TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Využití obvodu ESP32 pro virtuální průmyslovou sběrnici

Bakalářská práce

Studijní program: B2612 Elektrotechnika a informatika
Studijní obor: Elektronické informační a řídicí systémy

Autor práce: **Filip Hess**
Vedoucí práce: Ing. Miroslav Holada, Ph.D.
Ústav informačních technologií a elektroniky

Liberec 2020

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Zadání bakalářské práce

Využití obvodu ESP32 pro virtuální průmyslovou sběrnici

Jméno a příjmení: **Filip Hess**
Osobní číslo: M15000092
Studijní program: B2612 Elektrotechnika a informatika
Studijní obor: Elektronické informační a řídicí systémy
Zadávající katedra: Ústav informačních technologií a elektroniky
Akademický rok: **2019/2020**

Zásady pro vypracování:

1. Seznamte se s obvodem ESP32, zaměřte se na dostupné vývojové kity. Prostudujte možnosti obvodu se zaměřením na webový server, UART a poloduplexní sběrnici RS485.
2. Navrhněte využití obvodu ESP32 pro virtualizaci UART a RS485 přes wifi rozhraní. Pro konfiguraci a nastavení přenosu navrhněte webové rozhraní.
3. Naprogramujte softwarové vybavení čipu ESP32 a realizujte prototyp včetně návrhu plošného spoje.
4. Vytvořený prototyp otestujte a diskutujte možnosti jeho použití.

Rozsah grafických prací:	dle potřeby dokumentace
Rozsah pracovní zprávy:	cca 30-40 stran
Forma zpracování práce:	tištěná/elektronická
Jazyk práce:	Čeština

Seznam odborné literatury:

[1] NOVÁK, Petr. Mobilní roboty: pohony, senzory, řízení. 1. vyd. Praha: BEN – technická literatura, 2005. ISBN 80-7300-141-1.

[2] AXELSON, Jan. *Serial port complete: programming and circuits for RS-232 and RS-485 links and networks*. Madison, WI: Lakeview Research, c1998. ISBN 0965081923.

[3] www.esp32.com

Vedoucí práce:	Ing. Miroslav Holada, Ph.D. Ústav informačních technologií a elektroniky
Datum zadání práce:	9. října 2019
Předpokládaný termín odevzdání:	18. května 2020

L.S.

prof. Ing. Zdeněk Plíva, Ph.D.
děkan

prof. Ing. Ondřej Novák, CSc.
vedoucí ústavu

V Liberci dne 18. října 2019

Prohlášení

Byl jsem seznámen s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu Technické univerzity v Liberci.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti Technickou univerzitu v Liberci; v tomto případě má Technická univerzita v Liberci právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Bakalářskou práci jsem vypracoval samostatně jako původní dílo s použitím uvedené literatury a na základě konzultací s vedoucím mé bakalářské práce a konzultantem.

Současně čestně prohlašuji, že texty tištěné verze práce a elektronické verze práce vložené do IS/STAG se shodují.

3. ledna 2020 Filip Hess

Poděkování

Tímto bych poděkoval Ing. Miroslavu Holadovi, Ph.D,, za vedení mé práce, poskytnuté materiály, rady a nadstandartní konzultační hodiny. Veškerý můj ostatní vděk patří panu Ing. Leoši Petržílkovi za zhotovení prototypových desek plošných spojů.

Abstrakt

Tato práce se zabývala volbou bezdrátového modulu, jeho programováním a návrhu prototypu pro virtualizaci průmyslové sběrnice. Cílem práce tedy bylo vytvořit bezdrátový přenos dat mezi dvěma moduly. Data byla posílána do modulů po sběrnicích.

Řešení práce spočívalo v naprogramování webového serveru a jeho stránek, kde je možnost nastavení sítí a jejich připojení. Jsou zde také dodatečné informace o navázaném spojení. Samotný přenos dat byl pomocí TCP spojení dvou modulů (jeden modul se připojil do sítě druhého). Schéma a návrh desky prototypu byly zhotoveny v softwaru Eagle od firmy Autodesk. Pro programování bylo využito Arduino a knihovny pro vybraný modul.

Posílání dat, testováno v sériových terminálech, přes rozhraní UART proběhlo úspěšně. U posílání dat z UART přes WiFi do RS485 byl problém v přepínání režimu čtení a zápisu., protože je RS485 poloduplexní. Data, která byla posílána do rozhraní příliš rychle, dorazila neúplná. Tuto příčinu se při testování podařilo odstranit blokovaným přístupem.

Přínosem této práce bylo ukázat využití tohoto obvodu, jeho programování webového serveru a ostatních použitých funkcí, včetně práce s WiFi, TCP a pamětí. Výsledkem práce byla zhotovená aplikace pro virtualizaci sběrnice na vytvořeném a testovaném prototypu.

Klíčová slova: ESP32, webový server, UART, WiFi, Eagle, virtualizace

Abstract

This thesis dealt with the choice of wireless module, its programming and design of prototype for bus virtualization. Purpose of that work was create wireless data transmitter between two modules. Data was sent to module by bus.

Solution of the work was to program web server and its html pages for ability to set networks and connections. There are also additional infromation about connection. Data transmitting was made by connecting two module via TCP protocol (one module connected to network of the other module). Scheme and PCB of the prototype were designed in Eagle software from Autodesk. Arduino IDE was used for programming and uploading sketch to the module.

Data transmitting via UART was successful. Otherwise there was a problem with sending data from UART via WiFi to RS485. Main reason was caused by switching mode for reading or transmitting, because RS485 is halfduplex. Data was sending via UART too quickly and because of that data on RS485 was incomplete.That error was fixed during testing by blocking the access in module.

The benefit of this work was to show usage of the circuit, programming web server and other features including work with WiFi, TCP and memory. Result of the work was fully builded aplication for bus virtualization in developed prototype.

Keywords: ESP32, webserver, UART, WiFi, Eagle, virtualization

Obsah

Seznam použitých obrázků 9
Seznam zdrojových kódů 9
Seznam zkratek 10
1. Úvod 11
2. WiFi modul ESP32 12
2.1 Srovnání s předchůdcem ESP8266 13
2.2 Verze modulů 13
2.3 Dostupné vývojové kity 14
3. Vizualizace a programování ESP32 16
3.1 Možnosti programování 16
3.2 Nastavení WiFi módů a jejich implementace 17
3.3 Vytvoření WebServeru a jeho obsah 19
3.4 Nastavení rozhraní UART 21
3.5 Přenos dat pro RS485 21
4. Prototyp s modulem ESP32 23
4.1 Schéma napájecího zdroje 23
4.2 Schéma modulu a rozhraní 24
4.3 Prototyp DPS 25
5. Implementace a popis webového rozhraní 26
6. Závěr 29
Použitá literatura 30
Přílohy 31
A. Blokové schéma 31
B. Celé schéma se zdrojem 32
C. DPS s popisky 33
D. Webové rozhraní na PC (rozlišení 1920x1080) 34
E. Vývojový diagram spojení 35
F. Obsah přiloženého CD 36

Seznam použitých obrázků

Obrázek 1: Blokové schéma čipu ESP32[10] 12
Obrázek 2: Předchozí model ESP8266[11] 13
Obrázek 3: Vybraný modul ESP-WROOM-32D 14
Obrázek 4: Vývojový kit ESP32 DevKitC V4[12] 14
Obrázek 5: Vývojový kit ESP-WROVER v4.1[13] 15
Obrázek 6: Schéma vizualizace prvků 16
Obrázek 7: Znázornění spojení dvou ESP32 18
Obrázek 8: Zobrazení přenosu dat na UART pomocí osciloskopu 22
Obrázek 9: Schéma napájecího zdroje prototypu 23
Obrázek 10: Schéma modulu ESP32 24
Obrázek 11: Návrh desky plošného spoje prototypu 25
Obrázek 12: Hlavní stránka v MASTER režimu 26
Obrázek 13: Stránka nastavení a uart 27
Obrázek 14: Stránka režimu SLAVE a výběr sítě 28
Obrázek 15: Hlavní stránka po připojení 28

Seznam zdrojových kódů

Zdrojový kód 1: Připojení k existující síti 17
Zdrojový kód 2: Otevírání souboru s JSON 19
Zdrojový kód 3:Vyvolaná metoda přechodem "/" 19
Zdrojový kód 4: Nastavení stránek s vyvolanou funkcí 20
Zdrojový kód 5: Funkce pro scanování dostupných sítí 21
Zdrojový kód 6: Definování UART2 21

Seznam zkratek

AP	Access Point
BLE	Bluetooth Low Energy
CAN	Controller Area Network
CMOS	Complementary Metal Oxide Semiconductor
DE	Driver Output Enable
DEF	Default
DHCP	Dynamic Host Configuration Protocol
GPIO	General Purpose Input Output
HTML	HyperText Markup Language
JSON	Javascript Object Notation
JTAG	Joint Test Action Group
LDO	Low Dropout Outpu
P2P	Peer to Peer
PCB	Printed Circuit Board
RE	Reciever Output Enable
ROM	Read Only Memory
SPI	Seriál Peripheral Interface
SPIFFS	SPI Flash File System
SRAM	Static Random Access Memory
SSID	Service Set Identifier
TCP	Transmission Control Protocol
UART	Universal Asynchronous Reciever-Transmitter
ULP	Ultra Low Power
WPA	WiFi Protected Access

1. Úvod

Účelem práce je vytvořit bezdrátový most pro UART/RS485, kde figurují dvě zařízení s čipem ESP32, které se spolu spojí přes WiFi. Data přivedená na rozhraní UART na jednotlivých ESP32 jsou posílána protokolem TCP na rozhraní druhého ESP. Zde jsou data nadále posílána přes drátovou sběrnici.

