



## Zugangsbeschränkung zu Geräten per Contactless Card mit der NodeMCU und dem RC522 Modul

In dieser mehrteiligen Reihe widmen wir uns den Möglichkeiten der Zugangsbeschränkung zu Geräten oder Systemen mit kontaktlosen Karten der MiFare Reihe unter Nutzung des relativ weit verbreiteten RFID-RC522 Moduls.

In ersten Teil dieser Reihe schauen wir uns an welche Komponenten wir brauchen, wie unsere Hardwaregrundschaltung aussieht und programmieren eine erste Kartenanwendung. In dieser Anwendung wird ein Relais von einer im Programmcode festgelegten autorisierten Karte ein und ausgeschaltet. Durch eine mehrfarbige LED wird angezeigt, ob die Karte gelesene berechtigt ist (war) das Relais zu schalten oder nicht. Durch das Relais können potentialfrei Verbraucher geschaltet werden. Es gilt:

**WARNUNG!! Es besteht beim Schalten von Lasten über 30 Volt oder 230 Volt Netzspannung Lebensgefahr durch einen elektrischen Schlag. Bitte achtet auf eine entsprechende Isolierung und auf die eingängigen Schutzvorschriften und Betriebsvorschriften nach der DIN VDE 0105.**

Insbesondere sind immer die 5 Sicherheitsregeln anzuwenden, die unter [https://de.wikipedia.org/wiki/F%C3%BCnf\\_Sicherheitsregeln](https://de.wikipedia.org/wiki/F%C3%BCnf_Sicherheitsregeln) nachzulesen sind.

Kommen wir aber nun zurück zu unserem Projekt und unserer Teileliste. Ihr braucht folgende Teile:

- [1x NodeMcu](#)
- [1x 2-fach Relais](#)
- [1x RC522 CardReader + Mifare Classic Card](#)
- [1x Netzteiladapter](#)
- [1x RGB LED Modul](#)

Hinweise:

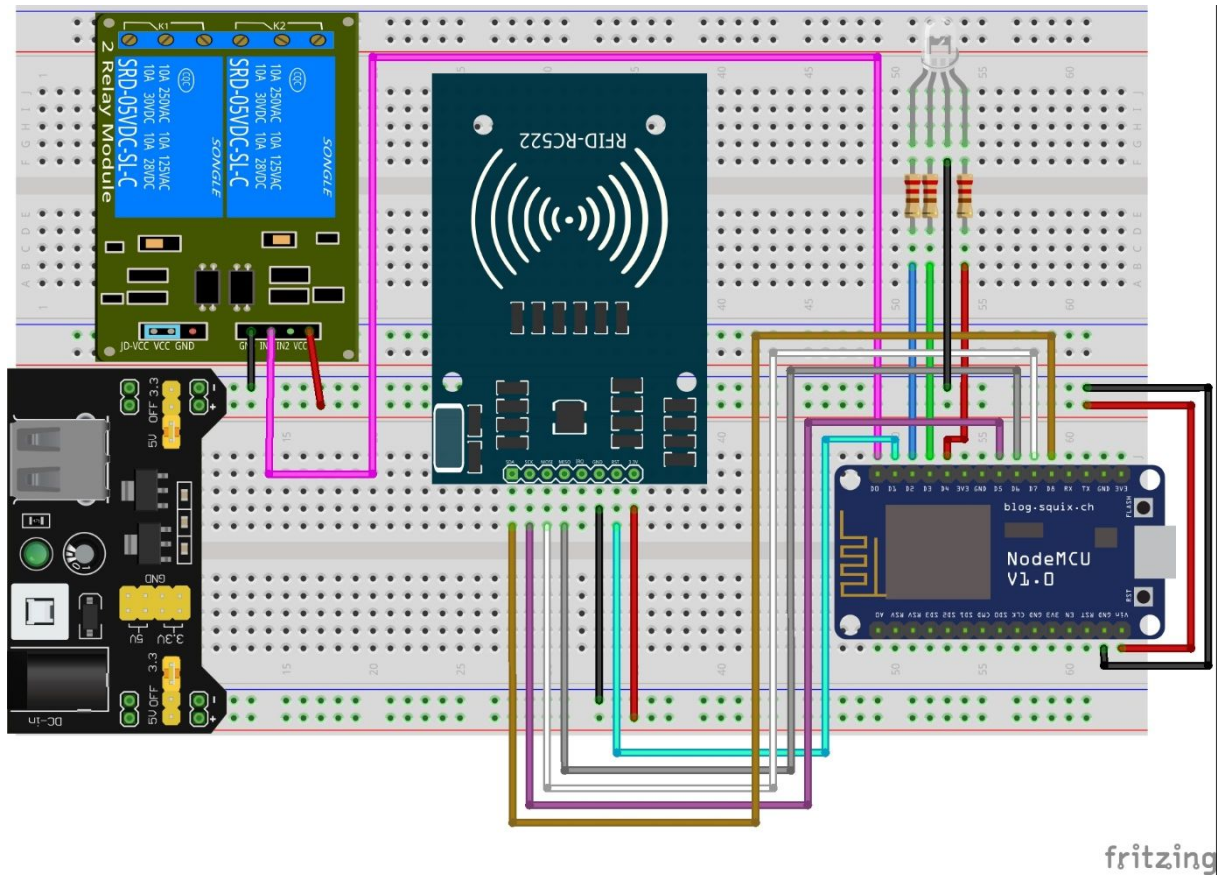
In dem RC522 Cardreaderpaket sind schon MiFare Classic Karten enthalten, die für unser Projekt völlig ausreichend sind. Es ist daher nicht nötig, separate Rfid-Karten zu bestellen.

