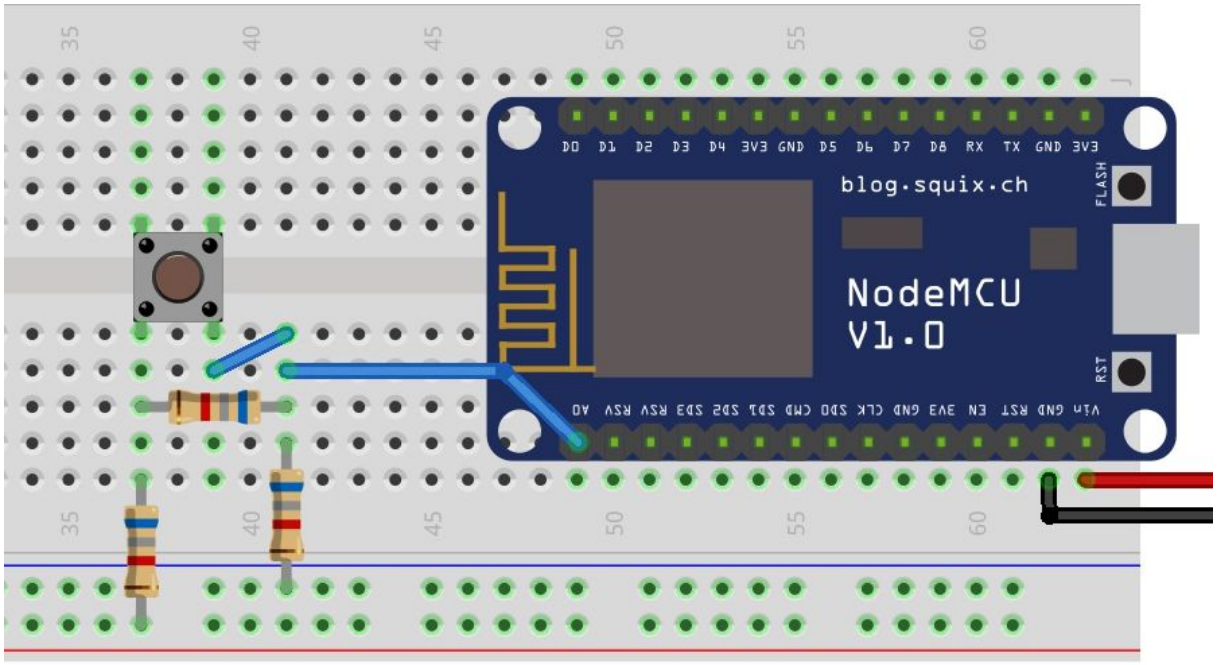
Zugangsbeschränkung zu Geräten per Contactless Card mit der NodeMCU und dem RC522 Modul dritter Teil – WPA ,eine Weboberfläche mit Passwortschutz, Statusinformationen und einer erweiterbaren Menüleiste.

Nachdem wir im ersten Teil die Hardware definiert haben und eine Karte fest zum schalten des Relais verwenden können, spendieren wir unserem ESP im dritten Teil dieser Reihe einen WPS Push Button, um die WLAN Konfiguration automatisch übernehmen zu könmnen, sowie eine Anmeldeseite für unseren ESP, um die Konfiguration gegen unbefungtes Ändern zu schützen.

WPA Anmeldung:

Um zuküftig die WLAN Daten über die WPA Push Button Methode in unserem ESP speichern zu können, müssen wir unsere im Teil eins vorgestellte Grundschaltung etwas erweitern.



Wir brauchen dazu 3 präzise Wiederstände mit 6,8 KOhm und einen Taster. Dieser Taster schließt den mittleren Wiederstand des Spannugnsteilers kurz, damit unser ESP den Tastendruck regisitrieren kann. Oben genante Schaubild demonstriert die notwendige Verschaltung, und muss unserer Schaltung aus dem ersten Aritkel HINZUGEFÜGT werden.

WARNUNG!! Es besteht beim Schalten von Lasten über 30 Volt oder 230 Volt Netzspannung Lebensgefahr durch einen elektrischen Schlag. Arbeiten dürfen nur durch qualifizierte Elektrofachkräfte durchgeführt werden!

