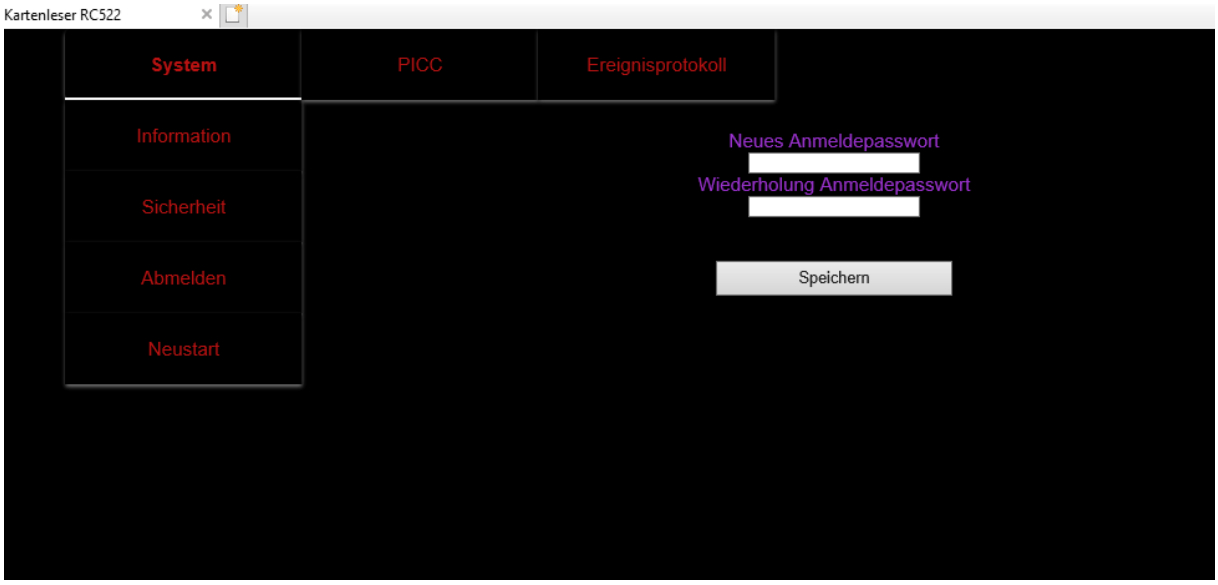
Zugangsbeschränkung zu Geräten per Contactless Card mit der NodeMCU und dem RC522 Modul fünfter Teil – Sicherheit.

Hallo und Willkommen zu einem neuen Blogbeitrag für den MiFare Card Reader.

In dem heutigen und den nächsten Blog Beitrag in der Reihe widmen wir uns den Bereichen Sicherheit und Flexibilität.

Wir wollen die Sicherheit und Flexibilität unseres Kartenlesers etwas beleuchten, sowie für mehr Transparenz bei den Kartenleseraktionen sorgen. Ein großes Manko, das die vorausgehenden Teile dieser Reihe hatten, war, dass sich zwar das Anmeldekennwort des Administrators (admins) auf der Weboberfläche, von Kartenleser zu Karteleser unterschied, dieses Kennwort jedoch nicht individuell änderbar war.
Dies wollen wir heute ändern. Ab sofort ist es möglich, über die Weboberfläche unter System -> Sicherheit ein eigenes Anmeldekennwort für den Benutzer "admin" einzutragen.



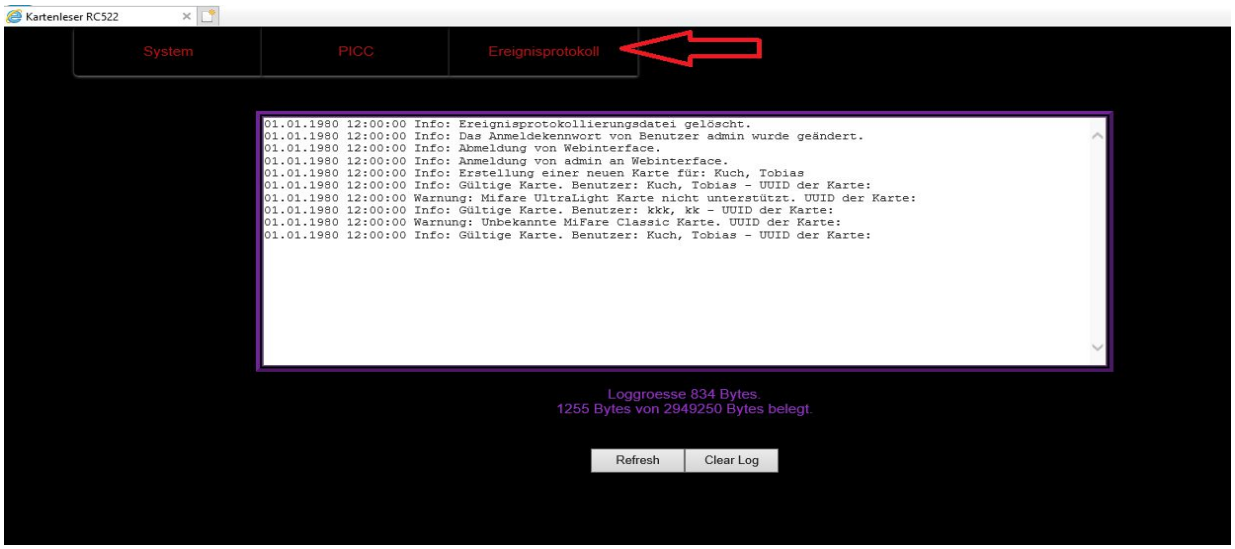
Dieses eigene Kennwort kann zwischen 3 und 16 Zeichen lang sein.
Eingestellt werden kann diese definierte Zeichenspanne mit den Definitionen PwMinLength und PwMaxLength im Code. Wenn kürzere bzw. längere Passwortgrenzen gewünscht werden, können die Werte individuell angepasst werden.

