TRANCA DE ARMÁRIOS AUTOMATIZADA

Louise Carvalho Aldrighi
João Lucas Hammes

Estávamos tendo muitos problemas com a organização dos armários do clube de robótica, em que sempre que chegamos no laboratório havia peças e coisas desnecessárias e fora do lugar dentro do armário do clube de Arduino

Nossa solução foi automatizar as trancas de forma com que apenas quem tivesse autorização, pudesse abrir as portas do armário.

Como? Bom foi utilizado um módulo de identificação de rádio frequência que faz uma leitura através de pequenos chipes capazes de armazenarem informações (como os que tem no crachá) . Caso o chip esteja cadastrado com permissão no sistema, será liberado a abertura ou fechamento da tranca.

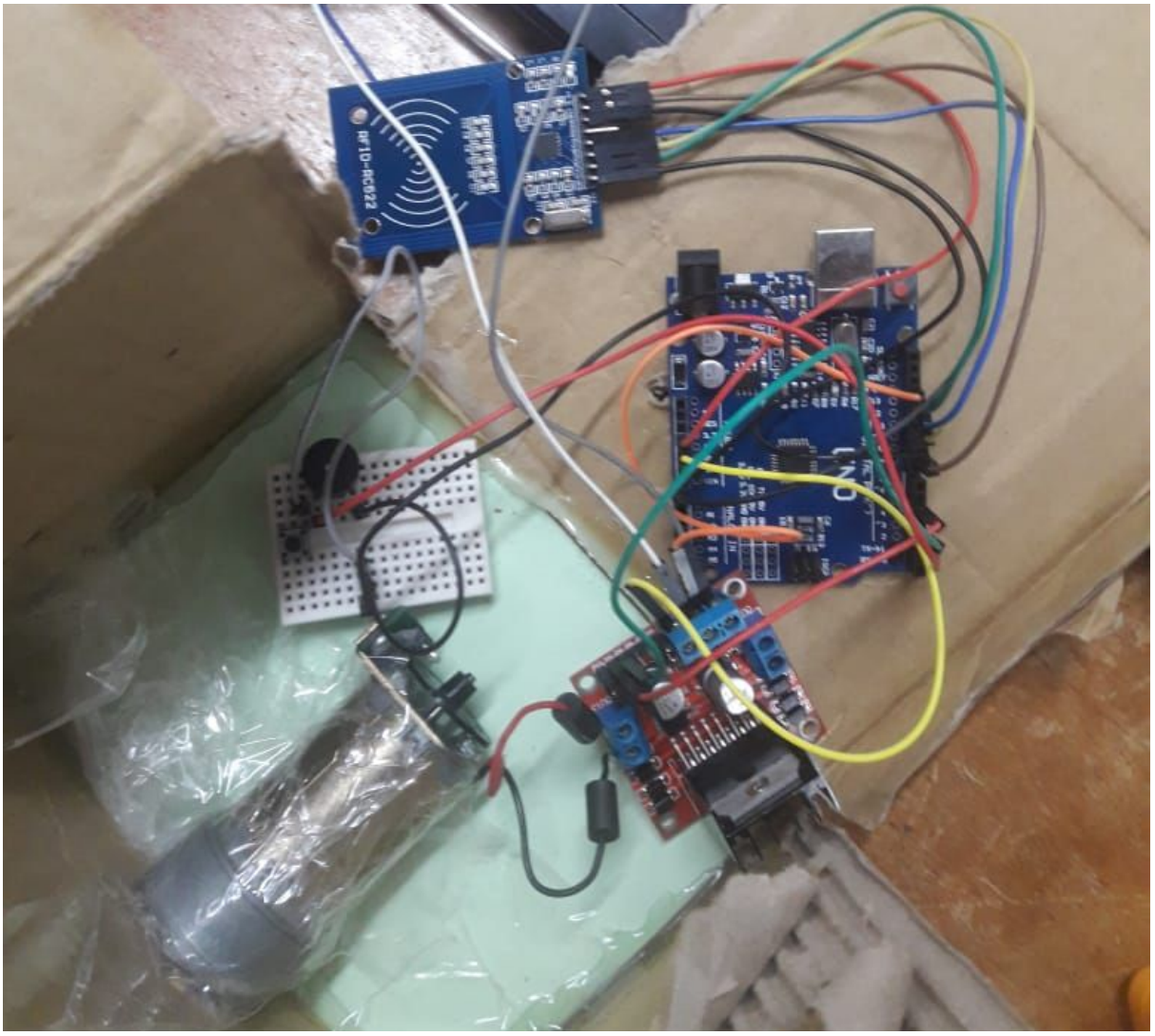
A permissão é dada através de um “master card” que é capaz de habilitar um modo de cadastro e descadastro de cartões do sistema