Wir kopieren nun folgenden Code in unsere IDE:

```
#include <SPI.h>
#include <MFRC522.h>
#include <ESP8266WiFi.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

ADC_MODE(ADC_TOUT);   // Analogeingang A0 auf extern konfigurieren. ADC_TOUT (for external voltage), ADC_VCC (for
system voltage).
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 /AP Point 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 Reserved[4];
 };

WiFiEEPromData MyWiFiConfig;
PICCardEEPRomData MyEEPROMValidCardUID;

// Global Genutzte Variable
bool Result  = false;
bool LearnNewCard = false;
String temp ="";
unsigned long SessionID;

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
 SetRGBLed(0,0,0,false);           //Led AUS
 Serial.begin(115200);             // Serielle Kommunikation mit dem PC mit 115200 Baud initialisieren
 yield();
 Serial.println("");
 temp = "ATSN:"+ String(ESP.getChipId());
 Serial.println(temp);
 // Serial.print("ADC Value:");Serial.println(analogRead(A0));
 SPI.begin();                      // Initialisiere SPI Kommunikation
 ESP.wdtEnable(WDTO_4S);           // Starte Watchdog
 ESP.wdtFeed();
 digitalWrite(RST_PIN,LOW);
```

```
  SessionID = millis();
  ESP.wdtFeed();
  Result  = startWiFiClient();
  ESP.wdtFeed();
  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();
  ESP.wdtFeed();
  SetRGBLed(255,0,255,false);       //Led Farbe Lila Initalisierung abgeschlossen
  }


// ******************* Start Helper/ Optimization Functions *********************************

void SetRGBLed(byte RedValue,byte GreenValue,byte BlueValue,boolean SlowFade)  //Funkion zur Steuerung der RGB Led
{
  digitalWrite(RGBLED_R,LOW);
  digitalWrite(RGBLED_G,LOW);
  digitalWrite(RGBLED_B,LOW);
  if (RedValue == 255)   { digitalWrite(RGBLED_R,HIGH); }
  if (GreenValue == 255) { digitalWrite(RGBLED_G,HIGH); }
  if (BlueValue == 255)  { digitalWrite(RGBLED_B,HIGH); }
}

// ******************* Stop Helper/ Optimization Functions *********************************

// ******************* Start Functions Webserver ***********************************************

//Cookie Basisroutinen basieren auf GIT Auszug:
//https://github.com/esp8266/ESPWebServer/blob/master/examples/SimpleAuthentification/SimpleAuthentification.ino
bool is_authentified()
{
   if (server.hasHeader("Cookie")){
     // Cookie gefunden
     temp = server.header("Cookie");
     //Serial.println(temp);
     String SessionStr = String(ESP.getChipId()) + "=" + String(SessionID);
     if (temp.indexOf(SessionStr) != -1) {
       // Web Authentification erfolgreich
       temp = "";
       return true;
     }
   }
   // Web Authentification fehlgeschlagen
   temp = "";
   SessionID = millis();
   return false;
}

void handleLogin(){
  String msg;
  //String cookie = server.header("Cookie");
  //Serial.println(cookie);
  if (server.hasArg("DISCONNECT")){
    //Disconnection Benutzers;
    server.sendHeader("Location","/login");
    server.sendHeader("Cache-Control","no-cache");
    SessionID = millis();
    temp = String(ESP.getChipId()) + "= NA ; HttpOnly ; SameSite=Strict";
    server.sendHeader("Set-Cookie",temp);
    temp = "";
    server.send(301);
    return;
  }
  if (server.hasArg("USERNAME") && server.hasArg("PASSWORD")){