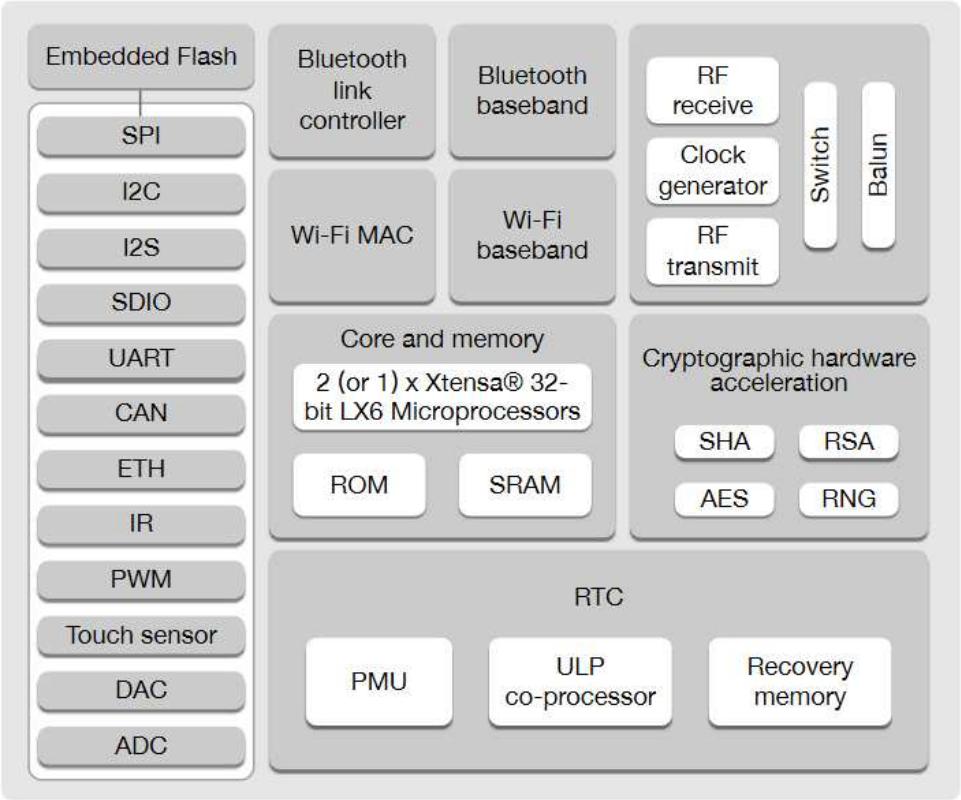
Na trhu existují podobné moduly od firmy Wi2Wi, které se v základních vlastnostech a ceně v podstatě neliší. Společnost Microchip má v produkci chipy s označením ATWIN, také v příjemné výkonné a cenové dostupnosti. Texas Instruments se svým modelem CC3220MOD sází na zvýšený výkon antény a velký počet uživateli přístupných pinů. Cena za jeden kus je však desetinásobná. Pro domácí a nekomerční projekty jsou právě nejrozšířenější produkty od firmy Espressif. Jak starší model ESP8266 či nejnovější moduly osazeny čipem ESP32.

Na trhu je k dispozici zařízení DTU-H100, které již umožňuje přenos ze sériové linky či sběrnice RS232 nebo RS485. Data mohou být posílána jak přes WiFi, tak kabelovým připojením Ethernet. Zařízení je již složitější, má přímo zabudované konektory a externí odnímatelnou anténu. Cena tohoto převodníku dosahuje několika násobek zhotoveného a testovaného prototypu této práce.

Cílem práce je seznámit se s vývojovými kity, moduly ESP32 a jejich verzemi. Zvolit optimální typ pro vytvoření vlastního prototypu s napájením a ostatníma periferiema jako např. převodník na RS485, signalizace. Dále vyrobený prototyp naprogramovat a otestovat funkčnost webového rozhraní a přenosu dat.

2. WiFi modul ESP32

Jedná se o kombinovaný čip (viz Obrázek 1) s přidanými periferiemi designovaný a vyráběný firmou Espressif. Je postaven na 32-bitovém 2jádrovém (či 1jádrovém) mikroprocesoru Xtensa LX6 s operační frekvencí až 240MHz, pamětí ROM 448KB a pamětí SRAM 520KB. K samotnému čipu jsou v modulu zabudované periferie. Například I^2C, UART, A/D i /DA převodníky, CAN, SPI a další.



Obrázek 1: Blokové schéma čipu ESP32[10]

Většina modulů[2] má již zabudovanou anténu pro 2,4GHz s rychlostí přenosu pro standard 802.11n až 150Mbit/s. Pro některé moduly jako je např. ESP32-WROOM-32U je možnost připojení antény externí přes standardní U FL konektor. Modul je také osazen i o další možnost bezdrátového přenosu, a to je Bluetooth na verzi 4.2 s vlastností BLE.

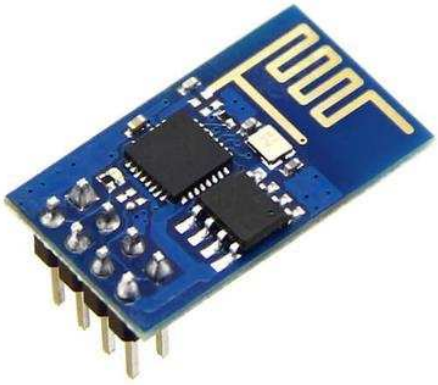
Modul musí být napájen napětím 3,3V a jelikož se jedná o CMOS procesor, je jeho spotřeba nízká. Výrobce uvádí, že v režimu Modern-Sleep při použití WiFi může být odebíraný proud až 19mA. Při využití režimu Deep-Sleep v závislosti na použitých periferiích a koprocesoru ULP lze spotřebu stáhnout až na 10µA. V normálním aktivním

režimu se může spotřeba v závislosti na využití WiFi či Bluetooth zvýšit na více než 200mA.

2.1 Srovnání s předchůdcem ESP8266

Před rokem 2016 byl tzv. vlajkovou lodí firmy Espressif jejich předchozí modul označován jako ESP8266[8] (viz Obrázek 2), který nenabízel tolik možností jako modul stávající. Mezi hlavním rozdílem byl zajisté výkon, kdy starší model běžel na procesoru pouze s jedním jádrem, proto nebylo výhodné modul používat pro mnoho náročných operací, včetně fungování WiFi sítě se zobrazováním stránek pro vícero klientů.

S novým modelem došlo také o rozšíření o Bluetooth, navýšení počtu sériového rozhraní UART, přidání komunikační sběrnice CAN a téměř dvojnásobný počet GPIO.



Obrázek 2: Předchozí model ESP8266[11]

2.2 Verze modulů

Z úvodního popisu (viz kapitola 2) lze verze jednoduše rozdělit na 1 jádrové (značení S) a 2 jádrové (značení D). Přičemž ještě záleží, jestli se jedná o verzi modulu U/I/IPEX, které jako jediné nemají integrovanou anténu.

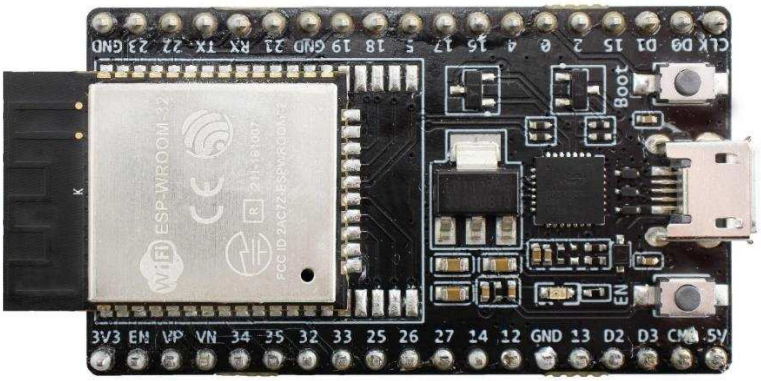
První základní verze modulů, označována jako WROOM a druhá WROVER. Hlavním rozdílem je to, že WROVER má navíc PSRAM paměť o velikosti 8MB. Paměť je také dělena na dvě verze podle operačního napájení a frekvence (1,8V až 144MHz, 3,3V až 133MHz).

Pro zhotovení práce byl vybrán modul ESP32-WROOM-32D[4] (viz Obrázek 3) pro svou dostupnost a dostatečnost k dosažení a splnění zadání. K programování tohoto modulu byl použit programátor ESP-PROG[3] od stejného výrobce.

Obrázek 3: Vybraný modul ESP-WROOM-32D

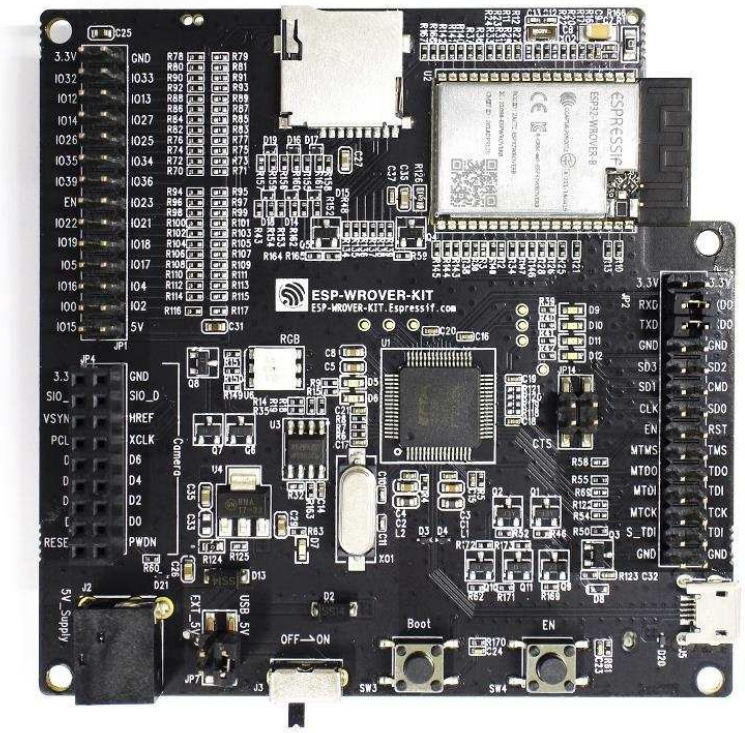
2.3 Dostupné vývojové kity

Pro univerzálnější a jednodušší použití těchto modulů jsou vytvořeny tzv. kity. Příkladem je aktuální verze ESP32 DevKitC V4 (viz Obrázek 4). Je to menší vývojová deska, osazena modulem WROOM. Nechybí zde integrace programového rozhraní s výstupem na microUSB, které zajišťuje i napájení celé vývojové desky. Snížení napětí z USB na 3,3V zajišťuje jednoduchý LDO stabilizátor. Na desce jsou rovněž vyvedena tlačítka pro bootování a resetování. Jednotlivé GPIO piny jsou vyvedeny pro snadnější připojení.



Obrázek 4: Vývojový kit ESP32 DevKitC V4[12]

Mezi větší variantu patří ESP-WROVER-KIT v4.1(viz Obrázek 5). Tato deska je osazena modulem VROVER, navíc může být napájena externě 5V a může být programována přes JTAG rozhraní. Velká pozornost byla také věnována zadní straně, zde byl osazen 3,2” LCD displej. Pro větší uložiště jde zde zakomponován i slot na MicroSD karty.

