Card cloner

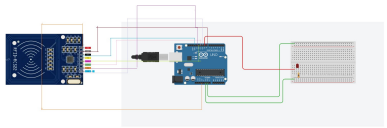
Introducere

Scopul proiectului este realizarea unui copiator de carduri, un aparat care permite citirea unui card/cartele prin intermediul unui modul special pentru scrierea si citirea cardurilor si copierea datelor de pe un card pe altul.. Reusirea citirii cardului va fi semnalata de aprinderea unui LED.

Descriere generală

Prin utilizarea Modulului RFID522 putem citi o cartela/card astfel incat se poate realiza “accesul” respectiv. Modulul va fi conectat la o placa Arduino UNO care va citi semnalul si in cazul in care accesul este permis, un LED se va aprinde.

Hardware Design



Lista componente:

- Arduino UNO
- Modul RFID522
- Jumper-wires
- LED

Software Design

Arduino IDE

Libraries:

1. SPI.h - folosit pentru ca Arduino sa poata comunica cu modulul RFID
2. MFRC522.h - folosit pentru a se putea introduce functii care interactioneaza cu modulul RFID

Cod #include <SPI.h> #include <MFRC522.h>

#define RST_PIN 9 #define SS_PIN 10

MFRC522 mfrc522(SS_PIN, RST_PIN);

byte buffer[18]; byte block; byte waarde[64][16]; MFRC522::StatusCode status;

MFRC522::MIFARE_Key key;

#define NR_KNOWN_KEYS 8

byte knownKeys[NR_KNOWN_KEYS][MFRC522::MF_KEY_SIZE] = {

```
  {0xff, 0xff, 0xff, 0xff, 0xff, 0xff},
  {0xa0, 0xa1, 0xa2, 0xa3, 0xa4, 0xa5},
  {0xb0, 0xb1, 0xb2, 0xb3, 0xb4, 0xb5},
  {0x4d, 0x3a, 0x99, 0xc3, 0x51, 0xdd},
  {0x1a, 0x98, 0x2c, 0x7e, 0x45, 0x9a},
  {0xd3, 0xf7, 0xd3, 0xf7, 0xd3, 0xf7},
  {0xaa, 0xbb, 0xcc, 0xdd, 0xee, 0xff},
  {0x00, 0x00, 0x00, 0x00, 0x00, 0x00}
```

};

char choice;

void setup() {

```
  Serial.begin(9600);
  while (!Serial);
  SPI.begin();
  mfrc522.PCD_Init();
  pinMode(7, OUTPUT);
  Serial.println(F("Try the most used default keys to print block 0 to 63 of
a MIFARE PICC."));
  Serial.println("1.Read card \n2.Write to card \n3.Copy the data.");
```

```
  for (byte i = 0; i < 6; i++) {
      key.keyByte[i] = 0xFF;
  }
```

}

void dump_byte_array(byte *buffer, byte bufferSize) {

```
  for (byte i = 0; i < bufferSize; i++) {
```

```
      Serial.print(buffer[i] < 0x10 ? " 0" : " ");
      Serial.print(buffer[i], HEX);
  }

}
```

void dump_byte_array1(byte *buffer, byte bufferSize) {

```
for (byte i = 0; i < bufferSize; i++) {
  Serial.print(buffer[i] < 0x10 ? " 0" : " ");
  Serial.write(buffer[i]);
}
```

}

bool try_key(MFRC522::MIFARE_Key *key) {

```
  bool result = false;

  for(byte block = 0; block < 64; block++){

  status = mfrc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_A, block,
key, &(mfrc522.uid));
  if (status != MFRC522::STATUS_OK) {
      Serial.print(F("PCD_Authenticate() failed: "));
      Serial.println(mfrc522.GetStatusCodeName(status));
      return false;
  }
```

```
  byte byteCount = sizeof(buffer);
  status = mfrc522.MIFARE_Read(block, buffer, &byteCount);
  if (status != MFRC522::STATUS_OK) {
      Serial.print(F("MIFARE_Read() failed: "));
      Serial.println(mfrc522.GetStatusCodeName(status));
  }
  else {
      result = true;
      Serial.print(F("Success with key:"));
      dump_byte_array((*key).keyByte, MFRC522::MF_KEY_SIZE);
      Serial.println();

      Serial.print(F("Block ")); Serial.print(block); Serial.print(F(":"));
      dump_byte_array1(buffer, 16);
      Serial.println();

      for (int p = 0; p < 16; p++)
      {
        waarde [block][p] = buffer[p];
        Serial.print(waarde[block][p]);
        Serial.print(" ");
      }
```

```
    }
  }
  Serial.println();

  Serial.println("1.Read card \n2.Write to card \n3.Copy the data.");
```

```
  mfrc522.PICC_HaltA();
  mfrc522.PCD_StopCrypto1();
  return result;

  start();
```