Peças utilizadas:

- Arduino UNO
- Uma mini photoboard
- RFID RC522
- Um buzzer
- Um motor grande
- Um botão
- Placa de motor DC (Ponte H L298N)

(Infelizmente não havia as peças utilizadas no tinkercad e o fritzing estava MUITO lagado no meu computador, então segue abaixo as fotos do projeto)

Projeto dentro de sua caixa



Projeto de forma aberta

Leitor de chip dos cartões

Caixa onde está o projeto

Código:

```
//bibliotecas
#include <EEPROM.h>
#include <SPI.h>
#include <MFRC522.h>

//variaveis
#define COMMON_ANODE
#define buzzer 2
#define motore 3
#define motor 5
#define wipeB 6

bool programMode = false;

uint8_t successRead;

byte storedCard[4];
byte readCard[4];
byte masterCard[4];

#define SS_PIN 10
#define RST_PIN 9
MFRC522 mfrc522(SS_PIN, RST_PIN);

//botao

bool monitorWipeButton(uint32_t interval) {
  Serial.print("1");
  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)
        Serial.println("botao");
        return false;
    }
  }
```

```
  return true;
}



// pega ID

uint8_t getID() {
    //Serial.println("2");
  if ( ! mfrc522.PICC_IsNewCardPresent()) {
    Serial.println("21");
    return 0;
  }
  if ( ! mfrc522.PICC_ReadCardSerial()) {
    Serial.println("22");
    return 0;
  }
  for ( uint8_t i = 0; i < 4; i++) {  //
    readCard[i] = mfrc522.uid.uidByte[i];
  }
  mfrc522.PICC_HaltA();
  return 1;
}

// le ID

void readID( uint8_t number ) {
  Serial.println("3");
  uint8_t start = (number * 4    ) + 2;
  for ( uint8_t i = 0; i < 4; i++ ) {
    storedCard[i] = EEPROM.read(start + i);
  }
}

//adiciona ID para EEPROM

void writeID( byte a[] ) {
    Serial.println("4");
  if ( !findID( a ) ) {
    uint8_t num = EEPROM.read(0);
    uint8_t start = ( num * 4 ) + 6;
    num++;
    EEPROM.write( 0, num );
    for ( uint8_t j = 0; j < 4; j++ ) {
      EEPROM.write( start + j, a[j] );
    }
```

```
  }
  else {
  }
}

// remove ID do EEPROM

void deleteID( byte a[] ) {
    Serial.println("5");
  if ( !findID( a ) ) {
  }
  else {
    uint8_t num = EEPROM.read(0);
    uint8_t slot;
    uint8_t start;
    uint8_t looping;
    uint8_t j;
    uint8_t count = EEPROM.read(0);
    slot = findIDSLOT( a );
    start = (slot * 4) + 2;
    looping = ((num - slot) * 4);
    num--;
    EEPROM.write( 0, num );
    for ( j = 0; j < looping; j++ ) {
      EEPROM.write( start + j, EEPROM.read(start + 4 + j));
    }
    for ( uint8_t k = 0; k < 4; k++ ) {
      EEPROM.write( start + j + k, 0);
    }
  }
}

bool checkTwo ( byte a[], byte b[] ) {
    Serial.println("6");
  for ( uint8_t k = 0; k < 4; k++ ) {
    if ( a[k] != b[k] ) {     // IF a != b
       return false;
    }
  }
  return true;
}

// acha espaço livre

uint8_t findIDSLOT( byte find[] ) {
    Serial.println("7");
```

```
  uint8_t count = EEPROM.read(0);
  for ( uint8_t i = 1; i <= count; i++ ) {
    readID(i);
    if ( checkTwo( find, storedCard ) ) {