Die Vorgehensweise für eine Passwortänderung des Benutzers „admin“ ist:

- Erstanmeldung an die Weboberfläche mit dem Benutzernamen "admin" und dem Passwort bestehend aus der ESP ChipId. (Wie in den vorherigen Teilen der Reihe bereits beschrieben)
- Navigation unter System -> Sicherheit dann 2x das gleiche (neue) Passwort eingeben und Button ... drücken.
- Logout unter System -> Abmeldung. Das neue Passwort ist sofort gültig.

Somit haben wir die Sicherheit schon mal etwas erhöht, sodass jemand, der unser Projekt kennt und Zugang zur seriellen Schnittstelle und über Netzwerk hat, nicht gleich das gültige Anmeldepasswort herausfinden kann.
Kommen wir nun zu einem weiteren Punkt der Rubrick Sicherheit: Es wäre doch aus Sicherheitssicht wünschenswert zu wissen, wer wann mit welcher Karte unseren Kartenleser erfolgreich oder auch nicht erfolgreich verwendet hat. Dazu schaffen wir jetzt die Möglichkeit. Unter den neuen Menüpunkt "Ereignisprotokoll" sehen wir nun verschiedene Einträge bei bestimmten Aktionen oder Ereignissen rund um unseren Leser.
Beispielsweise sehen wir nun im Ereignisprotokoll alle Versuche mit gültigen und ungültigen Karten unseren Leser zu einer Aktion zu bewegen.

Folgender Screenshot zeigt dies exemplarisch:



Wie im o.g Screenshot zu erkennen ist, ist die Auflistung der Uhrzeit und des Datums noch generisch auf einen fixen Wert gesetzt, da uns noch eine valide Uhrzeitquelle fehlt. Diese Quelle werden wir in einen der nächsten Teile noch hinzufügen.
Wesentlich für das Ereignisprotokoll ist die korrekte Initalisierung und Verwendung des internen ESP SPIFFS Filesystems. Dazu stellen wir die Kompilieroptionen so ein, das von unseren verfügbaren 4 MB an Programmspeicher 2 MB als SPIFFS Dateisystem formatiert werden. Wie dieses geht, habe ich bereits in einem anderen Blogbeitrag „ESP8266 – Alles SPIFFS oder was?“ von mir unter Einrichtung beschrieben.

Wichtig: Ohne die korrete SPIFFS Einstellung wird der Code zwar compliliert und auf den ESP hochgeladen, jedoch beibt beim Booten der ESP mit der Meldung „ATFS:NOK“ auf der seriellen Schnittstelle „hängen“ und die weitere Initalisierung schlägt fehl. *Diese SPIFFS Einstellung müssen wir für alle weiteren folgenden Teile dieser Reihe beibehalten.*