    temp = String(ESP.getChipId());
    if (server.arg("USERNAME") == "admin" &&  server.arg("PASSWORD") == temp ){
      server.sendHeader("Location","/");
      server.sendHeader("Cache-Control","no-cache");
      SessionID = millis();
```

```
      temp = String(ESP.getChipId()) + "=" + String(SessionID) + "; HttpOnly ; SameSite=Strict";
      server.sendHeader("Set-Cookie",temp);
      temp = "";
      server.send(301);
      return;
    }
  msg = "<script>alert('Falscher Benutzername oder falsches Passwort !');</script>";
  }
   CSS_Header_Template();
   temp = "<head><title>Login</title></head><body><DIV ALIGN=CENTER>";
   server.sendContent(temp);
   temp = "<h2>Anmeldung an Kartenleser RC522</h2><body><br><br><table border=0 bgcolor=black><tr><th><DIV
ALIGN=RIGHT>";
   server.sendContent(temp);
   temp = "<form action='/login' method='post'>Benutzername: <input type=text Name='USERNAME' Size=17 required><br>";
   server.sendContent(temp);
   temp = "Passwort: <input type=password Name='PASSWORD' Size=17 required><br><br><br><button type='submit' ";
   server.sendContent(temp);
   temp = "name='Login_Button' value='1' style='height: 30px; width: 100px'
>Login</button><br></th></tr></form></DIV></table>";
   server.sendContent(temp);
   temp = "<br><SMALL>Damit die Anmeldung funktioniert, sind Cookies für diese Webseite zu erlauben.</SMALL>";
   server.sendContent(temp);
   temp = msg + "</DIV></body></HTML>";
   server.sendContent(temp);
   temp = "";
}

void handleNotFound()
  {
  SessionID = millis();
  temp = "Seite nicht gefunden.\n\n";
  temp += "URI: ";
  temp += server.uri();
  temp += "\nMethod: ";
  temp += (server.method() == HTTP_GET)?"GET":"POST";
  temp += "\nArguments: ";
  temp += server.args();
  temp += "\n";
  for (uint8_t i=0; i<server.args(); i++){
    temp += " " + server.argName(i) + ": " + server.arg(i) + "\n";
  }
  server.send(404, "text/plain", temp);
  temp = "";
  }


void handleNewPICC()
{
  if (!is_authentified())
    {
    server.sendHeader("Location","/login");
    server.sendHeader("Cache-Control","no-cache");
    server.send(301);
    return;
    }
  CSS_Header_Template();
  temp = "<head><title>Kartenleser RC522</title></head><body>";
  server.sendContent(temp);
  HtmlNavStructure();
  temp = "<script>alert('Bitte JETZT Karte vor den Leser halten!');</script>";
  server.sendContent(temp);
  SetRGBLed(255,255,0,false);       //Led Farbe Gelb Programmierungsmodus
  LearnNewCard = true;
  temp = "</body></html>";
  server.sendContent(temp);
  server.client().stop();
  temp = "";
}



void handleRoot(){
  if (!is_authentified()){
    server.sendHeader("Location","/login");
    server.sendHeader("Cache-Control","no-cache");
    server.send(301);
```

```
    return;
    }
  // HTML Content
  CSS_Header_Template();
  temp = "<head><title>Kartenleser RC522</title></head><body>";
  server.sendContent(temp);
  HtmlNavStructure();
  temp = "<div ALIGN=CENTER><br><br><br><br><BIG>Willkommen auf der Smartkartenleser RC522
Webseite.</BIG><br>";
  server.sendContent(temp);
  temp = "Resetgrund: " + String(ESP.getResetReason()) + "<br>";
  server.sendContent(temp);
  temp = "Freier Heapspeicher: " + String(ESP.getFreeHeap()) + " Bytes<br>";
  server.sendContent(temp);
