SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Univerzálny IoT Wifi modul s dlhým dosahom

NAVEZ – Návrh elektronických zariadení

Dokumentácia k semestrálnemu projektu

Oliver Hollý

Obsah

Úvod

Internet vecí spolu so štvrtou priemyselnou revolúciou sa pomaly prejavuje aj v osobných a užívateľských aplikáciách a tak vzniká rozmanité pole domácej automatizácie. Tento technologický smer zaujal aj mňa, a tak som sa zoznámil s elektronickými platformami využívanými v tejto oblasti. Výber elektronickej platformy na IoT aplikáciu je podmienený možnosťou pripojenie sa na internet, resp. wifi. Medzi najznámejšie IoT hardvérové platformy pre DIY projekty patrí určite Arduino a Raspberry, prípadne Intel. Avšak objavili sa na trhu moduly od Espressif na báze 32 bitových MCU ESP8266. Táto platforma má výborný výsledok v pomere ceny a kvality, preto je čoraz viac obľúbenejšia podporovaná.

Rozhodol som sa, že si vyskúšam navrhnúť vývojovú dosku, resp. modul na báze spomínanej ESP platformy. Takýto modul musí zabezpečovať energetické napájanie, stabilizáciu napájania, ošetrenia GPIO pinov, programovacie a bootovacie zapojenia, vďaka čomu sa uľahčí a zrýchli vývoj IoT projektov. Modul môže obsahovať prípadne špecifický senzor alebo hardvérovú prípravu na konkrétny projekt (napr. napäťový delič).

Počas mojich minulých experimentoch na projektoch som používal platformu NodeMCU na báze modulu ESP-12, ktorý disponuje internou Wifi anténou na plošnom spoji. Takáto anténa je efektívna z hľadiska priestoru a veľkosti, čo je často potrebné pri IoT projektoch. Avšak jej veľkosť a gain je úmerná dosahu, čo sa prejavilo ako nedostatočný faktor pri jednom mojom projekte, kedy som potreboval prijímať Wifi signál z domu v exteriéry. Riešením sa zdalo byť použitie podobného modulu ESP-07 s konektorom na externú anténu, ktorá ma spravidla lepšie vlastnosti. Je zaujímavé, že pre tieto moduly nie je veľký výber platforiem a vývojových dosiek.

Preto cieľom tejto práce bude návrh a realizácia IoT modulu na báze ESP-07 (MCU ESP8266) so všetkými potrebnými komponentami a jedným senzorom teploty a vlhkosti. Výsledkom bude univerzálna vývojová IoT platforma so zameraním sa na dosah prijímania Wifi signálu.

1 Návrh schémy zapojenia a dosky plošného spoja

Cieľom návrhu hardvérovej platformy je grafický obraz dosky plošného spoja, ktorý sa dá následne vyrobiť komerčnej spoločnosti. Prvým krokom bude schematický návrh zapojení a potom priestorové rozloženie komponentov na DPS. Vývoj budem vykonávať v programe KiCad.

1.1 Výber wifi modulu

Návrh platformy univerzálneho IoT modulu je založený na Wifi module, ktorého výber bol zdôvodnený v úvode. Na Obr. 1.1 je vidieť rozdielnosť v implementácií antény medzi ESP12 a ESP-07.

Obr. 1.1. Moduly ESP-12 (vľavo) a ESP-07 (vpravo)

1.2 Ošetrenie GPIO pinov

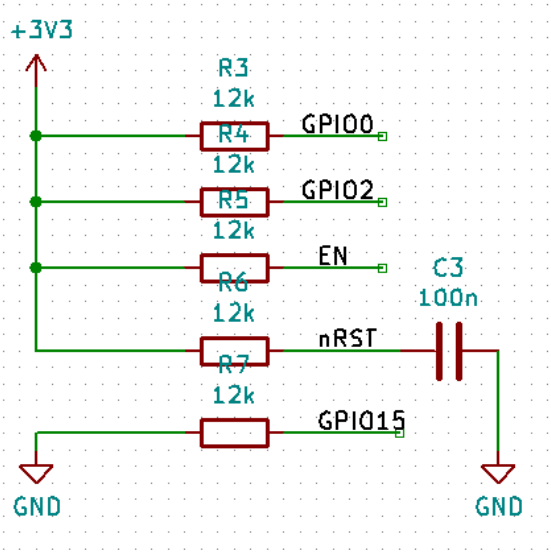
Ošetrenie GPIO pinov znamená zapojiť ich externe tak, aby boli vždy (aj po resete) v dovolenom definovanom stave. To je dôležité napríklad pri bootovaní procesoru podľa Obr. 1.2, alebo pri alternatívnych funkciách (napríklad I2C bus, externé prerušenie atď.)