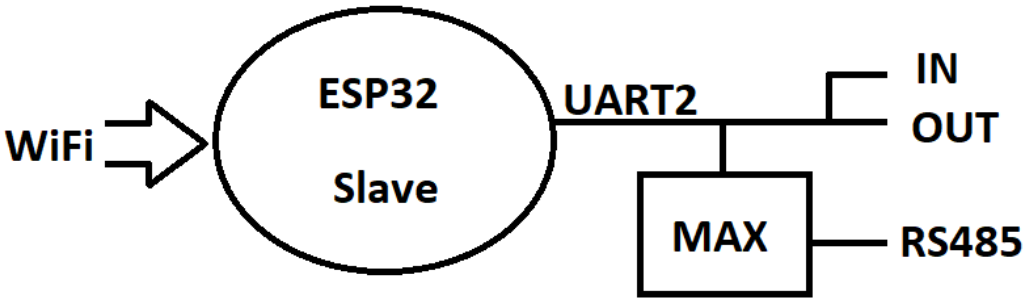


Obrázek 5: Vývojový kit ESP-WROVER v4.1[13]

Tyto vývojové kity nebyly posouzeny a vybrány jako vhodné ke splnění některých částí zadání. Na těchto přípravcích je možné připravit a otestovat softwarové funkce a skripty, které mohou být nadále užity ve vyrobeném vlastním prototypu. Hlavní příčinou výběru samostatného modulu bylo to, že je možné zhotovit vlastní externí napájení a přidání na PCB převodník na RS485 s možností vyvedením sériového rozhraní UART.

3. Vizualizace a programování ESP32

Pro jednoduchou představu jak je celý koncept a jak jsou všechny prvky a preiferie zahrnuty do výsledného přenosu dat ukazuje (Obrázek 6). Vyobrazení obou modulů je znázorněno v příloze A.



Obrázek 6: Schéma vizualizace prvků

Podrobněji popsáno se jedná o bezdrátový přenos dat. Data jsou do ESP32 vedena přes UART napřímo nebo přes převodník MAX485 ze sběrnice RS485. Data jsou uložená do vnitřního bufferu sériového rozhraní. Ty jsou nadále převzata a poslána přes TCP packet do druhého ESP32, kde je opačný tok dat jako při vstupu do prvního bezdrátového modulu. Moduly ESP jsou značeny jako Master a Slave.. První má vytvořenou vlastní AP síť, do které se připojuje uživatel, aby mohl měnit nastavení. Slave má taktéž vlastní AP, ale má možnost se ještě připojit k již existující síti. V tomto případě k AP síti druhého ESP v režimu Master. Po udělení IP adresy od DHCP serveru se vytvoří TCP spojení na obou stranách, tzn. TCP server i TCP klient. Oba dva prototypy jsou tedy navzájem propojení P2P sítí.

3.1 Možnosti programování

Z důvodu vybraného modulu, je použit programátor ESP-PROG (viz kapitola 2.2), který převádí bootovací rozhraní UART1 na USB. Nevyužitou možností zůstává programování a ladění přes rozhraní JTAG, který umožňuje jak flashování tak debugování. Je potřeba však provést manuální konfiguraci, nastavení a použití více pinů.

Zvolené programování přes Arduino[6] zato nabízí hned několik výhod. Obsáhlé knihovny pro jednotlivé funkce a k nim určené a přehledné metody, proměnné a popisy s příkladem použití. Nevýhodou však je to, že se nezaobírá příliš do hloubky. Všechno je definované obecně a defaultně. Proto je potřeba podrobněji knihovnu znát a vážit všech nenucených úprav v ní.

3.2 Nastavení WiFi módů a jejich implementace

Pro ovládání a manipulaci se využívá standardizovaná knihovna „WiFi.h“. Vytvoření lokální sítě je potvrzeno příkazem WiFi.softAP(ssid,password), kde ssid je podmíněno min. třemi znaky. Položka password je nepovinná, ale pro můj účel je voleno osmimístné heslo pro zabezpečovací protokol WPA2. Před samostatným provedením tohoto příkazu je určena konfigurace lokální IP adresy, defaultní brány a masky. Zvolený mód pro WiFi u obou prototypů s ESP32 je AP/STA. Toto nastavení umožňuje zároveň provozovat vlastní AP síť a zároveň se k již existující síti připojit.

Příkaz připojení k síti naopak není definován v metodě setup(), která se provede při každém restartu modulu, ale v metodě ConnectToNET(), bez návratové hodnoty.

```
if(WiFi.status() == WL_CONNECTED) {WiFi.disconnect();}
  WiFi.begin(ssidNET,passNET);
  int t = 0;
  while (WiFi.status() != WL_CONNECTED && (t < 10)) { //
Wait for the Wi-Fi to connect
    delay(500);
    Serial.print('.');
    t++;
  }
```

Zdrojový kód 1: Připojení k existující síti

Na začátku této metody (viz Zdrojový kód 1) je podmínka, když je již vytvořené síťové spojení, tak se nejdříve ukončí, než se bude volat WiFi.begin(ssidNET, passNET) a vytvoří se spojení nové. Poté se ve smyčce, po dobu stanovenou zdržením delay() v počtu deseti taktů, čeká na potvrzení o připojení kontrolou statusu.

Obrázek 7: Znázornění spojení dvou ESP32

Vstupní proměnné funkce WiFi.begin() jsou definovány jako globální a jejich hodnotu určuje vyplnění formuláře na webovém rozhraní. SSID či název sítě uživatel nevyplňuje, po nastavení prototypu na chování módu SLAVE taktéž ve webovém rozhraní. Jsou všechny dostupné WiFi sítě detekovány a jejich názvy vypsány do selectboxu ve formuláři o nastavení nového připojení. Vstupní pole pro heslo má skryté znaky a jeho hodnota je po odeslání formuláře uložena do globální proměnné stejně jako název sítě.

Pro změnu aktuální AP sítě je taktéž zapotřebí metody s globálními proměnnými, jelikož se tato metoda volá kdykoliv, kdy je nějaká operace se sítí a okolnosti to tak vyžadují např. změna módů MASTER/SLAVE, připojení k jiné síti, změna lokální sítě. V metodě CreateAp() se znova nastavuje lokální síť. Avšak do globálních proměnných se načítá název a heslo sítě, které je uloženo v paměti. Toto je z důvodu, aby se po změně nastavení sítě a následném restartu uchovalo poslední nastavení sítě. To že byla zaznamenána změna sítě, se také ukládá do paměti, aby se při dalším restartu vytvořila uložená síť poslední a ne defaultní. Průběh spojení je znázorněn na vývojovém diagramu v příloze E.

```
File file = SPIFFS.open(filename);
  StaticJsonDocument<512> doc;
  DeserializationError error = deserializeJson(doc, file);
  if (error)
    Serial.println(F("Failed to read file, using default
configuration"));
```

Zdrojový kód 2: Otevírání souboru s JSON

Pomocí knihoven ArduinoJson a SPIFFS lze přistupovat k paměti a ukládat či číst soubory ve formátu JSON. Manipulace s daty v paměti se provádí najednou. Po formátování flash paměti se vytvoří textový konfigurační soubor, ve kterém jsou předepsané položky, kam se budou zapisovat hodnoty. Pro utvořenou strukturu se ukládají a načítají názvy a hesla pro AP, NET a DEF síť. Kde AP je zmiňovaná lokální síť, NET síť druhého modulu. Pro síť DEF jsou definované defaultní hodnoty pro načtení po resetu. V ukázce (Zdrojový kód 2) je příklad otevírání souboru.

3.3 Vytvoření WebServeru a jeho obsah

Pro vytvoření webového serveru je zapotřebí mít obsaženou knihovnu „Webserver.h“. Nyní je možné deklarovat server na portu 80, což je HTTP server. Pro zobrazení stránek, správnou funkci a odezvu je nutné definovat vlastnosti veškerých použitých stránek, formulářů a skriptů na začátku v setupu (Zdrojový kód 4). První tři příkazy odkazují na metody, ostatní také ale se zvoleným prostředním parametrem pro získání dat z formuláře. Poslední tři příkazy fungují s odkazem na vypisování chybných hlášek na HTML stránku přes skript.

V uvedeném příkladu funkce je znázornění poslání dat k zobrazení uživateli. Globální proměnná main_page funguje jako index stránky a její obsah je po celém programu modifikován, aby zobrazoval aktuální žádaná data. Proměnná není však v hlavním programovém souboru, ale pro přehlednost je umístěna do souboru „index.h“.

```
void handleRoot()
{
    String m = main_page;
    server.send(200, "text/html", m);
}
```

Zdrojový kód 3:Vyvolaná metoda přechodem "/"

Funkce handleSet a handleUart jsou stejné jako funkce předchozí s rozdílem, že proměnná představuje jiný řetězec či stránku k zobrazení.

Zato handleArg a další pracují s návratovými hodnotami z formulářů z několika stránek. Pomocí příkazu *server.arg(„název“)* je možné ukládat hodnoty z formulářových inputů do proměnných a rovnou je, pokud je to za potřebí, formátovat či přetypovávat. U této specifické funkce se právě kontroluje, zdali jsou nově zadaný název a heslo lokální AP sítě korektně zadaný, aby se předešlo chybám. Jelikož jsou určitá pravidla pro vytvoření WiFi sítě (viz kapitola 3.2).

```
server.on("/", handleRoot);
server.on("/set", handleSet);
server.on("/uart", handleUart);
server.on("/arg", HTTP_POST, handleArg);
server.on("/uset", HTTP_POST, handleUset);
server.on("/mode", HTTP_POST, handleMode);
server.on("/connect", HTTP_POST, handleConnect);
server.on("/disconnect", HTTP_POST, handleDisconnect);
server.on("/Err", handleErr);
server.on("/Err2", handleErr2);
server.on("/Err3", handleErr3);
server.onNotFound(handle_NotFound);
```

Zdrojový kód 4: Nastavení stránek s vyvolanou funkcí

Při změně módu v handleMode na režim SLAVE se volá v podmínce funkce ScanWiFi()(Zdrojový kód 5), jejíž návratová proměnná je datového typu boolean, který má jen dva stavy. Uvnitř se volá metoda scanNetworks(), jejíž návratovou hodnotou je počet viditelných sítí. Pokud je počet roven nule, jako návratová hodnota se volí false v příkaze return. Je-li však nalezena nějaká síť, tak se jednoduchým cyklem typu for zapíší do definovaného pole. Tyto data jsou nadále zdrojem pro selectbox, který byl již zmíněn v kapitole 3.2.

```
bool ScanWifi()
{
  int n = WiFi.scanNetworks();
  Wifi_count = n;
  if (n == 0)
  {
    Serial.println("no networks found");
    return false;
  }
  else
  {
    Wifi_array[n];
    Serial.println(n);
    for(int i = 0; i < n; i++)
    {
      Serial.println(WiFi.SSID(i).c_str());
      Wifi_array[i] = WiFi.SSID(i).c_str();
    }
    return true;
  }
}
```

Zdrojový kód 5: Funkce pro scanování dostupných sítí

3.4 Nastavení rozhraní UART

Sériové rozhraní, které je tvořeno dvěma vodiči Rx pro čtení a Tx pro zapisování je využíváno pro bootování/flashování programu do paměti. Přesněji k tomu slouží označený UART1.