Der Code für unseren heutigen Blog ist:

```
#include <SPI.h>
#include <MFRC522.h>
#include <ESP8266WiFi.h>
#include <ESP8266WebServer.h>
#include <ESP8266mDNS.h>
#include <EEPROM.h>
#include <FS.h>          // Include the SPIFFS library

#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 KEYA true        //PICC Flag Definition
#define KEYB false       //PICC Flag Definition

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

#define PwMinLength 3  // Minimum Login Passwortlänge
#define PwMaxLength 16  // Maximum Login Passwortlänge
#define LoginNameMaxLength 16  // Maximum Login Name Length

#define USED_Sector 1  // Kartensektor, der für die Authentifizierungs - und Konfigurationsdaten evrwendet wird ( gültig: 1- 15)

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
File myfile;            // erstelle ein SPIFFS Handling Variable
FSInfo fs_info;

struct WiFiEEPromData // 68 Bytes Object Lenght
 {
  char ESPHostName[ESPHostNameLen]; // 20
  char APSTAName[STANameLen]; // STATION /AP Point Name TO Connect, if definded   20
```

```
  char WiFiPwd[WiFiPwdLen]; // WiFiPAssword, if definded  25
  char ConfigValid[3]; //If Config is Vaild, Tag "TK" is required" 3
 };

struct MiFareClassicKeyTable
 {
  byte Key_A[6] = {0x22,0x44,0xFA,0xAB,0x90,0x11};   // Schlüssel für PICC Karte bitte ändern.
  byte Key_B[6] = {0xFE,0xE1,0xAA,0x3D,0xDF,0x37};   // Schlüssel für PICC Karte bitte ändern.
  char ConfigValid[3]; //If Config is Vaild, Tag "TK" is required"
 };

struct SysConf //
 {
  char LoginName[LoginNameMaxLength]; // 16
  char LoginPassword[PwMaxLength];  // 16
  bool PCD_Disabled; // 1
  byte USED_PICC_Sector; // 1
  char ConfigValid[3]; //If Config is Vaild, Tag "TK" is required" // 3
 };




MiFareClassicKeyTable MiFareClassicKey;
WiFiEEPromData MyWiFiConfig;
SysConf SystemConfiguration;


// Global genutzte Variablen
bool Result  = false;
bool LearnNewCard = false;
bool EraseCard = false;
bool ExpirationDateActive = false;
bool PCD_Disabled = false;
String Surname;
String Givenname;
String ExpirationDate;
String temp;
unsigned long SessionID;
unsigned long PCD_ServiceCall_Handler = 0;
unsigned long PCD_WatchDog_Handler = 0;
uint8_t DataBuffer[18]  = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 };


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(255,0,255,false);      //Led Farbe Lila Initalisierung beginnen
 Serial.begin(115200);            // Serielle Kommunikation mit dem PC mit 115200 Baud initialisieren
 Serial.println(F(" "));
 temp = "ATSN:"+ String(ESP.getChipId());
 Serial.println(temp);
 // Serial.print("ADC Value:");Serial.println(analogRead(A0));
 SPI.begin();                     // Initialisiere SPI Kommunikation
 PCDHardReset();
 SessionID = millis();
 Result = loadSysConfig();
 Result = InitalizeFileSystem();
 Result = startWiFiClient();
 InitalizeHTTPServer();
 temp = "Systemstart: ";
 temp += String(ESP.getResetReason());
 WriteToLog(temp, 0);
 temp = "";
 ESP.wdtEnable(4000);           // Starte Watchdog
 SetRGBLed(0,0,255,false);      //Led Farbe Blau 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);
    yield();
    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")){
   //Disconnect Benutzers;
   SessionID = millis();
   temp = String(ESP.getChipId()) + "= NA ; HttpOnly ; SameSite=Strict";
   server.sendHeader("Set-Cookie",temp);
   temp = "Abmeldung von Webinterface.";
   WriteToLog(temp,0);
   temp = "";
   PCD_Disabled = false; // PCD aktivieren nach logout ;
   SetRGBLed(0,0,255,false);      //Led Farbe Blau StandardModre
   server.sendHeader("Location","/login");
   server.sendHeader("Cache-Control","no-cache");
   server.send(301);
   server.send(301);
   //yield();
  }
  if (server.hasArg("USERNAME") && server.hasArg("PASSWORD")){
   temp = String(ESP.getChipId());
   if (server.arg("USERNAME") == SystemConfiguration.LoginName &&  server.arg("PASSWORD") == SystemConfiguration.LoginPassword
){
     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 = "Anmeldung von " + server.arg("USERNAME") + " an Webinterface.";
     WriteToLog(temp,0);
     temp = "";
     server.send(301);
     PCD_Disabled = true; // PCD deaktivieren, solange Per Web gearbeitet wird;
     SetRGBLed(255,0,255,false);      //Led Farbe Lila in Servicemode
     return;
    }
  msg = "<script>alert('Falscher Benutzername oder falsches Passwort !');</script>";
  temp = "Fehlerhafter Anmeldeversuch von " + server.arg("USERNAME") + " an Webinterface. Passwort: " +server.arg("PASSWORD") ;
  WriteToLog(temp,1);
  temp = "";
  }
   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";
 }
 yield();
 server.send(404, "text/plain", temp);
 temp = "Aufruf der ungültigen URL: " + server.uri() + " Parameter: " +String(server.args()) ;
 WriteToLog(temp,2);
 temp = "";
 }



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

```
 temp = "<head><title>Kartenleser RC522</title></head><body>";
 server.sendContent(temp);
 HtmlNavStructure();
 temp = "<script>alert('Bitte JETZT die zu löschende Karte vor den Leser halten!');</script>";
 server.sendContent(temp);
 yield();
 EraseCard = true;
 PCD_Disabled = false;
 SetRGBLed(0,255,255,false);      //Led Farbe cyan Programmierungsmodus
 temp = "</body></html>";
 server.sendContent(temp);
 server.client().stop();
 temp = "";
}

void handleSettings()
{
 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 = "<br><br><br><div align=center><form method='post' action='/settings' method='post'>";
 server.sendContent(temp);
 temp = "<p><label>Neues Anmeldepasswort</label><br> <input type='password' name='newPassword'>";
 server.sendContent(temp);
 temp = "</p><p><label>Wiederholung Anmeldepasswort</label><br> <input type='password'
name='confirmNewPassword'><br><br><br>";
 server.sendContent(temp);
 temp = "<button type='submit' name='Save' value='1' style='height: 30px; width: 200px' >Speichern</button></p></form></div>";
 server.sendContent(temp);
 // temp = "</p><p><button type='submit' name='save'>Speichern</button></p></form></div></body></html>";
 if (server.hasArg("Save"))
   {
   if (server.arg("confirmNewPassword") == server.arg("newPassword"))
    {
    temp = server.arg("newPassword");
    if ((temp.length() < PwMinLength ) |  (temp.length() > PwMaxLength ))
     {
     temp = "<br><div align=center><br>Das Kennwort muss zwischen mindestens ";
     server.sendContent(temp);
     temp = String(PwMinLength) + " und maximal " + String(PwMaxLength) + " Zeichen lang sein.</div>";
     server.sendContent(temp);
     } else
     {
     temp = server.arg("newPassword");
```