In dem KY-016 LED RGB Modul sind schon die Vorwiderstände zum direkten Betrieb an den Portpins enthalten. Daher werden diese nicht mehr separat gebraucht. Falls lieber ohne RGB Modul gearbeitet werden soll, werden neben der RGB Led mit gemeinsamer Kathode 3x 220 Ohm Widerstände benötigt.

Das Pin - Mapping für die RGB LED ist:

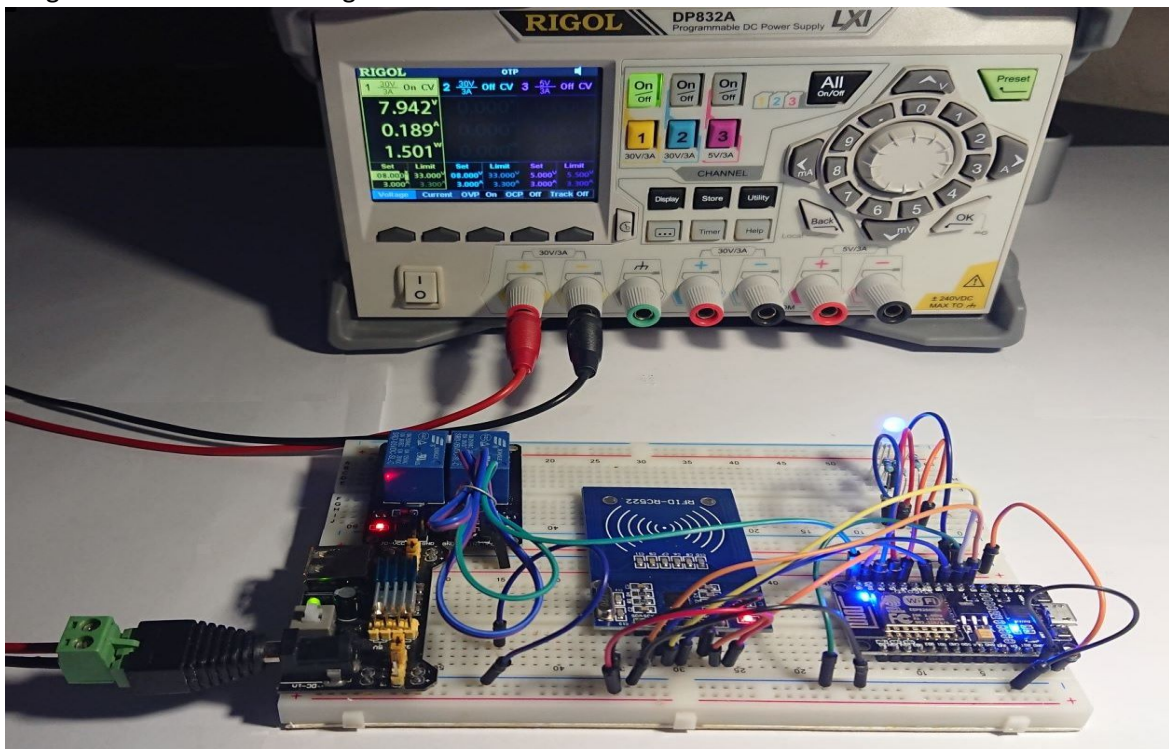
Ausgang NodeMcu	LED Farbe
D2	(B)lau
D3	(G)rün
D4	(R)ot