UART2 je tedy volný a přístupný pro využití dle libosti uživatele. V Arduinu se k němu přistupuje jako Serial2, avšak pro přehlednost a vlastní konfiguraci je použita možnost definování vlastního pojmenování pomocí knihovny „rom/rtc.h“. Tímto definováním a funkcí begin je port otevřen. V parametrech je definovaná přenosová rychlost v baudrate, počet bitů a parita. Poslední dva parametry udávají pin Rx a Tx.

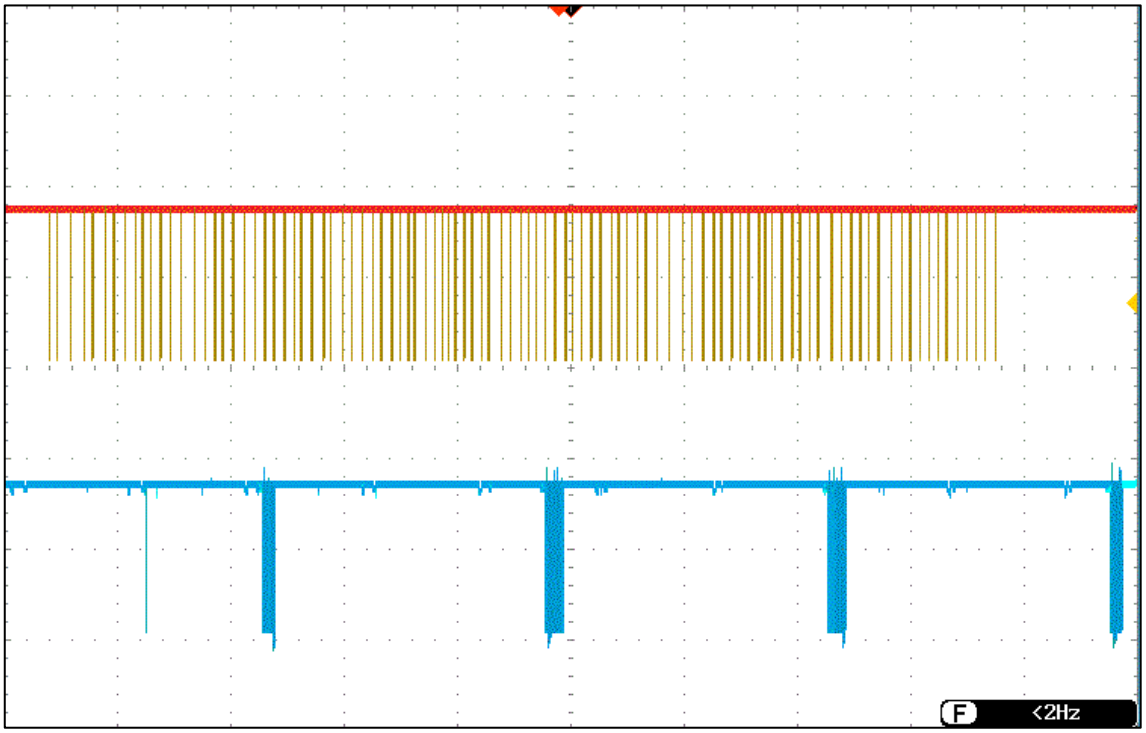
```
HardwareSerial UART2(2);
UART2.begin(UART2_BAUD,UART2_PARAM,UART2_RX, UART2_TX);
```

Zdrojový kód 6: Definování UART2

3.5 Přenos dat pro RS485

Pro správné nastavení komunikace po sběrnici je potřeba určit logickou úroveň na vstupních pinech RE a DE na převodníku. V tomto případě jsou piny propojené, jelikož reagují na opačnou logickou úroveň. Pro logickou nulu je aktivní vstup RE, ten zajišťuje,

že převodník poslouchá a čte data na sběrnici. Při logické jedničce a aktivním vstupu DE se právě zapisuje na sběrnici. Jelikož je to sériová komunikace vedená přes rozhraní UART, stačí akorát správně přepínat tyto dva režimy převodníku v závislosti na přijímaných packetech přes TCP spojení a datech na UART2. Na tomto obrázku níže je vidět, jak probíhá přenos bajtů na sériové lince v závislosti na čase. Takže lze určit, jaké má bezdrátový přenos dat zpoždění.



Obrázek 8: Zobrazení přenosu dat na UART pomocí osciloskopu

Na vloženém obrázku (viz Obrázek 8) je žlutě znázorněné větší množství dat (100B). Znaky jsou posílány se zpožděním 5ms. Na modře vyznačených datech je vidět packetizace. Data jsou přes TCP protokol plněna do bufferu a jakmile je buffer plný, odešle se packet s daty do druhého modulu. Vyobrazená data jsou měřena osciloskopem na sériovém rozhraní.

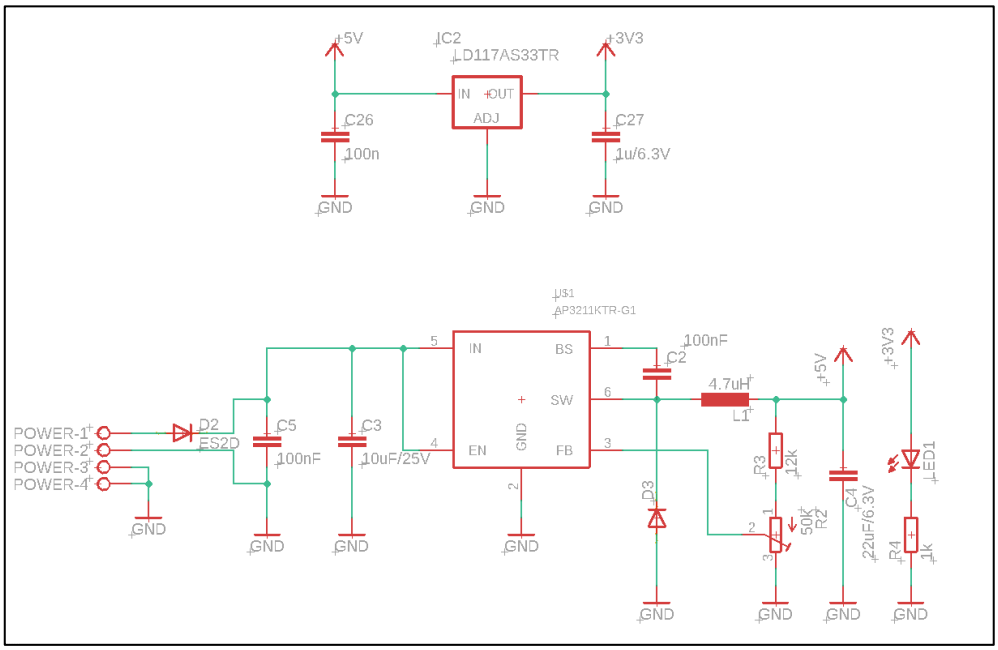
Při testování komunikace ve smyčce a na reálném zařízení připojeném k RS 485 se při provedení 2316 přenosů 3% provedlo nesprávně. Příchozí nebo odchozí data byla neúplná. Tuto chybu zapříčinila komunikace na sběrnici, kdy master čekal na odpověď slavu a zároveň se již dotazoval na posílání dalších dat. Toto vedlo k useknutí dat o pár Bytů. Pro nápravu této závady byl utvořen algoritmus pro blokovaný přístup na sběrnici.

4. Prototyp s modulem ESP32

Pro účel a také cílem této práce je zhotovit jednostrannou desku plošného spoje z utvořeného a navrženého schématu. Je žádoucí, aby byl na stejné desce navržen i stabilizátor pro napájení potřebné k modulu či ostatních zařízení. Pro jednotlivé návrhy a schémata byl použit software Eagle, spadající pod Autodesk.

4.1 Schéma napájecího zdroje

Podle datasheetu a informací z kapitoly 2 je daný základ, co je od napájecího zdroje zapotřebí. K napájecímu napětí 3,3V a popsané spotřebě při zatížení modulu byl jako zdroj zvolen AP3211[5]. Jedná se o dostupný DC-DC konvertor neboli měnič stejnosměrného napětí. Jeho maximální proudové zatížení je dle dokumentace 1.5A. Výstupní napětí se volí odporovým děličem na výstupu zdroje. Vstupní napětí je v rozsahu 4,5 až 18V.Je tedy možno celou desku napájet akumulátorem nebo klasickými monočlánky. Tento zdroj má taktéž vysokou účinnost, dosahující 92%.