```
      temp.toCharArray(SystemConfiguration.LoginPassword ,PwMaxLength);
      saveSysConfig();
      temp = "Das Anmeldekennwort von Benutzer " + String(SystemConfiguration.LoginName) + " wurde geändert.";
      WriteToLog(temp,0);
      temp = "<script>alert('Das Anmeldekennwort wurde erfolgreich geändert.');</script>";
      server.sendContent(temp);
      }
     } else
     {
     temp = "<br><div align=center><br>Die eingegebenen Kennwörter stimmen nicht überein. Bitte nochmals eingeben.</div>";
     server.sendContent(temp);
     }
   }
 temp = "</body></html>";
 server.sendContent(temp);
 server.client().stop();
 temp = "";
}

void handleNewPICC()
{
  if (!is_authentified())
    {
    server.sendHeader("Location","/login");
    server.sendHeader("Cache-Control","no-cache");
    server.send(301);
    return;
    }
  if (server.hasArg("Surname") && server.hasArg("Givenname"))
  {
    Surname = server.arg("Surname");
    Givenname = server.arg("Givenname");
    ExpirationDate = server.arg("ExpDate");
    if (server.hasArg("ExpDateOption")) { ExpirationDateActive = true; } else { ExpirationDateActive = false; }
    temp = "Erstellung einer neuen Karte für: " +server.arg("Givenname") +", " + server.arg("Surname");
    WriteToLog(temp,0);
    PCD_Disabled = false;
    LearnNewCard = true;
    CSS_Header_Template();
    temp = "<head><title>Kartenleser RC522</title></head><body>";
    server.sendContent(temp);
    HtmlNavStructure();
    temp = "<script>alert('Bitte JETZT die neue Karte vor den Leser halten!');</script>";
    server.sendContent(temp);
    temp = "</body></html>";
    server.sendContent(temp);
    server.client().stop();
    SetRGBLed(255,255,0,false);      //Led Farbe Gelb Programmierungsmodus
    yield();
    return;
  }
  CSS_Header_Template();
```

```
  temp = "<head><title>Kartenleser RC522</title></head><body>";
  server.sendContent(temp);
  HtmlNavStructure();
  temp = "";
  temp = "<br><br><br><br><table border=0 ALIGN=CENTER><th>";
  server.sendContent(temp);
  temp = "<table border=1 bgcolor = black><form action='/newPICC' method='post'>";
  server.sendContent(temp);
  temp = "<tr><th>Karteninhaber:<br><div ALIGN=RIGHT>";
  server.sendContent(temp);
  temp = "Vorname: <input type=text Name='Surname' Size=17 maxlenght=16 placeholder='Max' required><br>";
  server.sendContent(temp);
  temp = "Nachname: <input type=text Name='Givenname' Size=17 maxlenght=16 placeholder='Mustermann' required><br>";
  server.sendContent(temp);
  temp = "</div></th><th>Kartenmetadaten:<br><DIV ALIGN=RIGHT>";
  server.sendContent(temp);
  temp = "<input Name='ExpDateOption' TYPE=checkbox VALUE=1 >Ablaufdatum:<input type=date Name='ExpDate' Size = 17 >";
  server.sendContent(temp);
  temp = "<br><th><tr><th></table><br>";
  server.sendContent(temp);
  temp = "<button type='submit' name='NewCard' value='1' style='height: 30px; width: 200px' >Neue Smartcard erstellen</button>";
  server.sendContent(temp);
  temp = "<br></form></tr></th></table>";
  server.sendContent(temp);
  temp = "</body></html>";
  server.sendContent(temp);
  server.client().stop();
  temp = "";
}



void handleLog()
{
  int FileSize =  0;
  if (!is_authentified())
    {
    server.sendHeader("Location","/login");
    server.sendHeader("Cache-Control","no-cache");
    server.send(301);
    return;
    }
  if (server.hasArg("ClearLog"))
  {
  if (SPIFFS.exists ("/usage_log.csv") )  //Prüfe ob Datei usage_log.csv exisiert.
    {
      SPIFFS.remove("/usage_log.csv");  //Lösche Datei
      Serial.println(F("ATFS:LogDelete"));
      temp = "Ereignisprotokollierungsdatei gelöscht.";
      WriteToLog(temp,0);
      temp = "";
    } else { Serial.println(F("ATFS:LogDelete_ERR")); }
    yield();
```

```
 }
CSS_Header_Template();
yield();
temp = "<head><title>Kartenleser RC522</title></head><body>";
server.sendContent(temp);
HtmlNavStructure();
temp = "<br><br><br><br><table border=4 ALIGN=CENTER><th>";
server.sendContent(temp);
temp = "<textarea rows='20' cols='110' warp=soft readonly>";
server.sendContent(temp);
if (SPIFFS.exists ("/usage_log.csv"))  //Prüfe ob Datei usage_log.csv exisiert.
 {
  myfile = SPIFFS.open("/usage_log.csv", "r");  //Öffne die Datei usage_log.csv im Root Verzeichnis zum lesen
  if (!myfile)
   {
   temp = "Interner Fehler: Logdatei usage_log.csv konnte nicht geöffnet werden !";
   server.sendContent(temp);
   } else
   {
   FileSize = myfile.size();
   while (myfile .position()<myfile.size())
    {
      temp=myfile.readStringUntil('\n');
      temp.trim();
      temp += "\n"; // Add Character Line Feed
      server.sendContent(temp);
    }
   myfile.close();
   }
 } else
 {
  temp = "Interner Fehler: Logdatei usage_log.csv nicht gefunden !";
  server.sendContent(temp);
 }
temp = "</textarea></th></table><table  border=0 ALIGN=CENTER>";
server.sendContent(temp);
SPIFFS.info(fs_info);
temp = "<br><div ALIGN=CENTER> Loggroesse "+ String(FileSize)+ " Bytes.<br>";
server.sendContent(temp);
temp = String(fs_info.usedBytes) + " Bytes von " + String(fs_info.totalBytes) + " Bytes belegt.</div>";
server.sendContent(temp);
temp = "<th><form action='/log' method='post'><br><br>";
server.sendContent(temp);
temp = "<button type='submit' name='RefreshLog' value='1' style='height: 30px; width: 100px' >Refresh</button>";
server.sendContent(temp);
temp = "<button type='submit' name='ClearLog' value='1' style='height: 30px; width: 100px' >Clear Log</button>";
server.sendContent(temp);
temp = "</form></th></table></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();
 yield();
 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_SetAntennaGain(mfrc522.RxGain_max); // Setzt Antenne auf max. Empfang
 mfrc522.PCD_AntennaOn();
 yield();
 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>";
 server.sendContent(temp);
 temp = "<br>Programmiert von <a href='mailto:tobias.kuch@googlemail.com'>Tobias Kuch</a><br></div></body></html>";
 server.sendContent(temp);