}

void loop() {

```
start();
```

}

void start(){

```
choice = Serial.read();

if(choice == '1')
{
  Serial.println("Read the card");
  keuze1();
  digitalWrite(7, HIGH);
  delay(2000);
  digitalWrite(7, LOW);
```

```
  }
  else if(choice == '2')
  {
    Serial.println("See what is in the variables");
    keuze2();
    digitalWrite(7, HIGH);
  delay(2000);
  digitalWrite(7, LOW);
  }
  else if(choice == '3')
  {
    Serial.println("Copying the data on to the new card");
    keuze3();
    digitalWrite(7, HIGH);
  delay(2000);
  digitalWrite(7, LOW);
```

```
  }

}
```

void keuze2(){

```
for(block = 4; block <= 62; block++){
  if(block == 7 || block == 11 || block == 15 || block == 19 || block == 23
|| block == 27 || block == 31 || block == 35 || block == 39 || block == 43
|| block == 47 || block == 51 || block == 55 || block == 59){
    block ++;
  }

Serial.print(F("Writing data into block "));
Serial.print(block);
Serial.println("\n");

  for(int j = 0; j < 16; j++){
    Serial.print(waarde[block][j]);
    Serial.print(" ");
  }
  Serial.println("\n");

}

Serial.println("1.Read card \n2.Write to card \n3.Copy the data.");
start();
```

}

void keuze3(){ Serial.println(“Insert new card…”);

```
  if ( ! mfrc522.PICC_IsNewCardPresent())
      return;
```

```
  if ( ! mfrc522.PICC_ReadCardSerial())
      return;
```

```
  Serial.print(F("Card UID:"));
  dump_byte_array(mfrc522.uid.uidByte, mfrc522.uid.size);
  Serial.println();
  Serial.print(F("PICC type: "));
  MFRC522::PICC_Type piccType = mfrc522.PICC_GetType(mfrc522.uid.sak);
  Serial.println(mfrc522.PICC_GetTypeName(piccType));

  for (byte i = 0; i < 6; i++) {
      key.keyByte[i] = 0xFF;
  }
```

```
for(int i = 4; i <= 62; i++){
  if(i == 7 || i == 11 || i == 15 || i == 19 || i == 23 || i == 27 || i ==
```

```
31 || i == 35 || i == 39 || i == 43 || i == 47 || i == 51 || i == 55 || i ==
59){
    i++;
  }
  block = i;

  Serial.println(F("Authenticating using key A..."));
  status = (MFRC522::StatusCode)
mfrc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_A, block, &key,
&(mfrc522.uid));
  if (status != MFRC522::STATUS_OK) {
      Serial.print(F("PCD_Authenticate() failed: "));
      Serial.println(mfrc522.GetStatusCodeName(status));
      return;
  }

  Serial.println(F("Authenticating again using key B..."));
  status = (MFRC522::StatusCode)
mfrc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_B, block, &key,
&(mfrc522.uid));
  if (status != MFRC522::STATUS_OK) {
      Serial.print(F("PCD_Authenticate() failed: "));
      Serial.println(mfrc522.GetStatusCodeName(status));
      return;
  }

  Serial.print(F("Writing data into block "));
  Serial.print(block);
  Serial.println("\n");

  dump_byte_array(waarde[block], 16);


   status = (MFRC522::StatusCode) mfrc522.MIFARE_Write(block, waarde[block],
16);
    if (status != MFRC522::STATUS_OK) {
      Serial.print(F("MIFARE_Write() failed: "));
      Serial.println(mfrc522.GetStatusCodeName(status));
    }


   Serial.println("\n");

}
mfrc522.PICC_HaltA();
mfrc522.PCD_StopCrypto1();

Serial.println("1.Read card \n2.Write to card \n3.Copy the data.");
start();

}
```

void keuze1(){

```
Serial.println("Insert card...");
  if ( ! mfrc522.PICC_IsNewCardPresent())
      return;
```

```
  if ( ! mfrc522.PICC_ReadCardSerial())
      return;
```

```
  Serial.print(F("Card UID:"));
  dump_byte_array(mfrc522.uid.uidByte, mfrc522.uid.size);
  Serial.println();
  Serial.print(F("PICC type: "));
  MFRC522::PICC_Type piccType = mfrc522.PICC_GetType(mfrc522.uid.sak);
  Serial.println(mfrc522.PICC_GetTypeName(piccType));

  MFRC522::MIFARE_Key key;
  for (byte k = 0; k < NR_KNOWN_KEYS; k++) {
      for (byte i = 0; i < MFRC522::MF_KEY_SIZE; i++) {
          key.keyByte[i] = knownKeys[k][i];
      }
      if (try_key(&key)) {
          break;
      }
  }
```

}

Rezultate Obţinute

S-a reusit citirea, copierea si transferul datelor unui card pe celalalt. Din pacate nu merge si cu cartela de camin caci e pe alta frecventa.

Concluzii

Acest proiect m-a facut sa fiu mai curios despre ce device-uri poti concepe folosind materia predata si in continuare voi face mai mult research pe acest domeniu.

Download

Jurnal

Bibliografie/Resurse

Wikipedia, youtube si niste site-uri mai shady.

Export to PDF