Obrázek 9: Schéma napájecího zdroje prototypu

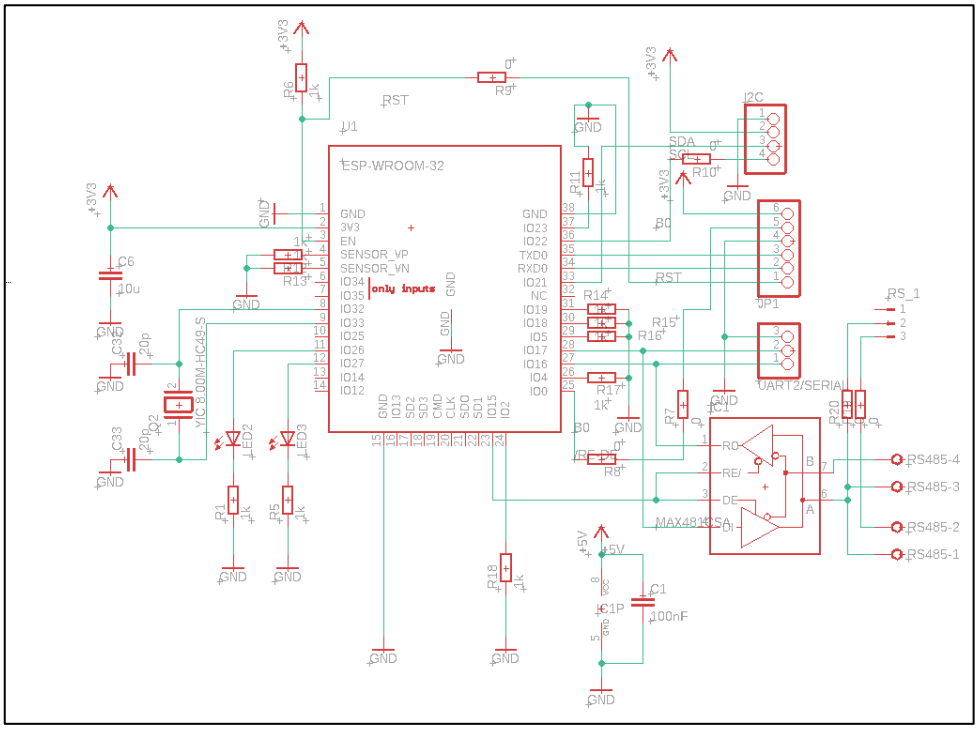
Pro testování je zde přidán i LDO stabilizátor na regulaci výstupního napětí. Zdroj je taky opatřen diodou, která funguje jako ochrana proti přepólování a následnému zkratu, což by zapříčinilo trvalé poškození modulu a součástek.

4.2 Schéma modulu a rozhraní

Uprostřed schématu (viz Obrázek 10) je symbol modulu se všemi vstupními a výstupními piny. Napětí je přivedeno přímo ze stabilizovaného zdroje na určené napětí. Piny 26 a 27 jsou nastaveny jako piny výstupní na světelnou signalizaci pomocí LED diod. K pozdějšímu návrhu byl přidán i oscilační krystal. Větší schéma je v příloze B.

Pro další možnosti modulu bylo vyvedeno rozhraní sběrnice I^2C, pro možnost připojení externích periferií, včetně paměti. Pro nahrávání programu slouží vyvedený hřeben označený JP1. Pro bootování pomocí ESP-PROG, je pro tyto účely zapotřebí mít přivedený EN a IO0 pin. Taktéž samozřejmě rozhraní UART1, který slouží výhradně pro nahrávání programu. Ve schématu reprezentován piny TXD0 a RXD0. Pro možnost absenci extérního napájení je v konektoru vedeno napájení přímo z programátoru. Musí se však dbát na pozici jumperu přepínajícího napětí mezi 3,3V a 5V.

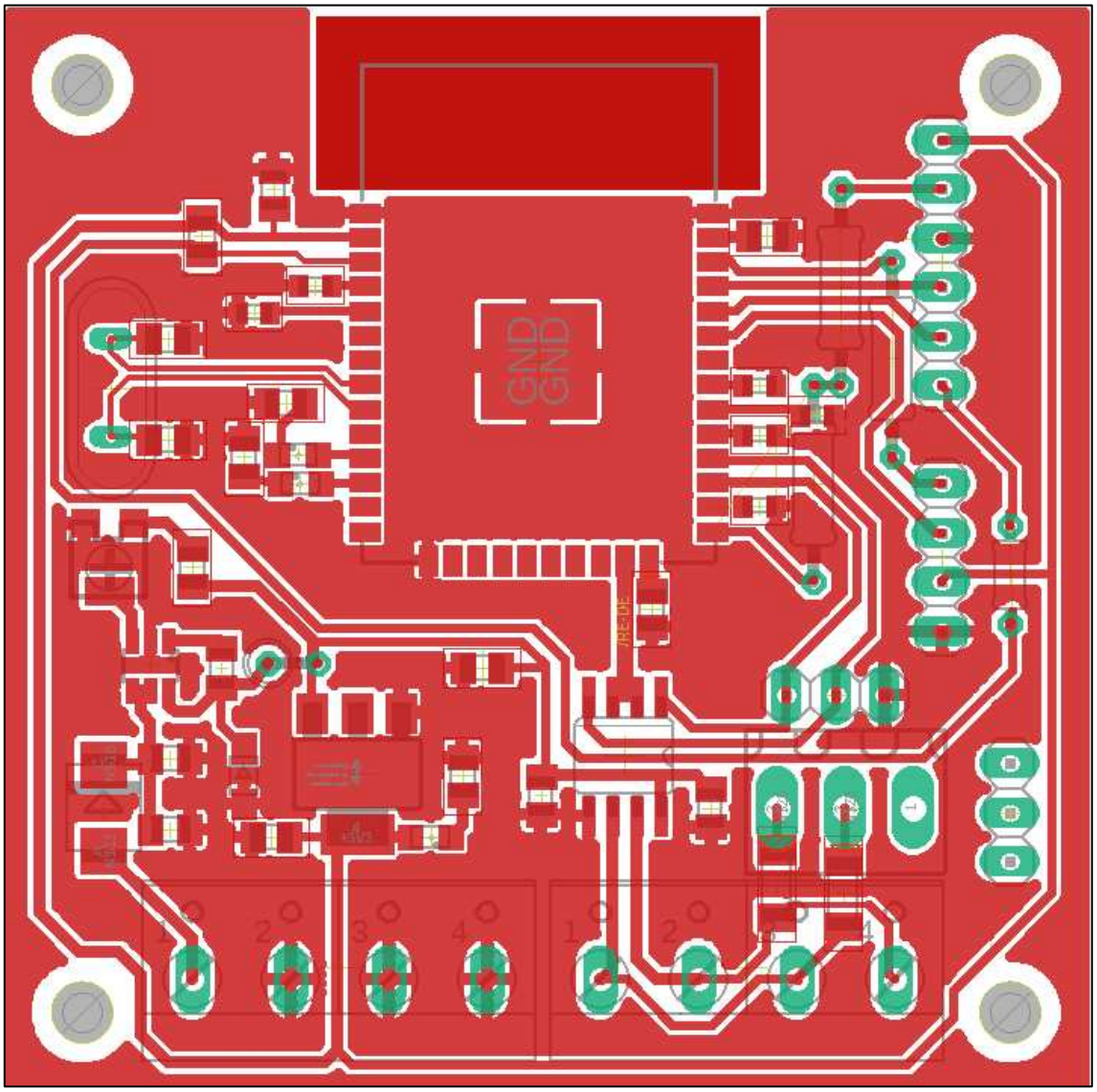
Pro uskutečnění posílání dat je sériové rozhraní UART2 jednoduše vyveden na 3 pinový hřeben. K tomu je paralelně připojen převodník na průmyslovou sběrnici RS485, jehož ovládací piny jsou spojené do jednoho. Pin je napojen přímo k modulu. K propojení sběrnice s převodníkem jsou použity standardní šroubovací svorky.



Obrázek 10: Schéma modulu ESP32

4.3 Prototyp DPS

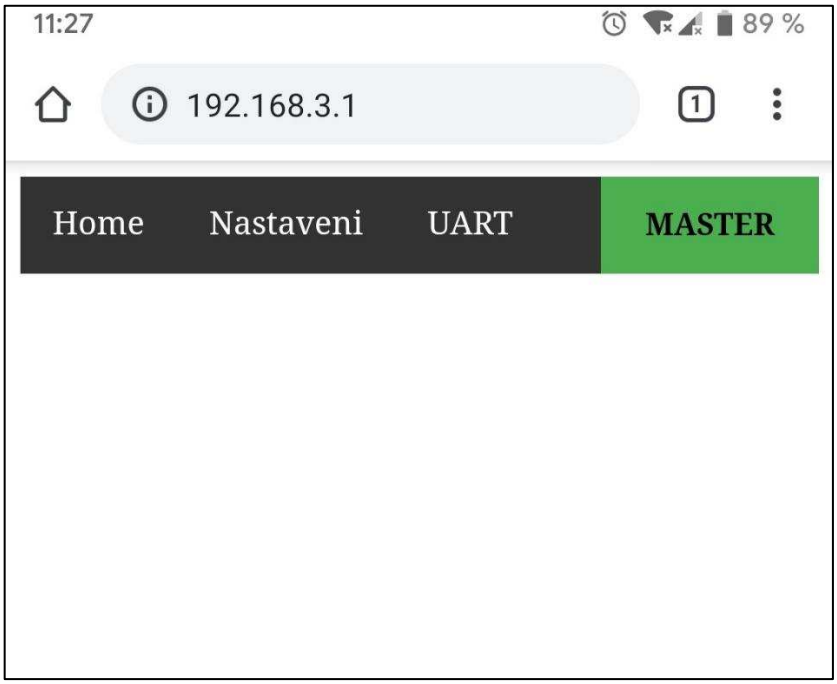
Rozměry prototypní desky byly určeny na 58mm na výšku a 58mm i na šířku. V rozích jsou přidány montážní otvory. Rozložení modulu, součástek, vstupní a výstupních konektorů bylo voleno souměrně. Velký červený obdélník ve vrchní části návrhu reprezentuje anténu. Některé piny, zdající se nezapojené, jsou spojené dohromady přes rozlitou měď. Jedná se o vyplnění prostoru mezi vedenými signály, značenými červeně. Všechny tyto propojené piny reprezentují GND zemnící signál. Deska se zobrazenými popisy je přiložena v příloze C.



Obrázek 11: Návrh desky plošného spoje prototypu

5. Implementace a popis webového rozhraní

Do osazeného a oživeného (zprovozněného) prototypu byl nahrán celý program společně s připojeným souborem obsahující definice vzhledu HTML stránek. Po prvním nahrání se vytvořily textové souboru s konfigurací všech sítí a s posledním stavem o připojení.