 if (server.hasArg("Reboot") )  // Reboot System
  {
  temp = "<script>alert('Das System startet JETZT neu.');</script>";
  server.sendContent(temp);
  server.client().stop();
  yield();
  temp = "";
  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 = "";
  server.send (200, "text/html", temp);
  temp = "<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.01 Transitional//EN'><html lang='de'><meta charset='UTF-8'>";
  server.sendContent(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='/settings'>Sicherheit</a></li>";
 server.sendContent(temp);
 temp = "<li><a href='/login?DISCONNECT=1'>Abmelden</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'>Neue Karte erstellen</a></li>";
```

```
  server.sendContent(temp);
  temp = "<li><a href='/erasePICC'>Karte löschen</a></li></ul>";
  server.sendContent(temp);
  temp = "</li>";
  temp = "</li><li><a href='/log'>Ereignisprotokoll</a></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.on("/erasePICC", handleErasePICC);
  server.on("/settings",handleSettings);
  server.on("/log", handleLog);
  server.onNotFound ( handleNotFound );
  server.collectHeaders(headerkeys, headerkeyssize );// Server anweisen, diese zu Tracken
  server.begin(); // Web server start
  }

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

// ******************* Start Functions Filesystem ******************************************
boolean InitalizeFileSystem()
{
  bool initok = false;
  initok = SPIFFS.begin();
  delay(200); // Without delay I've seen the IP address blank
  if (!(initok)) // Format SPIFS, of not formatted. - Try 1
  {
    Serial.println(F("ATFS:FRM"));
    SPIFFS.format();
    initok = SPIFFS.begin();
  }
  if (!(initok)) // Format SPIFS, of not formatted. - Try 2
  {
    SPIFFS.format();
    initok = SPIFFS.begin();
  }
  if (initok) { Serial.println(F("ATFS:OK")); } else { Serial.println(F("ATFS:NOK")); }
  return initok;
}

boolean WriteToLog(String LogEntry, byte Logtype)
{
```

```
bool writeok = false;
SPIFFS.info(fs_info);
if (fs_info.totalBytes - fs_info.usedBytes - 2000 > 1) // Speicherplatz überprüfen
 {
 myfile = SPIFFS.open("/usage_log.csv", "a");  //Öffne die Datei usage_log.csv im Root Verzeichnis zum schreiben
 if (!myfile)
  {
  myfile.close();
  Serial.println(F("ATFS:InitLog_ERR"));
  return writeok;
  }
 temp = "";
 temp +="01.01.1980 12:00:00 "; // noch EchtDatum hinzufügen
 switch (Logtype)
  {
  case 0:
   temp +="Info: ";
  break;
  case 1:
   temp +="Warnung: ";
  break;
  case 2:
   temp +="Fehler: ";
  break;
  default:
   temp +="Info: ";
  break;
  }
 temp +=LogEntry;
 writeok = myfile.println(temp);
 if (!(writeok)) { Serial.println(F("ATFS:WriteLog_ERR")); }
 myfile.close();
 temp = "";
 }
return writeok;
}


// ****************** End Functions Filesystem ***********************************************

// ****************** 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(F("ATWPS:OK"));
```

```
      saveCredentials(); // Save Credentials to EEPROM
    } else {
     Serial.println(F("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(F(".")); // Connect vorgang auf der seriellen Schnittstelle beobachten
    //ESP.wdtFeed();
    delay(500);
    yield();
    connRes  = WiFi.waitForConnectResult();
    }
   if (connRes == 4 ) { // if password is incorrect
    Serial.println(F("ATWIFI:PWDERR"));
    WiFi.disconnect();
    }
   if (connRes == 6 ) { //  module is not configured in station mode
    Serial.println(F("ATWIFI:STAERR"));
    WiFi.disconnect();
    }
  }
 if(WiFi.status() == WL_CONNECTED)
  {
  //ESP.wdtFeed();
  Serial.print(F("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(F("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();
    }
 }
 //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.
 }
 //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();
}

void saveSysConfig()
{
  byte i;
  strncpy(SystemConfiguration.ConfigValid , "TK", sizeof(SystemConfiguration.ConfigValid) );
  EEPROM.begin(512);
  EEPROM.put(80, SystemConfiguration);
  EEPROM.commit();
  EEPROM.end();
  //ESP.wdtFeed();