  temp = "Int. Flash: " + String(ESP.getFlashChipRealSize()) + " Bytes<br>";
  server.sendContent(temp);
  Result = mfrc522.PCD_PerformSelfTest();
  mfrc522.PCD_Init();                    // Initialisiere MFRC522 Lesemodul
  mfrc522.PCD_AntennaOn();
  if (Result) {temp = "RC522 PCD-Status: OK<br>"; } else {temp = "RC522 PCD-Status: Fehler!<br>"; }
  server.sendContent(temp);
  temp = "CPU ID: " + String(ESP.getChipId()) + " @ " + String(ESP.getCpuFreqMHz()) + " MHz<br>";
  server.sendContent(temp);
  temp = "<br>Sie sind erfolgreich angemeldet !<br><br><form action='/login' method='get'>";
  server.sendContent(temp);
  temp = "<button type='submit' name='DISCONNECT' value='YES' style='height: 30px; width: 200px' >Logout</button>";
  server.sendContent(temp);
  temp = "</form></div></body></html>";
  server.sendContent(temp);
  if (server.hasArg("Reboot") )  // Reboot System
    {
    ESP.wdtFeed();
    ESP.wdtDisable();
    temp = "<script>alert('Das System startet JETZT neu.');</script>";
    server.sendContent(temp);
    server.client().stop();
    temp = "";
    delay(1000);
    ESP.reset();
    delay(4000);
    }
  server.client().stop();
  temp = "";
}

void CSS_Header_Template() // Formatvorlage für alle internen ESP Webseiten. https://wiki.selfhtml.org/wiki/CSS
  {
   server.setContentLength(CONTENT_LENGTH_UNKNOWN);
   temp = "<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.01 Transitional//EN'><html lang='de'><meta charset='UTF-
8'>";
   server.send (200, "text/html", temp);
   temp = "<style type='text/css'>*{margin: 0;padding: 0;}body{background:black;color:darkorchid;font-size: 16px;";
   server.sendContent(temp);
   temp = "font-family: sans-serif,arial;}.nav{width: 1300px;height: 30px;margin: 0 auto;border-radius: 5px;}";
   server.sendContent(temp);
   temp = "ul li{list-style: none;width: 200px;line-height: 60px;position: relative;background: darkorchid;";
   server.sendContent(temp);
   temp = "box-shadow: 0px 2px 5px 0px grey;text-align: center;float: left;background-color: #010000;}ul li ul{";
   server.sendContent(temp);
   temp = "position: absolute;}.nav > ul > li:nth-of-type(1){border-radius: 5px 0px 0px 5px;}.nav > ul > li:nth-of-type(5)";
   server.sendContent(temp);
   temp = "{border-radius: 0px 5px 5px 0px;}ul li a{color: rgb(182, 18, 18);width: 200px;height: 58px;display: inline-block;";
   server.sendContent(temp);
   temp = "text-decoration: none;}ul li a:hover{font-weight: bold;border-bottom: 2px solid #fff;}ul li ul{display: none;}";
   server.sendContent(temp);
   temp = ".nav ul li:hover ul{display: block;}.fa{margin-right: 5px;}.container{width: 1000px;height: 200px;";
   server.sendContent(temp);
   temp = "margin: 0 auto;padding:20px 20px;}@media screen and (max-width: 480px){header{width: 100%;}";
   server.sendContent(temp);
   temp = ".nav{display: none;width: 100%;height: auto;}ul li{width: 100%;float: none;}ul li a{width: 100%;";
   server.sendContent(temp);
   temp = "display: block;}ul li ul{position: static;}ul li ul li a{background: #222;}.fa-list.modify{display: block;}";
   server.sendContent(temp);
   temp = ".container{width: 100%;height: auto;}body{overflow-x:hidden;}}</style>";
   server.sendContent(temp);
   temp = "";
  }
```