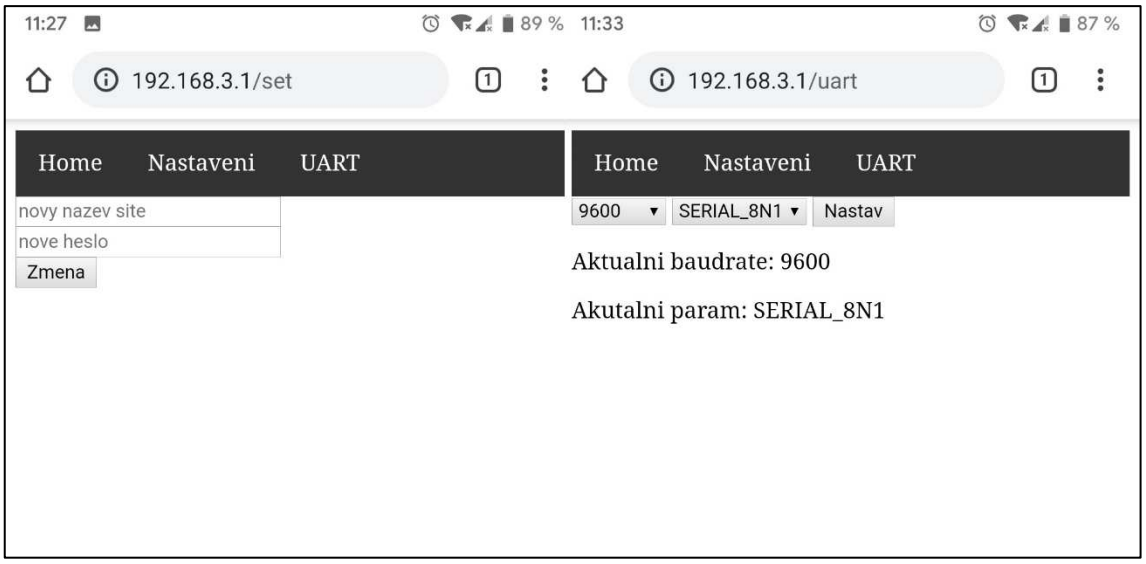


Obrázek 12: Hlavní stránka v MASTER režimu

Po připojení přes WiFi k modulu a zadání výchozí adresy do webového prohlížeče, v tomto případě chytrého mobilu, se uživateli zobrazí tzv. index (defaultní hlavní stránka). Po prvním spuštění je nastavený režim MASTER, to je znázorněno v pravém okraji navigační lišty.

V záložce NASTAVENÍ je zobrazený formulář s políčky pro změnu parametrů aktuální lokální AP sítě. Při špatném zadání jakéhokoliv parametru, uživatele upozorní chybová hláška objevující se pod formulářem. Zadaný text pro políčko s heslem je skrytý.

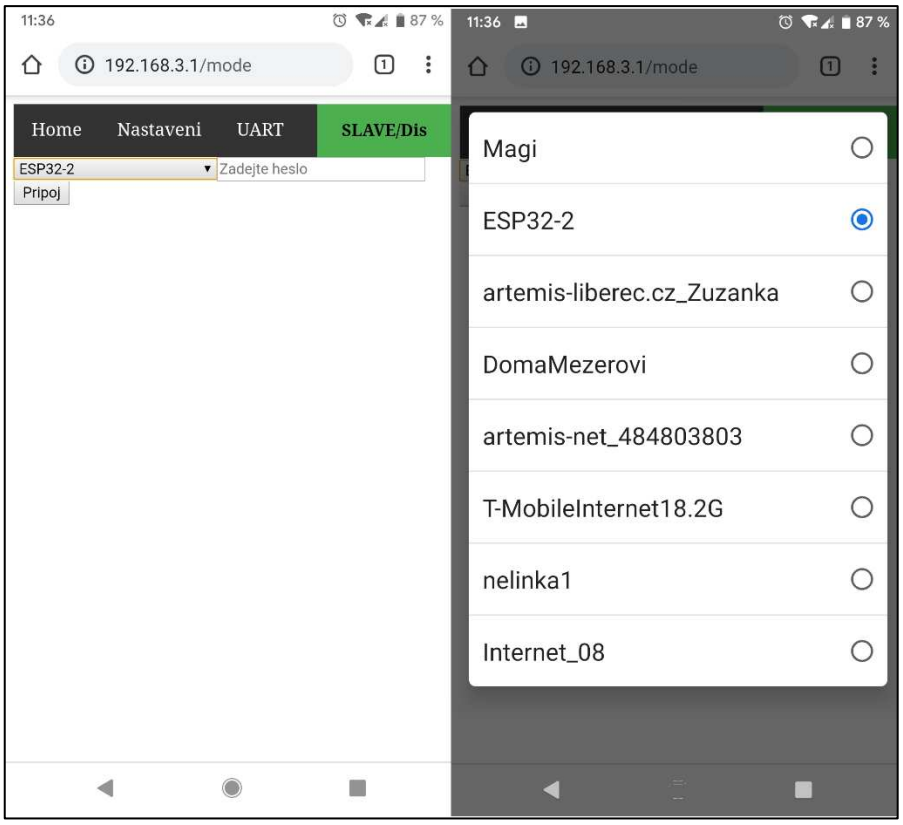
Na stránce UART jsou předpřipravené selectboxy se všemi definovanými parametry. Zde je zakomponována informační hláška o aktuálním nastavení parametrů na UART2. Obsah stránky znázorňuje Obrázek 13.



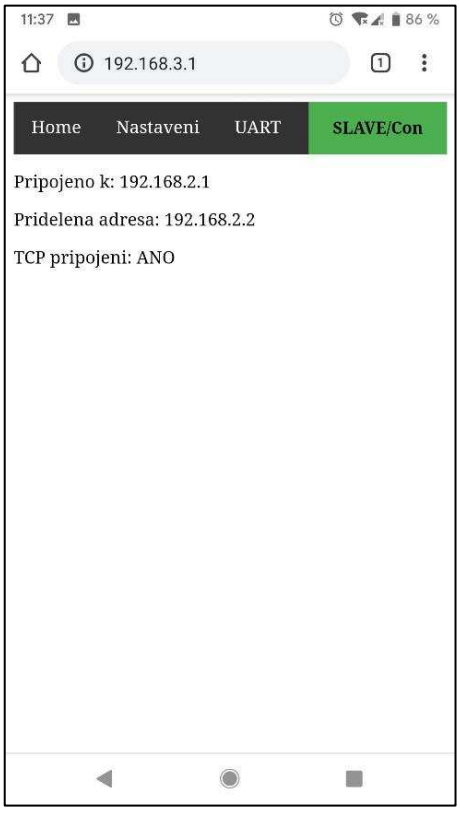
Obrázek 13: Stránka nastavení a uart

Po změně režimu, kliknutím na zelený tlačítko s nápisem MASTER (viz. Obrázek 14), se v případě přepnutí na mód SLAVE vyvolá funkce pro skenování dostupných okolních sítí. Seznam pro výběr je znázorněn v pravé části Obrázek 14. Po výběru sítě a zadání hesla se provede pokus o připojení a žádost o vydání adresy DHCP serverem. Po udělení IP adresy dojde k připojení na definovaný port, kde je nastaveno TCP spojení. Po všech těchto úkonech v jádru programu, jsou informace pomocí skriptu vypsány na hlavní stránku.

Tlačítko v pravém rohu se tedy mění v závislosti na zvoleném módu a na připojení. Na obrázku je vidět, že byl mód změněn, avšak nebylo provedeno připojení. Následujícím obrázkem je znázorněn index, vypsané informace a tlačítko s nápisem „Con“ míněno jako připojeno k druhému modulu.



Obrázek 14: Stránka režimu SLAVE a výběr sítě



Obrázek 15: Hlavní stránka po připojení

6. Závěr

Seznámil jsem se s dostupnými moduly a vývojovými kity s ESP32. Podle datasheetu zvoleného modulu jsem usoudil, jaký UART mohu využívat pro tuto aplikaci. Pak proběhla rešerše převodníků na sběrnici RS 485. Volbou byl obvod MAX485, protože se vyrábí ve verzi pro nižší napájení a je zde snadno dostupný k objednání.

Pro nastavování modulu jsem vytvořil webový server, kde je možné ovládat vše, co uživatel potřebuje. Bylo nadále nutné využít flash paměť pro ukládání celého textového souboru s konfigurací sítí. Pro komunikaci po sběrnici byl utvořen algoritmus na přepínání mezi režimem čtení a zápisu.

V softwaru Eagle, spadající pod Autodesk, jsem postupně navrhoval zdroj a připojení periferií k modulu. Pro připojení sběrnice jsem použil 4vodičovou svorku po jistém doporučení. Návrh desky plošných spojů doprovázely komplikace v rozmístění součástek. Chtěl jsem docílit toho, aby deska byla leptána jednostranně. Některé křížení cest se neobešlo bez použití propojky.

Vytvořené prototypy se osadily součástkami nutné pro funkčnost aplikace. Nadále proběhlo testování stability při zadávání požadavků z webového rozhraní. Poté došlo ke zkoušce posílání dat mezi moduly a pak jsem zapojil i UART/USB převodníky k testování posílání dat. K jednomu počítači byly připojené tedy dva porty, představující UART. Ve vytvořených terminálech jsem z jednoho portu posílal data a druhý port jsem měl otevřený pro příjem dat. Toto testování bylo úspěšné.

Pro test RS485 byla provedena smyčka, která posílala data jedním směrem dlouhodobě. Proběhlo zjištění ztráty dat a to právě na sběrnici. Tento problém byl následně eliminován pozdržením příkazu k odeslání dat, když data ještě nebyla odeslána.

Celé toto řešení bezdrátového přenosu z průmyslové sběrnice a UART se dá využít v prostředích, kde není možné kabel umístit či je jeho umístění nežádoucí.