}

bool loadSysConfig()
{
bool RetValue;
 EEPROM.begin(512);
 EEPROM.get(80,SystemConfiguration);
 EEPROM.end();
 if (String(SystemConfiguration.ConfigValid) == "TK")
  {
    RetValue = true;
  } else
  {
    temp = String(ESP.getChipId());
    temp.toCharArray(SystemConfiguration.LoginPassword ,PwMaxLength);
    temp = "";
```

```
   strncpy(SystemConfiguration.LoginName ,"admin", sizeof(SystemConfiguration.LoginName) );
   SystemConfiguration.PCD_Disabled = false;
   SystemConfiguration.USED_PICC_Sector = 1;
   saveSysConfig();
   RetValue = false; // Config Settings nicht gültig //Standardinitalisierung ausführe
  }
  //ESP.wdtFeed();
  return RetValue;
}

// ****************** END Functions StoreCredentialsto EEPROM ***************************

// ****************** Start Functions CardServices  ************************************

void PCDHardReset()
{
  digitalWrite(RST_PIN,LOW);
  delay(200);
  digitalWrite(RST_PIN,HIGH);
  mfrc522.PCD_Reset();
  mfrc522.PCD_Init();                       // Initialisiere MFRC522 Lesemodul
  mfrc522.PCD_SetAntennaGain(mfrc522.RxGain_max); // Setzt Antenne auf max. Empfang
  mfrc522.PCD_AntennaOn();
}

boolean CardAuthenticate(boolean ABKey, byte Sector,byte ikey[6])
{
const byte sectorkeytable [16] = {3,7,11,15,19,23,27,31,35,39,43,47,51,55,59,63};
byte statusA;
statusA = 0;
for (int a = 0; a < 6;a++)
  {
  key.keyByte[a] = ikey[a];
  }
  // Key A
if (ABKey)
  {
  statusA = mfrc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_A, sectorkeytable[Sector], &key, &(mfrc522.uid));
  if (statusA != MFRC522::STATUS_OK)
    {
    Serial.println(F("ATAUTH:ERR_A"));
    return false;
    }
  }
  // Key B
  else if (not ABKey)
  {
  statusA = mfrc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_B,sectorkeytable[Sector], &key, &(mfrc522.uid));
  if (statusA != MFRC522::STATUS_OK)
    {
    Serial.println(F("ATAUTH:ERR_B"));
    return false;
```

```
   }
  }
return true;
}

// WriteData . uses Global Variable DataBuffer for Data Return
boolean CardDataWrite(byte Sector,byte block,byte value[16])
{
byte status;
byte writevector;
byte sectorkeytable [16] = {3,7,11,15,19,23,27,31,35,39,43,47,51,55,59,63};
writevector = Sector * 4 + block -1 ;
for (byte a = 0; a < 16; a++)
  {
  if (writevector ==  sectorkeytable[a])
    {
    // Serial.println("NAK");
    return false;
    }
  }
status = mfrc522.MIFARE_Write(writevector, value, 16);
if (status != MFRC522::STATUS_OK)
  {
  Serial.println(F("ATPCD:W_ERR"));
  //Serial.println(mfrc522.GetStatusCodeName(status));
  return false;
  } else
  {
  // Serial.print(F("ATPCD:Write_Card_OK"));
  return true;
  }
}

// Read Data - uses Global Variable DataBuffer for Data Return
boolean CardDataRead(byte Sector,byte block)
{
byte statusi;
byte readvector;
const byte sectorkeytable [16] = {3,7,11,15,19,23,27,31,35,39,43,47,51,55,59,63};
byte sized = 18;
readvector = Sector * 4 + block -1 ;
for (byte a = 0; a < 16; a++)
  {
  if (readvector ==  sectorkeytable[a])
    {
    Serial.println(F("ATPCD:R_ERR"));
    return false;
    }
  }
statusi = mfrc522.MIFARE_Read(readvector, DataBuffer, &sized);
if (statusi != MFRC522::STATUS_OK)
  {
```

```
 Serial.println(F("ATPCD:R_ERR"));
 return false;
 } else
 {
 return true;
 }
}

boolean ResetCardToDefault()
{
byte ikey[16];
byte status,i;
byte writevector;
const byte sectorkeytable [16] = {3,7,11,15,19,23,27,31,35,39,43,47,51,55,59,63};
writevector = sectorkeytable[USED_Sector];
if (CardAuthenticate(KEYB,USED_Sector,MiFareClassicKey.Key_B)) // Sector Autenticate for WRITE Access
   {
    for (i = 0; i <= 16; i++) { DataBuffer[i] = 0; }
    if (!(CardDataWrite(USED_Sector,1,DataBuffer))) { return false; }
    for (i = 0; i <= 16; i++) { DataBuffer[i] = 0; }
    if (!(CardDataWrite(USED_Sector,2,DataBuffer))) { return false; }
    for (i = 0; i <= 16; i++) { DataBuffer[i] = 0; }
    if (!(CardDataWrite(USED_Sector,3,DataBuffer))) { return false;}
   }
for (byte i = 0; i <= 16; i++) { ikey[i] = 255; }  //Load Default Key for all Sectors
ikey[6] = 0xFF; // Default Setting for Access Bits
ikey[7] = 0x07; //
ikey[8] = 0x80; //
ikey[9] = 0x69;
status = mfrc522.MIFARE_Write(writevector, ikey, 16);
if (status != MFRC522::STATUS_OK)
 {
 return false;
 }
return true;
}

boolean SetSectorAccessControl (byte Sector,byte Akey[6],byte Bkey[6])
{
byte ikey[16];
byte status;
byte writevector;
const byte sectorkeytable [16] = {3,7,11,15,19,23,27,31,35,39,43,47,51,55,59,63};
writevector = sectorkeytable[Sector];
ikey[0] = Akey[0];
ikey[1] = Akey[1];
ikey[2] = Akey[2];
ikey[3] = Akey[3];
ikey[4] = Akey[4];
ikey[5] = Akey[5];
ikey[6] = 0x78; // Data Block 0-3 Access Conditions: Key B write / Key A Read
ikey[7] = 0x77; // KEY A & KEY B & Acces Bits Write:Key B  / Key A Read Access Bits
```