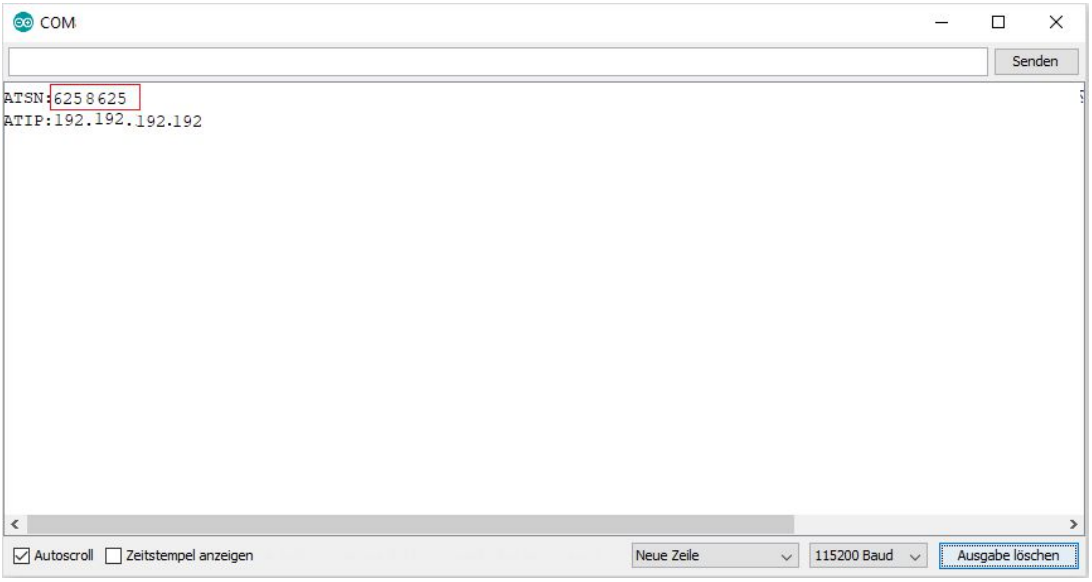
```
void HtmlNavStructure()
  {
  temp = "<div class='menu'><nav class='nav'><ul>";
  server.sendContent(temp);
  temp = "<li><a href='#'>System</a>";
  server.sendContent(temp);
  temp = "<ul><li><a href='/'>Information</a></li>";
  server.sendContent(temp);
  temp = "<li><a href='/?Reboot=YES'>Neustart</a></li>";
  server.sendContent(temp);
  temp = "</ul>";
  server.sendContent(temp);
  temp = "</li><li><a href='#'>PICC</a>";
  server.sendContent(temp);
  temp = "<ul><li><a href='/newPICC'>Karte registrieren</a></li></ul>";
  server.sendContent(temp);
  temp = "</li>";
  server.sendContent(temp);
  temp = "</ul></nav></div>";
  server.sendContent(temp);
  temp = "";
  }


void InitalizeHTTPServer()
  {
  bool initok = false;
  const char * headerkeys[] = {"User-Agent","Cookie"} ; //Header zum Tracken
  size_t headerkeyssize = sizeof(headerkeys)/sizeof(char*); //Header zum Tracken
  server.on("/", handleRoot);
  server.on("/login", handleLogin);
  server.on("/newPICC", handleNewPICC);
  server.onNotFound ( handleNotFound );
  server.collectHeaders(headerkeys, headerkeyssize );// Server anweisen, diese zu Tracken
  server.begin(); // Web server start
  }

// ******************* End Functions Webserver ********************************************

// ******************* Start Functions WiFi Management ************************************
// Funktion von https://www.az-delivery.de/blogs/azdelivery-blog-fur-arduino-und-raspberry-pi/wps-mit-dem-esp8266?ls=de
bool startWPS() {
  bool wpsSuccess = WiFi.beginWPSConfig();
  if(wpsSuccess) {
      // Muss nicht immer erfolgreich heißen! Nach einem Timeout ist die SSID leer
      String newSSID = WiFi.SSID();
       if(newSSID.length() > 0) {
        // Nur wenn eine SSID gefunden wurde waren wir erfolgreich
        yield();
        Serial.println("ATWPS:OK");
        saveCredentials(); // Save Credentials to EEPROM
      } else {
        Serial.println("ATWPS:NOK");
      }
  }
  return wpsSuccess;
}

bool startWiFiClient() {
  bool WiFiClientStarted = false;
  size_t A0_ADCValue = 0;
  byte i = 0;
  byte connRes = 0;
  Serial.setDebugOutput(false);  // 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
      ESP.wdtFeed();
```

```
      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)
    {
    ESP.wdtFeed();
    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
    {
    A0_ADCValue = analogRead(A0);
    //Wir waren nicht erfolgreich, daher starten wir WPS, wenn WPS Taster an A0 während des Resets gedrückt ist
    if (A0_ADCValue > 499)
      {
        if(startWPS())
          {
            ESP.wdtFeed();
            delay(500);
            WiFi.disconnect();
            WiFi.mode(WIFI_STA);
            WiFi.begin(WiFi.SSID().c_str(), WiFi.psk().c_str());
            ESP.wdtFeed();
            WiFiClientStarted = true;
          } else
          {
            WiFiClientStarted = false;
            WiFi.disconnect();
          }
      } else
      {
        WiFi.disconnect();
      }
  }
  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.
  }
  ESP.wdtFeed();
  return RetValue;
}

void saveCredentials() // Speichere WLAN credentials auf EEPROM
{
  size_t i;
  for (i = 0 ; i < sizeof(MyWiFiConfig) ; i++) // Loeschen der alten Konfiguration
    {
```

