Zugangsbeschränkung zu Geräten per Contactless Card mit der NodeMCU und dem RC522 Modul
Zweiter Teil – eine Weboberfläche zur Konfiguration

Nachdem wir im ersten Teil die Hardware definiert haben und eine Karte fest zum schalten des Relais verwenden können, spendieren wir unserem ESP im zweiten Teil dieser Reihe eine erste Bedienerweboberfläche , um Konfigurationstasks rund um unseren Kartenleser ausführen zu können. Die Webseite wird von Teil zu Teil dieser Reihe noch ausgebaut und mit mehr Funktionen ausgestattet.

Es gelten die Hinweise des ersten Teils.

Ale erstes müssen wir in folgenden Zeilen in dem Code auf unsere Bedürfnisse, bzw unser lokales WLAN anpassen:

```
WiFi.begin("Deine_WLANSSID","Dein_Passwort");
```

Wir laden den folgenden, nun angepassten Code auf unsere NodeMcu hoch:

```
#include <SPI.h>
#include <MFRC522.h>
#include <ESP8266WiFi.h>
//#include <WiFiClient.h>
#include <ESP8266WebServer.h>
#include <ESP8266mDNS.h>
#include <EEPROM.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 WiFiPwdLen   25   // Maximale WiFi Passwortlänge
#define STANameLen 20     // Maximale WiFi SSIDlänge
#define ESPHostNameLen 20 // Maximale Anzahl Zeichen ESPHostName

#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;
ESP8266WebServer server(80);       // Web Server Instanz erzeugen

struct WiFiEEPromData
 {
  char ESPHostName[ESPHostNameLen];
  char APSTAName[STANameLen]; // STATION Name TO Connect, if definded
  char WiFiPwd[WiFiPwdLen]; // WiFiPAssword, if definded
  char ConfigValid[3]; //If Config is Vaild, Tag "TK" is required"
 };

struct PICCardEEPRomData
 {
  char CardVaildUID[4];
  char Reservered[4];
 };

WiFiEEPromData MyWiFiConfig;
PICCardEEPRomData MyEEPROMValidCardUID;
bool Result  = false;
bool LearnNewCard = false;

void setup()
 {
 pinMode(RST_PIN,OUTPUT);
 digitalWrite(RST_PIN,HIGH);
 pinMode(RELAIS_PIN,OUTPUT);
 pinMode(RGBLED_R,OUTPUT);
 pinMode(RGBLED_G,OUTPUT);
 pinMode(RGBLED_B,OUTPUT);
 digitalWrite(RELAIS_PIN,HIGH);    // Relais inaktiv
 digitalWrite(RGBLED_R,LOW);      //Led AUS
 digitalWrite(RGBLED_G,LOW);
 digitalWrite(RGBLED_B,LOW);
 Serial.begin(9600);             // Serielle Kommunikation mit dem PC initialisieren
 yield();
 Serial.println(ESP.getChipId());
 SPI.begin();                    // Initialisiere SPI Kommunikation
 digitalWrite(RST_PIN,LOW);
 Result  = startWiFiClient();
 yield();
 EEPROM.begin(512);
 EEPROM.get(100,MyEEPROMValidCardUID);   // Ab Adresse 100 wird die gültige KArte abgelegt
 EEPROM.end();
 InitalizeHTTPServer();
 digitalWrite(RST_PIN,HIGH);
 mfrc522.PCD_Reset();
 mfrc522.PCD_Init();             // Initialisiere MFRC522 Lesemodul
 mfrc522.PCD_AntennaOn();
 yield();
```

```
  digitalWrite(RGBLED_R,HIGH);     //Led Farbe Lila Initalisierung abgeschlossen
  digitalWrite(RGBLED_G,LOW);
  digitalWrite(RGBLED_B,HIGH);
  }

// ******************* Start Functions Webserver
*****************************************
void handleNotFound()
  {
  }
void handleRoot()   // Hauptseite
  {
  String temp ="";
  if (server.hasArg("AuthCard") )  // Reboot System
    {
    temp = "Bitte Karte vor Leser vorhalten.";
    server.send ( 200, "text/html", temp );
    digitalWrite(RGBLED_R,HIGH);     //Led Farbe Gelb Programmierungsmodus
    digitalWrite(RGBLED_G,HIGH);
    digitalWrite(RGBLED_B,LOW);
    LearnNewCard = true;
    server.client().stop();
    }
  server.sendHeader("Cache-Control", "no-cache, no-store, must-revalidate");
  server.sendHeader("Pragma", "no-cache");
  server.sendHeader("Expires", "-1");
  server.setContentLength(CONTENT_LENGTH_UNKNOWN);
  // HTML Content
  server.send ( 200, "text/html", temp );   // Speichersparen - Schon mal dem Cleint senden
  temp = "";
  temp += "<!DOCTYPE HTML><html lang='de'><head><meta charset='UTF-8'><meta name=
viewport content='width=device-width, initial-scale=1.0,'>";