```
ikey[8] = 0x88; // Calculator: http://calc.gmss.ru/Mifare1k/
ikey[9] = 0x69; // Fixer Wert - > default hex 69
ikey[10] = Bkey[0];
ikey[11] = Bkey[1];
ikey[12] = Bkey[2];
ikey[13] = Bkey[3];
ikey[14] = Bkey[4];
ikey[15] = Bkey[5];
status = mfrc522.MIFARE_Write(writevector, ikey, 16);
if (status != MFRC522::STATUS_OK)
 {
 Serial.println(F("ATPCD:W_KEY_ERR"));
 return false;
 }else
 {
 return true;
 }
}

boolean CheckforDefaultCardKey ()
{
byte tkey[6];
boolean CardResult;
byte readvector;
byte statusi;
const byte sectorkeytable [16] = {3,7,11,15,19,23,27,31,35,39,43,47,51,55,59,63};
byte sized = 18;
for (byte i = 0; i <= 6; i++) { tkey[i] = 255; }  //Load Default Key for all Sectors
CardResult = true;
if (!CardAuthenticate(KEYA,USED_Sector,tkey)) { CardResult = false; };
readvector = sectorkeytable[USED_Sector];
statusi = mfrc522.MIFARE_Read(readvector, DataBuffer, &sized);
if (statusi != MFRC522::STATUS_OK) { CardResult = false; }
//if (!((DataBuffer[7] = 0x07) & (DataBuffer[7] = 0x80)))  { CardResult = false; };
return CardResult;
}

boolean WriteNewMiFareClassicPICC ()
 {
 byte tkey[6];
 byte i,a;
 boolean CardResult;
 if (CheckforDefaultCardKey())
  {
  for (i = 0; i <= 6; i++) { tkey[i] = 255; }  //Load Default Key for all Sectors
  for (i = 0; i <= 16; i++) { DataBuffer[i] = 0; } // Clear Variable Buffer
  CardResult = true;
  if (CardAuthenticate(KEYA,USED_Sector,tkey)) // Sector  Autenticate
   {
   // Serial.println("Auth Sec 0 OK");
   if (Surname.length() > 15) { a = 15; } else { a = Surname.length();}
   if (Surname.length() > 0)
```

```
      {
      for (i = 0; i <= 16; i++) { DataBuffer[i] = 0; }
      for (i = 0; i <= a; i++) { DataBuffer[i] = Surname[i]; }
      if (!(CardDataWrite(USED_Sector,2,DataBuffer))) { CardResult = false; } //Sector 0 Block 2 Vorname mit Key A schreiben
      }
    if (Givenname.length() > 15) { a = 15; } else { a = Givenname.length(); }
    if (Givenname.length() > 0)
      {
      for (i = 0; i <= 16; i++) { DataBuffer[i] = 0; }
      for (i = 0; i <= a; i++) { DataBuffer[i] = Givenname[i]; }
      if (!(CardDataWrite(USED_Sector,3,DataBuffer))) { CardResult = false; } //Sector 0 Block 3 Nachname mit Key A schreiben
      }
    if (!(SetSectorAccessControl (USED_Sector,MiFareClassicKey.Key_A,MiFareClassicKey.Key_B))) { CardResult = false; }   // (byte
Sector,byte Akey[6],byte Bkey[6])
    } else {
         CardResult = false;
         return CardResult;
         }
    } else {
         CardResult = false;
         return CardResult;
         }

   if (CardResult)
    {
    //Serial.println("PICC written");
    CardResult = true;
    }
   else
    {
    //Serial.println("PICC not empty");
    CardResult = false;
    }
   yield();
   return CardResult;
 }


boolean ReadMiFareClassicPICC ()
{
boolean CardResult;
byte i ;
CardResult = true;
if (CardAuthenticate(KEYA,USED_Sector,MiFareClassicKey.Key_A)) // Sector  Autenticate with READ Key A
 {
 Givenname = "              "; //PlaceHolder
 Surname   = "              "; //PlaceHolder
 for (i = 0; i < 18; i++) { DataBuffer[i] = 0; } // Clear Variable Buffer
 if (CardDataRead(USED_Sector,2)) // Feld Vorname auslesen
  {
  for (i = 0; i < 16; i++) { Surname[i] = char(DataBuffer[i]);  }
  } else {
```