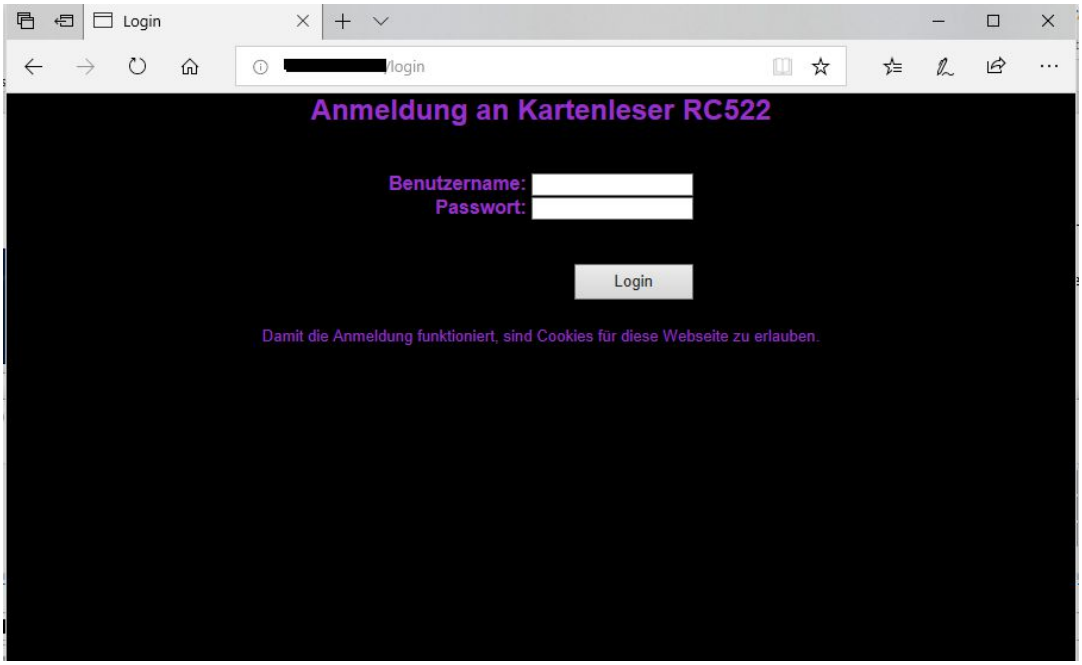
```
      EEPROM.write(i, 0);
    }
  for (i = 0 ; i < STANameLen  ; i++) // Loeschen der alten Konfiguration
    {
     MyWiFiConfig.WiFiPwd[i] = 0;
    }
  for (i = 0 ; i < WiFiPwdLen ; i++) // Loeschen der alten Konfiguration
    {
     MyWiFiConfig.APSTAName[i] = 0;
    }
  temp = WiFi.SSID().c_str();
  i = temp.length();
  temp.toCharArray(MyWiFiConfig.APSTAName,i+1);
  temp = WiFi.psk().c_str();
  i = temp.length();
  temp.toCharArray(MyWiFiConfig.WiFiPwd,i+1);
  temp = "";
  strncpy(MyWiFiConfig.ConfigValid , "TK", sizeof(MyWiFiConfig.ConfigValid) );
  EEPROM.begin(512);
  EEPROM.put(0, MyWiFiConfig);
  EEPROM.commit();
  EEPROM.end();
  ESP.wdtFeed();
}
// ****************** 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);
      SetRGBLed(0,255,0,false);        //Led Grün
      Serial.print("  gültig.");
      delay(2000);
      SetRGBLed(0,0,255,false);        //Led Farbe Blau Leser ist in Grundzustand
   } else
   {
      SetRGBLed(255,0,0,false) ;      //Led Rot - Letzte Karte war ungültig
      Serial.print(" ungültig.");
      delay(2000);
   }
    Serial.println();
    mfrc522.PICC_HaltA();  // Versetzt die gelesene Karte in einen Ruhemodus, um nach anderen Karten suchen zu können.
    delay(100);
```

```
  }
  server.handleClient(); // Webserveranfragen bearbeiten
  yield();               // interne ESP8266 Funktionen aufrufen
  ESP.wdtFeed();         // Watchdog zurückstellen.
  delay(20);
}
```