Auf einem Breadboard werden nun die Komponenten wie folgt miteinander verbunden:



fritzing

Aufgebaut sieht sie Schaltung dann so aus:



Auf dem Bild gut zu erkennen ist der Strombedarf der Schaltung bei aktiviertem Relais. Dieser beträgt bei 8 Volt Versorgungsspannung ca. 190 mA. Ohne aktiviertem Relais beträgt die der Strombedarf ca.

110 mA. Es ist also wichtig unbedingt eine externe Spannungsquelle anzuschließen und sich nicht auf die USB-Stromversorgung zu verlassen.

Nachdem wir alle Komponenten zusammen gebaut haben stellen wir unser Arduino DIE auf den Support des ESP8266 Chips um, wählen unter Werkzeuge -> Board das „Generic ESP8266 Modul“ aus. Danach wählen wir unter Werkzeuge → Port den COM Port aus, an den unsere NodeMcu angeschlossen ist. Wir fügen nun folgenden Code ein:

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

#define RST_PIN 5 // SPI Reset Pin (D1 Ausgang)
#define RELAIS_PIN 16 // Relais (D0 Ausgang) [LOW Aktiv] - Auch interne LED nahe USB Port
#define SS_PIN 15 // SPI Slave Select Pin

#define RGBLED_R 2 // Rot (D4 Ausgang)
#define RGBLED_G 0 // Grün (D3 Ausgang) - Auch interne LED auf dem ESP Modul
#define RGBLED_B 4 // Blau (D2 Ausgang)

#define LED_BUILTIN 16
#define PIN_WIRE_SDA 4
#define PIN_WIRE_SCL 5

MFRC522 mfrc522(SS_PIN, RST_PIN); // Instanz des MFRC522 erzeugen
MFRC522::MIFARE_Key key;

byte myValidCardUID[4] = {0x00,0x00,0x00,0x00}; // Hier die per serielle Schnittstelle gelesene
// UID eintragen, für die der Lesevorgang zukünftig
// gültig sein soll, und das Relais geschaltet werden
// soll.

void setup() {
  pinMode(RST_PIN,OUTPUT);
  pinMode(RELAIS_PIN,OUTPUT);
  pinMode(RGBLED_R,OUTPUT);
  pinMode(RGBLED_G,OUTPUT);
  pinMode(RGBLED_B,OUTPUT);
  digitalWrite(RELAIS_PIN,HIGH); // Relais inaktiv
  digitalWrite(RST_PIN,HIGH);
  digitalWrite(RGBLED_R,LOW); //Led AUS
  digitalWrite(RGBLED_G,LOW);
  digitalWrite(RGBLED_B,LOW);
  Serial.begin(9600); // Serielle Kommunikation mit dem PC initialisieren
  Serial.println("Ser. Komm. OK.");
  SPI.begin(); // Initialisiere SPI Kommunikation
  digitalWrite(RST_PIN,LOW);
  delay(300);
  digitalWrite(RST_PIN,HIGH);
  mfrc522.PCD_Reset();
  mfrc522.PCD_Init(); // Initialisiere MFRC522 Lesemodul
  mfrc522.PCD_AntennaOn();
  yield();
  digitalWrite(RGBLED_R,HIGH); //Led Farbe Lila - Initialisierung abgeschlossen
  digitalWrite(RGBLED_G,LOW);
  digitalWrite(RGBLED_B,HIGH);
}

void loop()
{
  if (mfrc522.PICC_IsNewCardPresent() && mfrc522.PICC_ReadCardSerial()) // PICC = proximity integrated circuit card
  // kontaktlose Chipkarte
  {
    Serial.print("PICC UID:");
    for (byte i = 0; i < mfrc522.uid.size; i++)
    {
      Serial.print(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " ");
      Serial.print(mfrc522.uid.uidByte[i], HEX);
    }
  }
  bool isValid = true;
```