Použitá literatura

[1] *Espressif Systems* [online]. 2019 [cit. 2020-01-05]. Dostupné z: https://www.espressif.com/

[2] *Espressif Systems: Modules* [online]. 2019 [cit. 2020-01-05]. Dostupné z: https://www.espressif.com/en/products/hardware/modules

[3] *Esp-iot-solution: Introduction to the ESP-Prog Board* [online]. 2019 [cit. 2020-01-05]. Dostupné z: https://github.com/espressif/esp-iot-solution/blob/master/documents/evaluation_boards/ESP-Prog_guide_en.md

[4] *Esp32-wroom-32d_esp32-wroom-32u_datasheet_en* [online]. 2019 [cit. 2020-01-05]. Dostupné z: https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32d_esp32-wroom-32u_datasheet_en.pdf

[5] *AP3211.pdf* [online]. 2019 [cit. 2020-01-05]. Dostupné z: https://www.diodes.com/assets/Datasheets/AP3211.pdf

[6] *Espressif: arduino-esp32* [online]. [cit. 2020-01-05]. Dostupné z: https://github.com/espressif/arduino-esp32/tree/master/libraries

[7] *MAX13487E-MAX13488E-1515036.pdf* [online]. 2015 [cit. 2020-01-05]. Dostupné z: https://cz.mouser.com/datasheet/2/256/MAX13487E-MAX13488E-1515036.pdf

[8] *ESP8266* [online]. 2019 [cit. 2020-01-05]. Dostupné z: https://www.esp8266.com/

[9] Espressif. In: *Modules* [online]. 2019 [cit. 2020-01-05]. Dostupné z: https://www.espressif.com/sites/default/files/modules/esp32-wroom-32d-01-01-2.png

[10] Espressif. In: *The Internet of Things with ESP32* [online]. 2018 [cit. 2020-01-05]. Dostupné z: http://esp32.net/images/_resources/ESP32_Function_Block_Diagram.svg

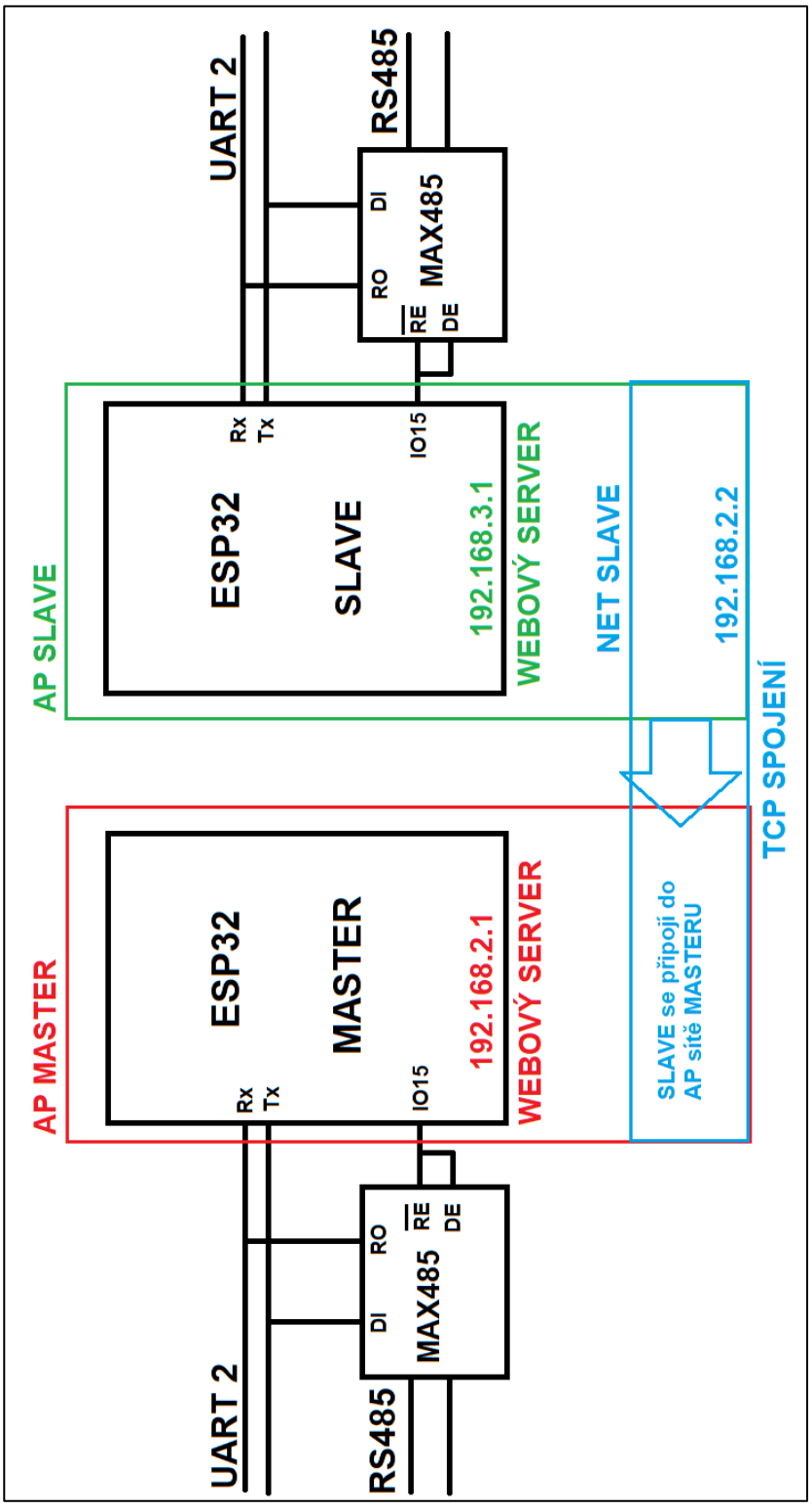
[11] In: *Modul ESP-01 ESP8266* [online]. 2019 [cit. 2020-01-05]. Dostupné z: https://www.hamshop.cz//data/product/272_417.jpg

[12] Espressif. In: *Development Board* [online]. 2018 [cit. 2020-01-05]. Dostupné z: https://www.espressif.com/sites/default/files/dev-board/ESP32-DevKitC-32D-F_01_0.jpg

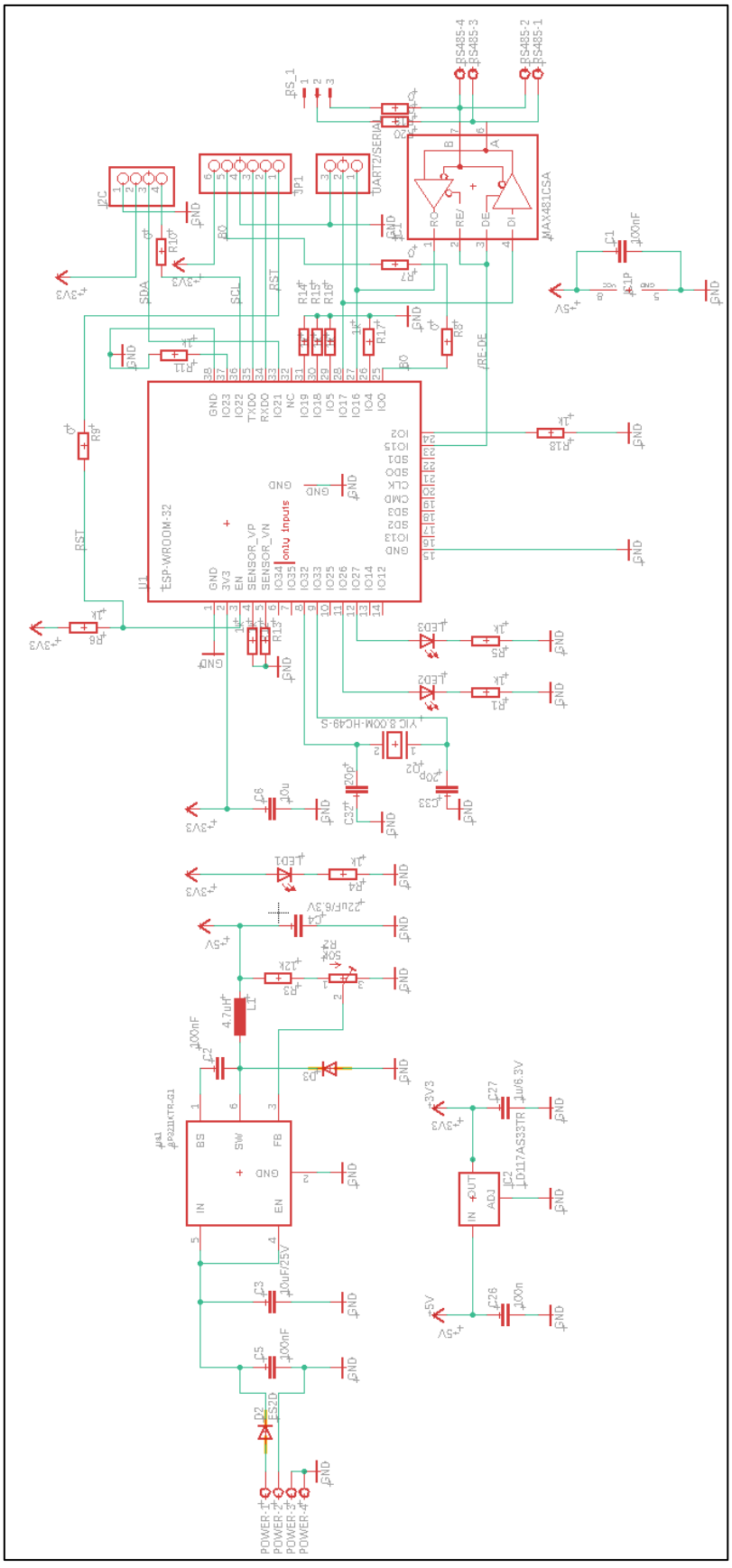
[13] Espressif. In: *Development Board* [online]. 2019 [cit. 2020-01-05]. Dostupné z: https://www.espressif.com/sites/default/files/dev-board/esp-wrover-kit-vb-s-1_0.png