Wir compilieren den Code und laden ihn auf unseren EDSP hoch. Wir starten den seriellen Monitor und sehen folgende Ausgabe:



In der ersten Zeile wird die Seriennummer des ESP's angezeigt. Diese notieren wir uns, da wir diese für die erste Anmeldung auf der Webseite brauchen. Die zweite Zeile gibt die IP Adresse in unseren LAN an. Diese IP geben wir im Browser ein, und erhalten folgendes Bild:

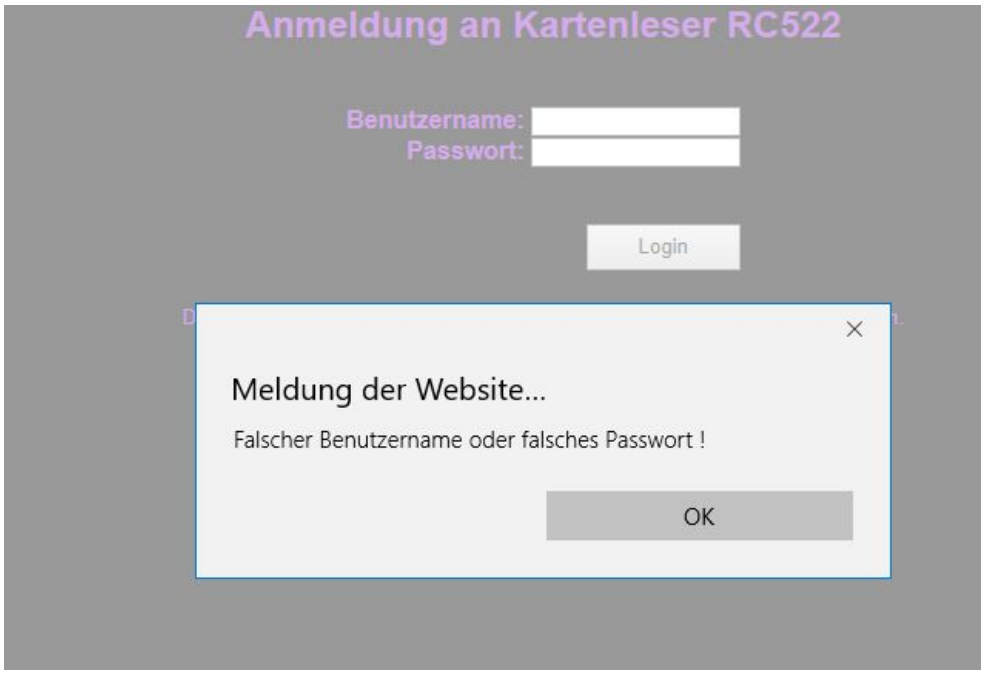


Als Anmeldekennung gilt:#

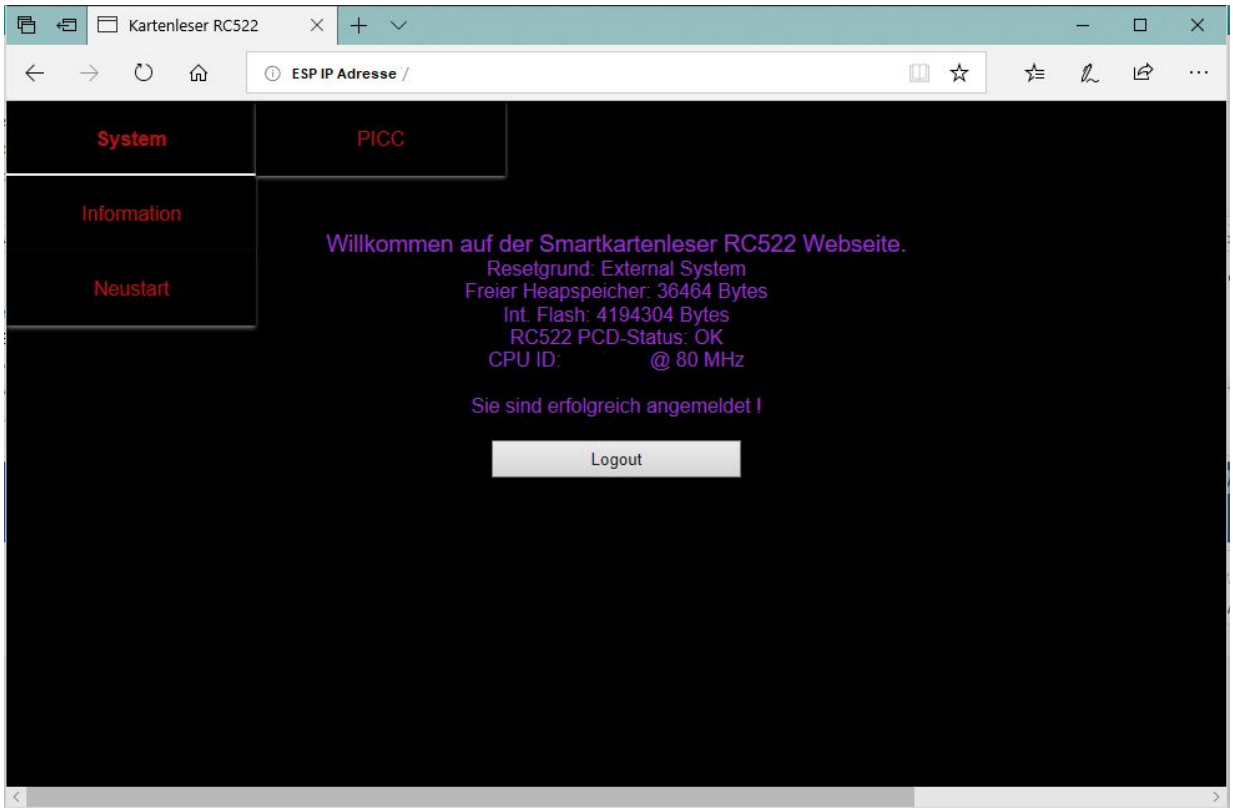
Benutzername: admin