```

for (byte i = 0; i < sizeof(myValidCardUID); i++)
{
  if (mfrc522.uid.uidByte[i] != myValidCardUID[i]) { isValid = false; }
}
if (isValid)
{
  bool PinState= digitalRead(RELAIS_PIN);
  PinState = !PinState;
  digitalWrite(RELAIS_PIN, PinState);
  digitalWrite(RGBLED_R,LOW); //Led Grün
  digitalWrite(RGBLED_G,HIGH);
  digitalWrite(RGBLED_B,LOW);
  Serial.print(" gültig.");
  delay(2000);
  digitalWrite(RGBLED_R,LOW); //Led Farbe Blau Leser ist in Grundzustand
  digitalWrite(RGBLED_G,LOW);
  digitalWrite(RGBLED_B,HIGH);
} else
{
  digitalWrite(RGBLED_R,HIGH); //Led Rot - Letzte Karte war ungültig
  digitalWrite(RGBLED_G,LOW);
  digitalWrite(RGBLED_B,LOW);
  Serial.print(" ungültig.");
  delay(2000);
}
Serial.println();

mfrc522.PICC_HaltA(); // Versetzt die aktuelle Karte in einen Ruhemodus.
delay(1000);
}
yield(); // interne ESP8266 Funktionen aufrufen
}

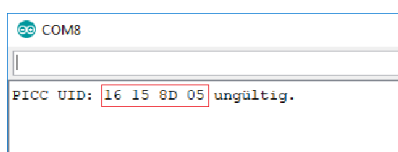
```

Wir kompilieren den Code und laden ihn auf unsere NodeMcu hoch. Während des Hochladens des Sketches leuchtet unsere RGB Led „rot“. Nach erfolgreichem Upload wechselt die Farbe der LED auf „lila“. Durch den Wechsel der Farbe wird angezeigt, dass der Upload Vorgang erfolgreich war.

**Damit ein Hochladen auf die NodeMcu funktioniert, muss während der Kompilierungsvorganges der Taster „Flash“ auf der NodeMcu gedrückt werden und gedrückt gehalten werden!**

Wenn der Upload -Vorgang erfolgreich abgeschlossen wurde, starten wir in der IDE unser Serielles Terminal und stellen die Übertragungsgeschwindigkeit auf 9600 Baud ein.

Nun ist unser Leser bereit die erste Karte einzulesen. Diese wird natürlich nicht als zugangsberechtigt angesehen, da unser Leser ja Ihre ID noch nicht kennt. Die Ausgabe im Terminal sollte dann wie folgt aussehen:



```

COM8
PICC UID: 16 15 8D 05 ungültig.

```

Dies ist eine erwartete Ausgabe. Damit diese Karte von unserem Leser zukünftig als gültig akzeptiert wird, ändern wir nun die Zeile „byte myValidCardUID[4] = {0x00,0x00,0x00,0x00};“ in „byte myValidCardUID[4] = {0x16,0x15,0x8D,0x05};“ kompilieren neu und laden unseren Code auf die Node MCU hoch. Danach wird diese Karte als gültig erkannt und man kann das Relais durch Vorhalten der gültigen Karte umschalten. Dies wird durch ein 2 sekündiges grünes Leuchten der LED bestätigt.

Falls eine ungültige Karte vorgehalten wird, schaltet die LED auf dauerhaft „rot“. Es ist somit erkennbar, ob in der Vergangenheit ein Fehlversuch durch eine nicht autorisierte Karte erfolgte.

Der Aufmerksame Leser wird sich sicherlich fragen, warum wir für diese Aufgabe eine NodeMcu verwenden, da diese Aufgabe z.B. auch durch einen Arduino Uno erfüllt werden könnte. Dies ist zwar prinzipiell richtig, jedoch wollen wir unseren Kartenleser um komfortable Konfigurationsmöglichkeiten per Netzwerk erweitern sowie Sicherheitsfunktionen ergänzen. All das ist aber Teil der kommenden Teile dieser Reihe.

Der Autor empfiehlt das hier beschriebene System nicht als Zugangskontrolle für Türsysteme oder anderen sicherheitskritischen Systemen einzusetzen, da u.a die dem System zugrundeliegende PICC „Mifare Classic“ bereits geknackt wurde und daher als unsicher eingestuft wurde.

Viel Spaß beim Nachbauen.