Přílohy

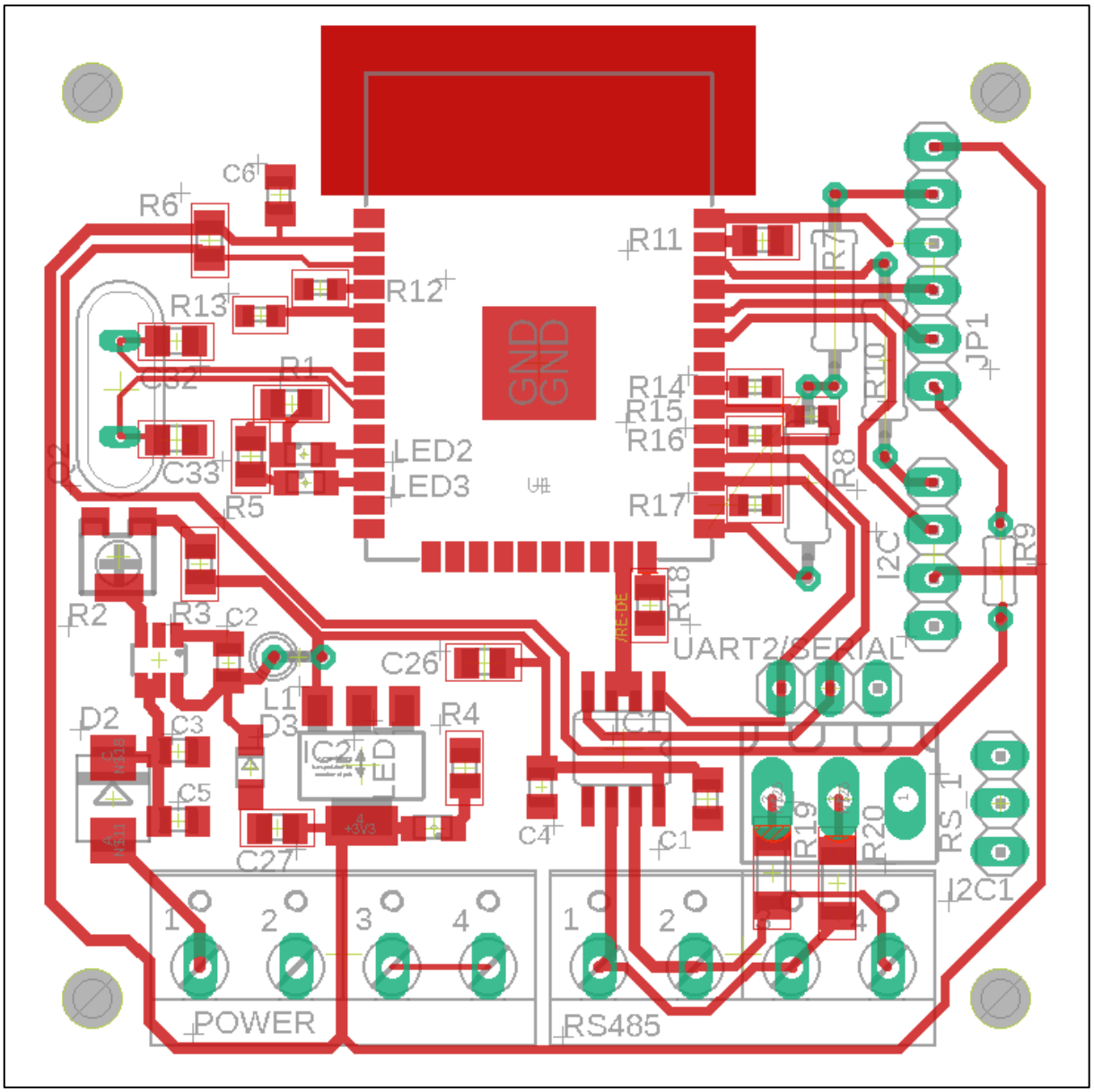
A. Blokové schéma



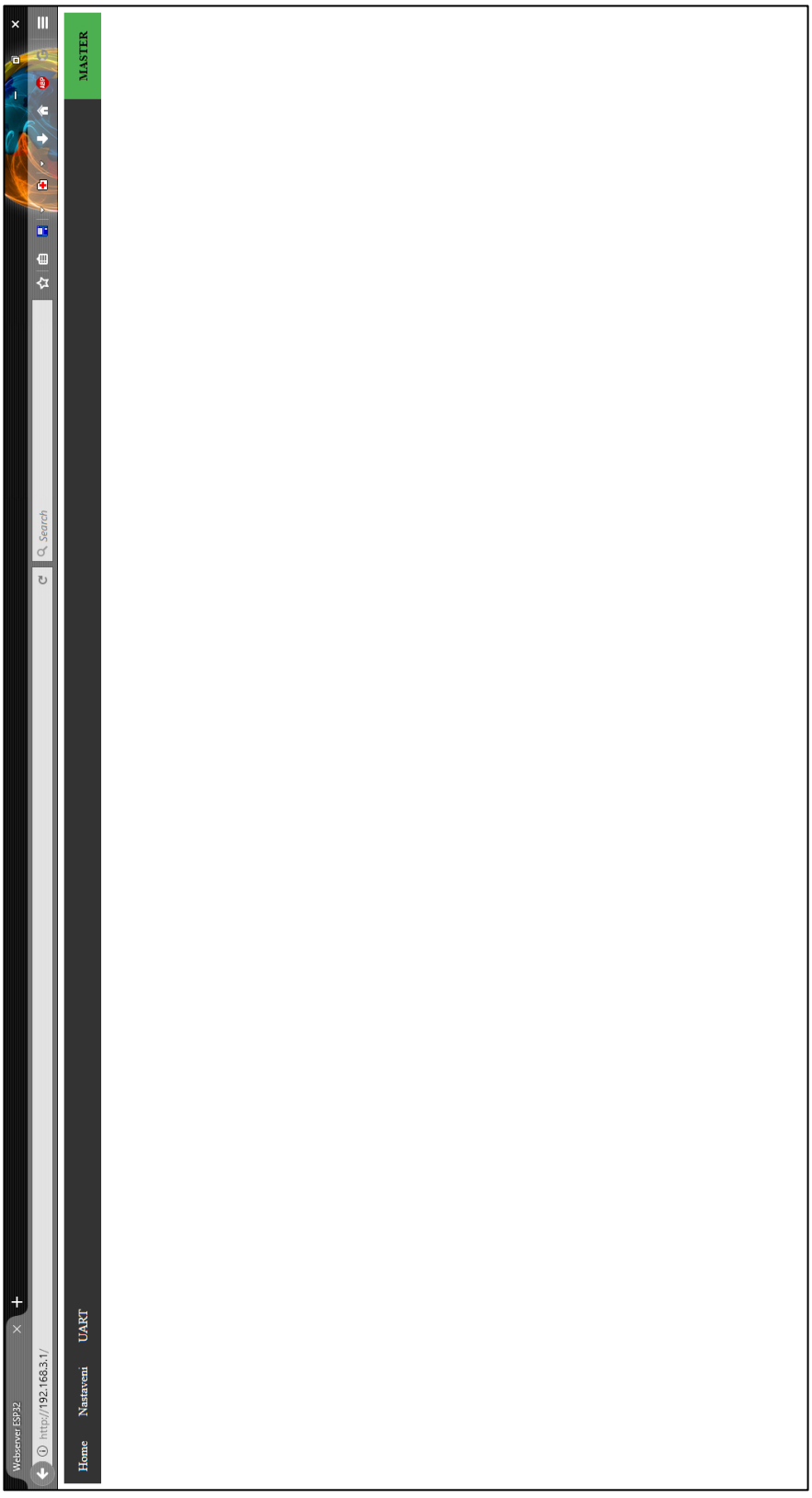
B. Celé schéma se zdrojem



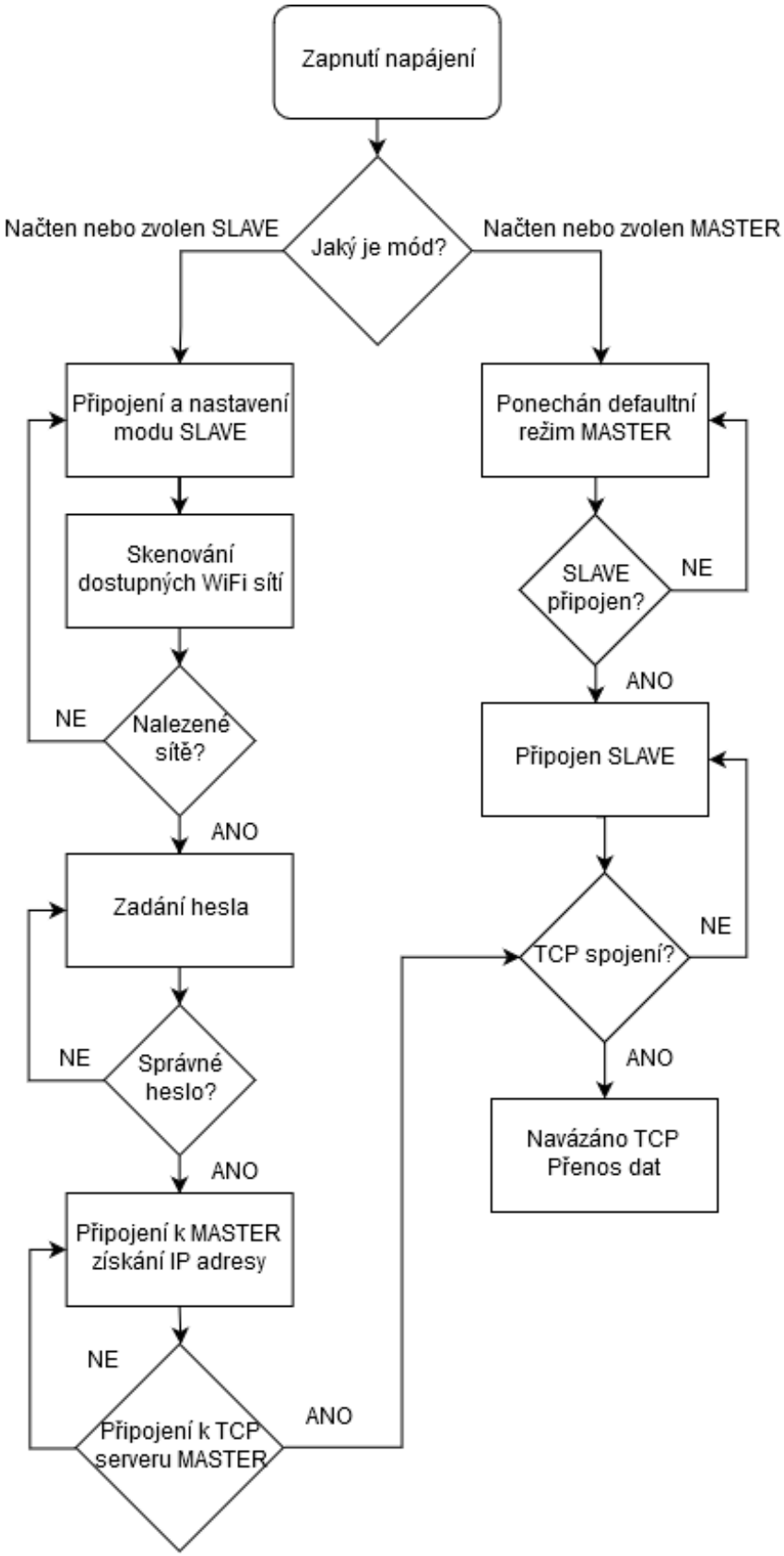
C. DPS s popisky



D. Webové rozhraní na PC (rozlišení 1920x1080)



E. Vývojový diagram spojení



F. Obsah přiloženého CD

- Text bakalářské práce
  - bakalarska_prace_2019_Filip_Hess.doc
  - bakalarska_prace_2019_Filip_Hess.pdf
- Zdrojový kód
  - web.ino
- Výkresová dokumentace
  - Schéma (esp32_FH.sch)
  - Deska (esp32_FH.brd)
- Katalogové listy
  - AP3211x.pdf
  - Esp32-wroom.pdf
  - Max485.pdf