's For Indication
///////////////////////////////////
void BlinkLEDS(int led) {
  digitalWrite(blueLed, LOW);   // Make sure blue LED is off
  digitalWrite(redLed, LOW);  // Make sure red LED is off
  digitalWrite(greenLed, LOW);  // Make sure green LED is off
  digitalWrite(BuzzerPin, HIGH);
  delay(200);
  digitalWrite(led, HIGH);  // Make sure blue LED is on
  digitalWrite(BuzzerPin, LOW);
  delay(200);
  digitalWrite(led, LOW);   // Make sure blue LED is off
  digitalWrite(BuzzerPin, HIGH);
  delay(200);
  digitalWrite(led, HIGH);  // Make sure blue LED is on
  digitalWrite(BuzzerPin, LOW);
  delay(200);
  digitalWrite(led, LOW);   // Make sure blue LED is off
  digitalWrite(BuzzerPin, HIGH);
  delay(200);
  digitalWrite(led, HIGH);  // Make sure blue LED is on
  digitalWrite(BuzzerPin, LOW);
  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;
}
////////////////// Counter to check in reset/wipe button is pressed or not
////////////////////
bool monitorWipeButton(uint32_t interval) {
  unsigned long currentMillis = millis(); // grab current time
  while (millis() - currentMillis < interval)  {
```

```
    int timeSpent = (millis() - currentMillis) / 1000;
    Serial.println(timeSpent);
    lcd.setCursor(10, 1);
    lcd.print(timeSpent);
    // check on every half a second
    if (((uint32_t)millis() % 10) == 0) {
      if (digitalRead(wipeB) != LOW) {
        return false;
      }
    }
  }
  return true;
}
////////////////////// Print Info on LCD
///////////////////////////////////
void ShowOnLCD() {
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print(" Access Control");
  lcd.setCursor(0, 1);
  lcd.print("   Scan a Tag");
}
```

Results

The project worked as expected. When initialized for the first time it will show us that no current Master Tag is defined and will wait until a Tag is presented to be used as Master.

Once a master tag is defined it will start in normal mode and expect a tag to be presented for access.

In normal mode if a known tag is presented the servo motor will rotate 90 degrees, the LCD will display an access granted message and the green LED will turn on. If an unknown tag is presented the LCD will display an access denied message, the red LED will turn on and the buzzer will sound.

In normal mode if the master tag is presented, it will start Program Mode (cycling between LEDs and displaying a message on the LCD) and expect a known or unknown tag to be presented to delete or add to the EEPROM accordingly.

When the system is initiated, if the wipe button is pressed during the first 10 seconds (showing a counter on the LCD) it will wipe the EEPROM

Prototype

Prototype #1

Video: Microprocessor Architecture Project - NFC/RFID Access Control - Nuno Bandarra 1222B - Prototype#1

Prototype #2

Prototype #2 - Interior

Prototype #2 - Exterior

Video: Microprocessor Architecture Project - NFC/RFID Access Control - Nuno Bandarra 1222B - Prototype#2

Conclusions

This was a very interesting project allowing us to use the technics learned throughout the year during the laboratories.

Although this was a simple project, it can be adapted and applied to lots of different environments, some examples include:

- Mailbox locking (No keys)
- Jewellery box protection
- Gunlocker secure but easy access
- etc...

By defying students with such projects, the world continues advancing and improving.

Download

NFC-RFID Access Control_NunoBandarra_1222B

Log

- 26.04.2022 - Project selection.
- 26.04.2022 - Wiki page created.
- 26.04.2022 - Introduction, Block Diagram, and Hardware List added.
- 27.04.2022-13.05.2022 - Research.
- 14.05.2022 - Wiring Diagram and Breadboard Schematic added.

- 14.05.2022 - First prototype completed.
- 28.05.2022 - First prototype video published and Results completed.
- 28.05.2022 - Table of content, Software design, Conclusions, Arduino Pins connections and Download added.
- 29.05.2022-01.06.2022 - Second prototype design and execution.
- 01.06.2022 - Second prototype video published.

Bibliography/Resources

Software Resources

- https://create.arduino.cc/projecthub/shubamtayal/rfid-scanner-full-tutorial-6518db
- https://create.arduino.cc/projecthub/muhammad-aqib/rfid-and-keypad-door-lock-and-alert-system-using-arduino-60f050?ref=search&ref_id=rfid&offset=0
- https://create.arduino.cc/projecthub/Aritro/security-access-using-rfid-reader-f7c746
- https://create.arduino.cc/projecthub/person87/rfid-rc522-sensor-7fc37b
- https://randomnerdtutorials.com/security-access-using-mfrc522-rfid-reader-with-arduino/
- https://lastminuteengineers.com/how-rfid-works-rc522-arduino-tutorial/

Hardware Resources

- http://69.195.111.207/tutorial-download/?t=UNO_R3_Project_The_Most_Complete_Starter_Kit_V2.0
- https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf
- https://www.nxp.com/docs/en/data-sheet/MFRC522.pdf
- http://www.handsontec.com/dataspecs/module/I2C_1602_LCD.pdf
- http://www.ee.ic.ac.uk/pcheung/teaching/DE1_EE/stores/sg90_datasheet.pdf
- https://components101.com/sites/default/files/component_datasheet/Buzzer%20Datasheet.pdf
- https://www.tme.eu/Document/d5041798b41b6ad5e98cd9d1377d272d/INR18650-25R.pdf

Export to PDF