  server.sendContent(temp);
  temp = "";
  temp += "<style type='text/css'><!-- DIV.container { min-height: 10em; display: table-cell; vertical-
align: middle }.button {height:35px; width:90px; font-size:16px}";
  server.sendContent(temp);
  temp = "";
  temp += "body {background-color: powderblue;}</style>";
  temp += "<head><title>Kartenleser RC522</title></head>";
  temp += "<h2>Kartenleser RC522 Konfiguration</h2>";
  temp += "<body>";
  server.sendContent(temp);
  temp = "";
  temp += "<form action='/' method='post'>";
  temp += "<button type='submit' name='AuthCard' value='1' style='height: 50px; width: 200px'
>Authorisiere nächste Karte an Leser</button>";
  temp += "</form>";
  server.sendContent(temp);
  temp = "";
  server.client().stop();
  }
```

```
void InitalizeHTTPServer()
 {
 bool initok = false;
 server.on("/", handleRoot);
 server.onNotFound ( handleNotFound );
 server.begin(); // Web server start
 }
// ******************* End Functions Webserver
*****************************************

// ******************* Start Functions WiFi Management
************************************

bool startWiFiClient() {
 bool WiFiClientStarted = false;
 size_t A0_ADCValue = 0;
 byte i = 0;
 byte connRes = 0;
 //Serial.setDebugOutput(true);  // Zu Debugzwecken aktivieren.
 WiFi.hostname("CrdRdr41667");
 WiFi.softAPdisconnect(true);
 WiFi.disconnect();
 WiFi.mode(WIFI_STA);
 if(loadCredentials())
  {
   WiFi.begin(MyWiFiConfig.APSTAName, MyWiFiConfig.WiFiPwd);
   while (( connRes != 3 ) and( connRes != 4 ) and (i != 30))  //if connRes == 0  "IDLE_STATUS -
change Statius"
    {
    i++;
    // Serial.print("."); // Connect vorgang auf der seriellen Schnittstelle beobachten
    delay(500);
    yield();
    connRes  = WiFi.waitForConnectResult();
    }
   if (connRes == 4 ) { // if password is incorrect
    Serial.println("ATWIFI:PWDERR");
    WiFi.disconnect();
    }
   if (connRes == 6 ) { //  module is not configured in station mode
    Serial.println("ATWIFI:STAERR");
    WiFi.disconnect();
    }
  }
 if(WiFi.status() == WL_CONNECTED)
  {
  Serial.print("ATIP:");
  Serial.println(WiFi.localIP());
  WiFi.setAutoReconnect(true); // Set whether module will attempt to reconnect to an access
point in case it is disconnected.
  // Setup MDNS responder
  if (!MDNS.begin("CrdRdr41667"))
   {
```

```
     Serial.println("ATMDNS:NOK");
     } else { MDNS.addService("http", "tcp", 80); }
    WiFiClientStarted = true;
    } else
    {
    WiFi.disconnect();
    WiFi.mode(WIFI_STA);
    WiFi.begin("Deine_WLANSSID","Dein_Passwort"); // Ändern bzw. anpassen!
    saveCredentials();
     while (( connRes != 3 ) and( connRes != 4 ) and (i != 30))  //if connRes == 0  "IDLE_STATUS -
change Statius"
     {
     i++;
     // Serial.print("."); // Connect vorgang auf der seriellen Schnittstelle beobachten
     delay(500);
     yield();
     connRes  = WiFi.waitForConnectResult();
     }
  }
  //WiFi.printDiag(Serial);      // Zu Debugzwecken aktivieren.
  return WiFiClientStarted;
}
// ******************* END Functions WiFi Management
************************************

// ******************* Start Functions Store WiFi Credentials to EEPROM ******************
bool loadCredentials()
{
 bool RetValue;
 EEPROM.begin(512);
 EEPROM.get(0,MyWiFiConfig);
 EEPROM.end();
 if (String(MyWiFiConfig.ConfigValid) == "TK")
  {
    RetValue = true;
  } else
  {
    RetValue = false; // WLAN Settings nicht gefunden.
  }
  return RetValue;
}

void saveCredentials() // Speicehre WLAN credentials auf EEPROM
{
  String A;
  size_t i;
  for (i = 0 ; i < sizeof(MyWiFiConfig) ; i++) // Loeschen der alten Konfiguration
    {
     EEPROM.write(i, 0);
    }
  for (i = 0 ; i < STANameLen  ; i++) // Loeschen der alten Konfiguration
    {
     MyWiFiConfig.WiFiPwd[i] = 0;
```