Passwort: Seriennummer des ESP Chips. (ATSN Nummer)

Falls die Anmeldung fehl schlägt sehen wir folgendes Bild:

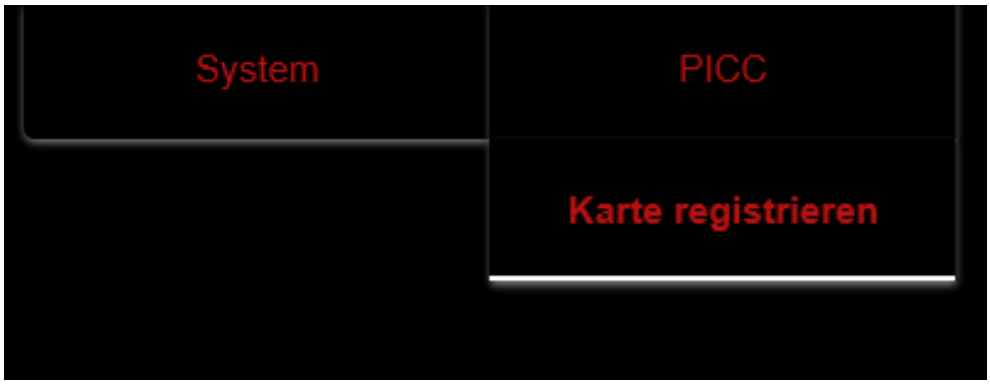


Andernfalls gelangen wir zu dem Hauptmenü:



PICC = Proximity Integrated Circuit Card

Um eine neue Karte zu autorisieren klicken wir nun unter dem Menü „PICC“ auf Karte registrieren:



Im nächsten Artikel in der Reihe heißt es: Down the the Rabbit Hole.. wir beschätigen uns mit den Funktionen der Classic Mifare Karte uns schreiben erstmals Daten auf die Chipkarte.