      return i;   //retorna espaço livre
    }
  }
}


//procura ID no EEPROM

bool findID( byte find[] ) {
    Serial.println("8");
  uint8_t count = EEPROM.read(0);
  for ( uint8_t i = 1; i < count; i++ ) {
    readID(i);
    if ( checkTwo( find, storedCard ) ) {
      return true; //se achar, retorna verdadeiro
    }
    else {
    }
  }
  return false; //se não achar, retorna falso
}

//checa se o cartao lido é Master
bool isMaster( byte test[] ) {
    Serial.println("9");
  return checkTwo(test, masterCard);
}


//setup

void setup() {

  //Pin

  pinMode(wipeB, INPUT_PULLUP);
  pinMode(buzzer, OUTPUT);
  pinMode(motor, OUTPUT);
  pinMode(motore, OUTPUT);

  //Protocologo
  Serial.begin(9600);
```

```
  SPI.begin();
  mfrc522.PCD_Init();

  if (digitalRead(wipeB) == LOW) {
    bool buttonState = monitorWipeButton(10000);
    if (buttonState == true && digitalRead(wipeB) == LOW) {
      for (uint16_t x = 0; x < EEPROM.length(); x = x + 1) {
        if (EEPROM.read(x) == 0) {
          //faz nd
        }
        else {
           Serial.println("9");
          EEPROM.write(x, 0);
        }
      }
    }
  }
  if (EEPROM.read(1) != 143) {
    do {
       //Serial.println("8");
      successRead = getID();
    }
    while (!successRead);
    for ( uint8_t j = 0; j < 4; j++ ) {
      EEPROM.write( 2 + j, readCard[j] );
    }
    EEPROM.write(1, 143);
  }
  for ( uint8_t i = 0; i < 4; i++ ) {
    masterCard[i] = EEPROM.read(2 + i);
  }
}

//void

void loop () {
    Serial.println("10");
  do {
    successRead = getID();
    if (digitalRead(wipeB) == LOW) {
      bool buttonState = monitorWipeButton(10000);
      if (buttonState == true && digitalRead(wipeB) == LOW) {
        EEPROM.write(1, 0);
        while (1);
      }
    }
```

```
}
while (!successRead);
if (programMode) {
  if ( isMaster(readCard) ) {
    programMode = false;
    return;
  }
  else {
    if ( findID(readCard) ) {
      deleteID(readCard);
    }
    else {
      writeID(readCard);
    }
  }
}
else {
  if ( isMaster(readCard)) {
    programMode = true;
    Serial.println("master mode");
    uint8_t count = EEPROM.read(0);
  }else{
        if ( findID(readCard) ) {
            Serial.println(F("autorizado"));
            digitalWrite(buzzer, HIGH);
            delay(50);
            digitalWrite(buzzer,LOW);
            delay(50);
            analogWrite(motor, 200);
            delay(500);
            analogWrite(motor, 0);
            delay(500);
            analogWrite(motore, 200);
            delay(600);
            analogWrite(motore, 0);
            delay(1000);

    }
    else {
      Serial.println(  F("nao autorizado"));
      digitalWrite(buzzer, HIGH);
      delay(50);
      digitalWrite(buzzer,LOW);
      delay(50);
      digitalWrite(buzzer, HIGH);
      delay(50);
```

```
        digitalWrite(buzzer,LOW);
        delay(50);

    }

  }
}
  }
```