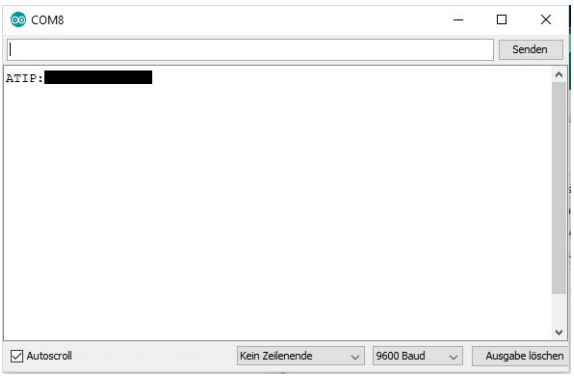
```
    }
  for (i = 0 ; i < WiFiPwdLen ; i++) // Loeschen der alten Konfiguration
    {
     MyWiFiConfig.APSTAName[i] = 0;
    }
  A = WiFi.SSID().c_str();
  i = A.length();
  A.toCharArray(MyWiFiConfig.APSTAName,i+1);
  A = WiFi.psk().c_str();
  i = A.length();
  A.toCharArray(MyWiFiConfig.WiFiPwd,i+1);
  strncpy(MyWiFiConfig.ConfigValid , "TK", sizeof(MyWiFiConfig.ConfigValid) );
  EEPROM.begin(512);
  EEPROM.put(0, MyWiFiConfig);
  EEPROM.commit();
  EEPROM.end();
}
// ******************** END Functions StoreCredentialsto EEPROM
**************************

void loop()  // Hauptschleife
{
  // Nur wenn eine Karte gefunden wird und gelesen werden konnte, wird der Inhalt von IF
ausgeführt
  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++)
    {
      // Abstand zwischen HEX-Zahlen und führende Null bei Byte < 16
      Serial.print(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " ");
      Serial.print(mfrc522.uid.uidByte[i], HEX);
    }
  bool IsValid = true;
  if (LearnNewCard)
  {
   for (byte i = 0; i < sizeof(MyEEPROMValidCardUID.CardVaildUID); i++)
    {
     MyEEPROMValidCardUID.CardVaildUID[i] = mfrc522.uid.uidByte[i];
     EEPROM.begin(512);
     EEPROM.put(100,MyEEPROMValidCardUID);
     EEPROM.commit();
     EEPROM.end();
     LearnNewCard = false;
    }
   IsValid = true;
  } else
  {
  for (byte i = 0; i < sizeof(MyEEPROMValidCardUID.CardVaildUID); i++)
    {
    if (mfrc522.uid.uidByte[i] != MyEEPROMValidCardUID.CardVaildUID[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();
    // Versetzt die gelesene Karte in einen Ruhemodus, um nach anderen Karten suchen zu können.
    mfrc522.PICC_HaltA();
    delay(1000);
  }
  server.handleClient(); // Webserveranfragen bearbeiten
  yield();              // interne ESP8266 Funktionen aufrufen
  delay(20);
}
```