```
        return false;
        }
   for (i = 0; i < 18; i++) { DataBuffer[i] = 0; } // Clear Variable Buffer
   if (CardDataRead(USED_Sector,3)) // Feld Nachname auslesen
    {
    for (i = 0; i < 16; i++) { Givenname[i] = char(DataBuffer[i]); }
    } else {
         return false;
         }
   } else
   {
   return false;
   }
return true;
}

void CardServer()
{
byte i ;
#define PCD_Poll_Interval 400
#define PCD_Watchdog_Interval 60000
if (millis() - PCD_ServiceCall_Handler >= PCD_Poll_Interval)
 {
  PCD_ServiceCall_Handler = millis();
   if (mfrc522.PICC_IsNewCardPresent())   // PICC = proximity integrated circuit card = kontaktlose Chipkarte
   {
    mfrc522.PICC_ReadCardSerial();
    yield();
   // Unterscheidung nach Kartentyp
   // 0x08 für MIFARE Classic 1K
   // 0x18 für MIFARE Classic 4K
   // 0x11 für MIFARE PLUS
   if (mfrc522.uid.sak == 0x08 || mfrc522.uid.sak == 0x18)
    {
    //  MiFare_Classic_Processor START (mfrc522.uid.sak);  // Nur ausführen wenn Eine Mifare Classic Karte vor den Leser gehalten
wurde.
    byte tkey[6];
    for (byte i = 0; i <= 6; i++) { tkey[i] = 255; }  //Load Default Key for all Sectors
    if(LearnNewCard) // neue Karte soll angelernt werden.
     {
      if (WriteNewMiFareClassicPICC()) {
                                SetRGBLed(0,255,0,false);
                                LearnNewCard = false;
                               } else
                               {
                                SetRGBLed(255,0,0,false);
                                LearnNewCard = false;
                               }
     }

    else if (EraseCard)  // KartenDaten sollen gelöscht werden.
     {
```

```
    if (ResetCardToDefault()) {
                        SetRGBLed(0,255,0,false);
                        EraseCard = false;
                       } else
                       {
                        SetRGBLed(255,0,0,false);
                        EraseCard = false;
                       }
  }
 else
  {
  if (ReadMiFareClassicPICC())
    { // Karte gültig !
    Serial.print (F("ATAUTH_S:"));
    Serial.println (Surname);
    Serial.print (F("ATAUTH_G:"));
    Serial.println (Givenname);
    temp = "Gültige Karte. Benutzer: ";
    for (i = 0; i < 16; i++)
      {
       if  (Givenname[i] == 0)
        {
          break;
        }
      temp += char(Givenname[i]);
      }
    temp += ", ";
    for (i = 0; i < 16; i++)
      {
       if  (Surname[i] == 0)
        {
          break;
        }
      temp += char(Surname[i]);
      }
    temp += " - UUID der Karte: ";
    for (byte i = 0; i < 4; i++) { temp += String(mfrc522.uid.uidByte[i],HEX) + " ";}
    WriteToLog(temp, 0);
    temp = " ";
    //Do actions if correct Card is detected

    bool PinState= digitalRead(RELAIS_PIN);
    PinState = !PinState;
    digitalWrite(RELAIS_PIN, PinState);
    SetRGBLed(0,255,0,false);       //Led Grün
    temp = "";
    } else {
          SetRGBLed(255,0,0,false);
          temp = "";
          for (byte i = 0; i < 4; i++) { temp += String(mfrc522.uid.uidByte[i],HEX) + " ";}
          temp = "Unbekannte MiFare Classic Karte. UUID der Karte: " + temp ;
          WriteToLog(temp, 1);
```

```
            temp = "";
           }
       }
    LearnNewCard = false;
    //  MiFare_Classic_Processor STOP (mfrc522.uid.sak);
    } else if (mfrc522.uid.sak == 0x00) // Mifare Ultralight
    {
     SetRGBLed(255,0,0,false);
     temp = "";
     for (byte i = 0; i < 7; i++) { temp += String(mfrc522.uid.uidByte[i],HEX) + " ";}
     temp = "Mifare UltraLight Karte nicht unterstützt. UUID der Karte: " + temp;
     WriteToLog(temp, 1);
     temp = "";
    } else
    {
     SetRGBLed(255,0,0,false);
     temp = "";
     for (byte i = 0; i < mfrc522.uid.size; i++) { temp += String(mfrc522.uid.uidByte[i],HEX) + " ";}
     temp = "Unbekannte Karte und Kartentyp. Typnummer: " + String(mfrc522.uid.sak) + " UUID der Karte: " + temp ;
     WriteToLog(temp, 1);
     temp = "";
    }
   mfrc522.PCD_StopCrypto1();
   mfrc522.PICC_HaltA();
   delay(2000);
   SetRGBLed(0,0,255,false);       //Led Farbe Blau Leser ist in Grundzustand
   }
  }

if (millis() - PCD_WatchDog_Handler >= PCD_Watchdog_Interval)
  {
   PCD_WatchDog_Handler = millis();
   Result = mfrc522.PCD_PerformSelfTest();
   yield();
   mfrc522.PCD_Init();                    // Initialisiere MFRC522 Lesemodul erneut
   mfrc522.PCD_SetAntennaGain(mfrc522.RxGain_max); // Setzt Antenne auf max. Empfang
   mfrc522.PCD_AntennaOn();
   yield();
   if (!(Result))
    {
     PCDHardReset();
     Serial.println(F("ATPCD:ERR_H"));
    }
  }
}

// ******************* Stop Functions CardServices  ************************************

void loop()  // Hauptschleife
{
  server.handleClient();                 // Webserveranfragen bearbeiten
  if (!(PCD_Disabled) | !(SystemConfiguration.PCD_Disabled)) { CardServer(); }    // Kartenleser spezifische Anfragen bearbeiten
```

```
  ESP.wdtFeed();                  // Watchdog zurückstellen. Deaktivieren mit "wdt_disable();"
}
```

Ich wünsche viel Spaß und viel Erfolg beim Nachbau des Projektes und bis zum nächsten Mal.