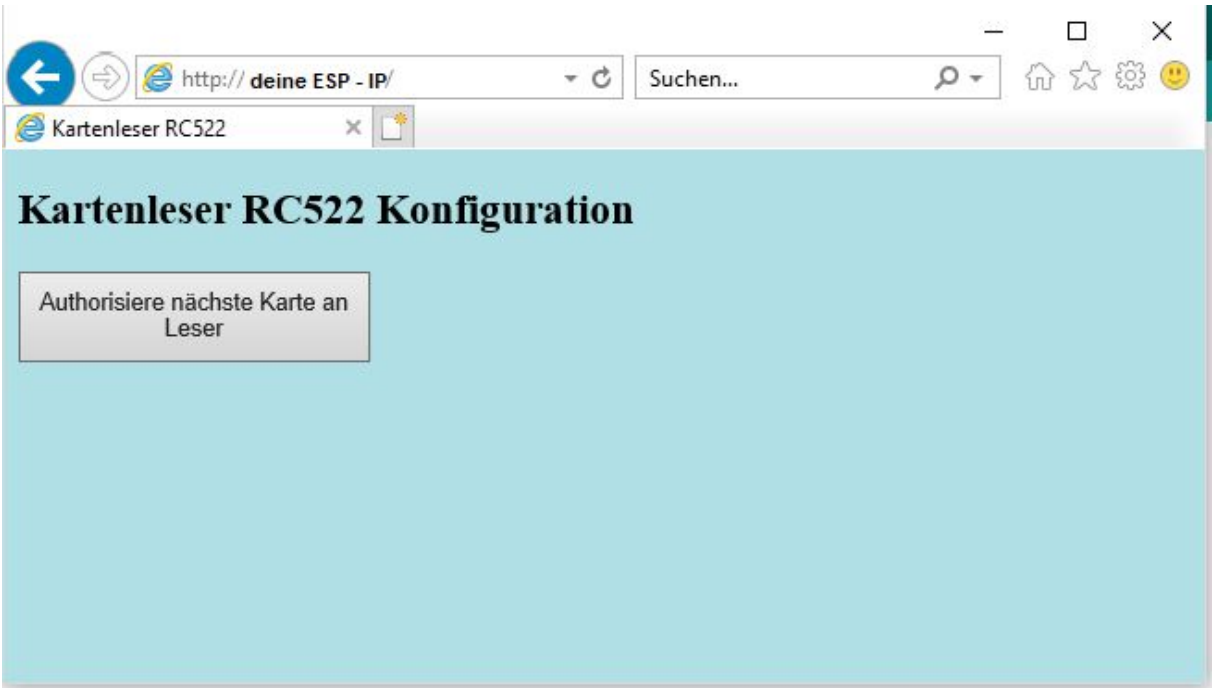
Wir kompilieren den Code und laden Ihn auf unsere NodeMcu hoch. Während des Hochladens des Sketches leuchtet unsere RGB Led „rot“. Nach erfolgreichem Upload wechselt die Farbe der LED nach einiger Zeit auf „lila“. Durch den Wechsel der Farbe wird angezeigt, dass der Upload Vorgang erfolgreich war.

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

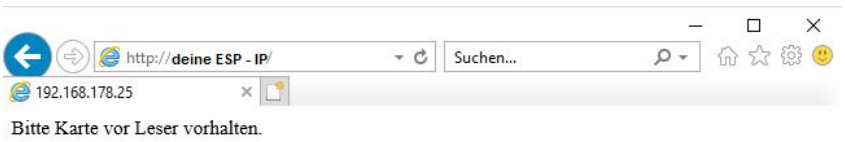
Wenn der Upload -Vorgang erfolgreich abgeschlossen wurde, starten wir in der IDE unser serielles Terminal und stellen die Übertragungsgeschwindigkeit auf 9600 Baud ein. Wir bekommen eine Ausgabe, die unsere IP Adresse im lokalen Netzwerk ausgibt:



Die angezeigte Adresse pingen wir an um testen, ob alles geklappt hat. Wenn wir eine Antwort auf unsere Pings bekommen, gehen wir auf die Webseite des ESP's indem wir die oben angezeigte IP-Adresse in den Browser eintippen:



Da unser ESP noch keine autorisierte Karte im internen EEPROM gespeichert hat, klicken wir nun auf den Button. Wir erhalten folgenden Hinweis:



Die LED leuchtet Gelb, um anzuzeigen das eine Karte vor den Leser gehalten werden kann. Die nächste Karte, die in diesem Modus vor den Leser gehalten wird, wird akzeptiert und als autorisiert gespeichert. Die vorhergehende autorisierte Karte wird dabei gelöscht. Es kann immer nur eine Karte autorisiert sein.

Im nächsten Teil werden wir einen Taster an unsere NodeMcu anschließen mit dem wir einen WPS Schlüsselaustausch automatisch mit unserem Router vornehmen können und die WLAN Zugangsdaten dauerhaft in unser EEPROM schreiben. Somit ist auch der letzte Teil an fester Konfiguration aus unserem Code dynamisiert. Der Code kann somit, einmal kompiliert, mit beliebigen Routern und Karten verwendet werden. Wir werden uns weiterhin das interne FileSystem des ESP zu Nutze machen um mehrere Karten über die Weboberfläche verwalten zu können.

Bis zum nächsten Teil