Table 4 Pin Mode

Mode	GPIO15	GPIO0	GPIO2
UART	low	low	high
Flash Boot	low	high	high

Obr. 1.2. Módy ESP8266 podľa stavov GPIO pinov po resete

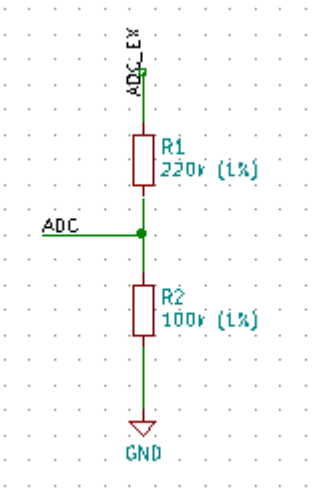
Z katalógového listu a informácií od výrobcu je potrebné počas každého Bootovania, resetovania, alebo prebúdzania zo spánku držať pin GPIO15 na LOW a pin GPIO2 na HIGH napäťovej úrovni. Definovaním napäťovej úrovne po resete na pine GPIO0 určujeme, či sa bootuje program z pamäte alebo sa ide nahrávať nový (FLASH). V prípade, že chceme MCU prebúdzať zo spánku musíme pin GPIO16 prepojiť s RST pinom, na čo som pri návrhu zabudol ale vo výslednom zariadení som to opravil prepojovacím káblom. Piny EN (enable) a RST (reset) majú byť taktiež default pull up.



Obr. 1.3. Definovanie napäťových úrovni na pinoch

1.3 Vstup AD prevodníka

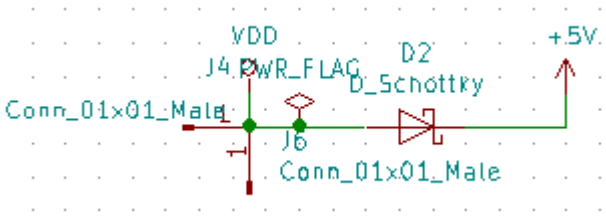
ESP8266 disponuje 10-bitovým AD prevodníkom s rozsahom 0-1 V. Pre intuitívnejšie a praktickejšie používanie som rozsah zväčšil na 0-3.3 V napäťovým deličom podľa Obr. 1.4. Trieda presnosti a tepelná závislosť rezistorov ma potom vplyv na meranie.



Obr. 1.4. Napäťový delič AD prevodníka

1.4 Ošetrenie prepoľovania

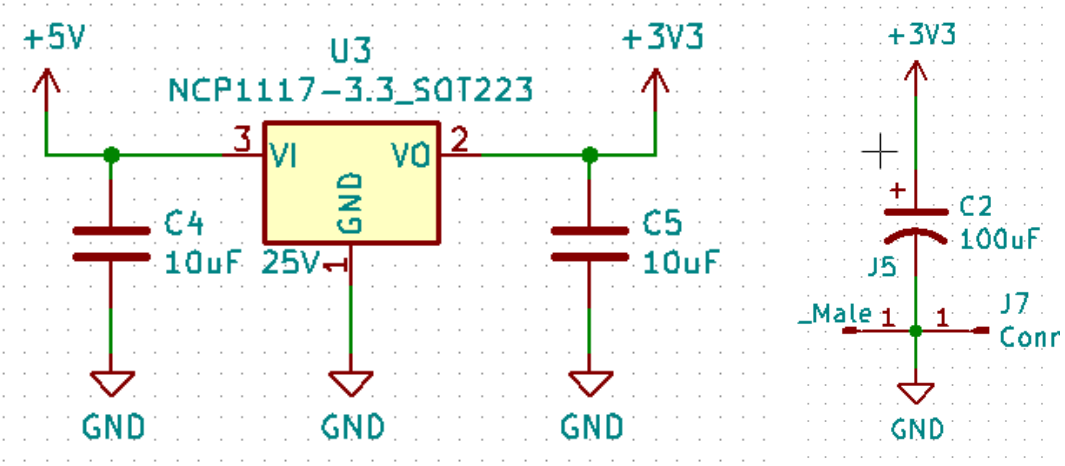
Základným ošetrením napájacieho vstupu je implementácia schottky diódy podľa Obr. 1.5.



Obr. 1.5. Ošetrenie vstupu napájania proti prepaľovaniu

1.5 Stabilizácia napájania

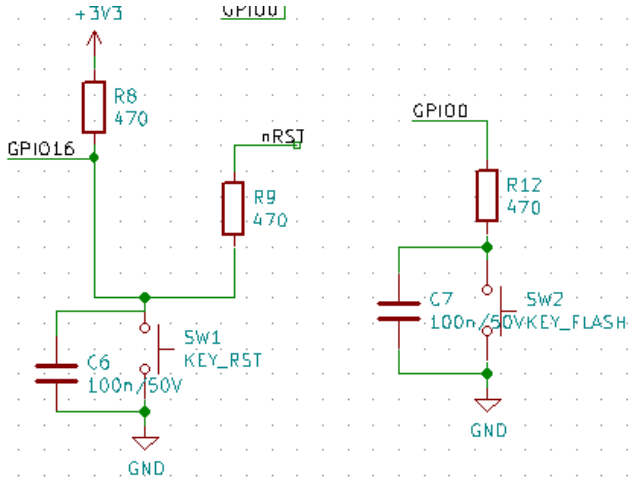
ESP8266 pracuje na 3.3V napájaní. Pre zabezpečenie stability a robustnosti sa používa zdroj s väčším napätím (napríklad 5V), ktoré je teplotne regulované na 3.3 V cez LDO regulátor napätia. Napájacia sústava na Obr. 1.6 obsahuje stabilizačné a filtračné kondenzátory.



Obr. 1.6. Napájacia sústava

1.6 Programovacie a bootovacie nastavenia

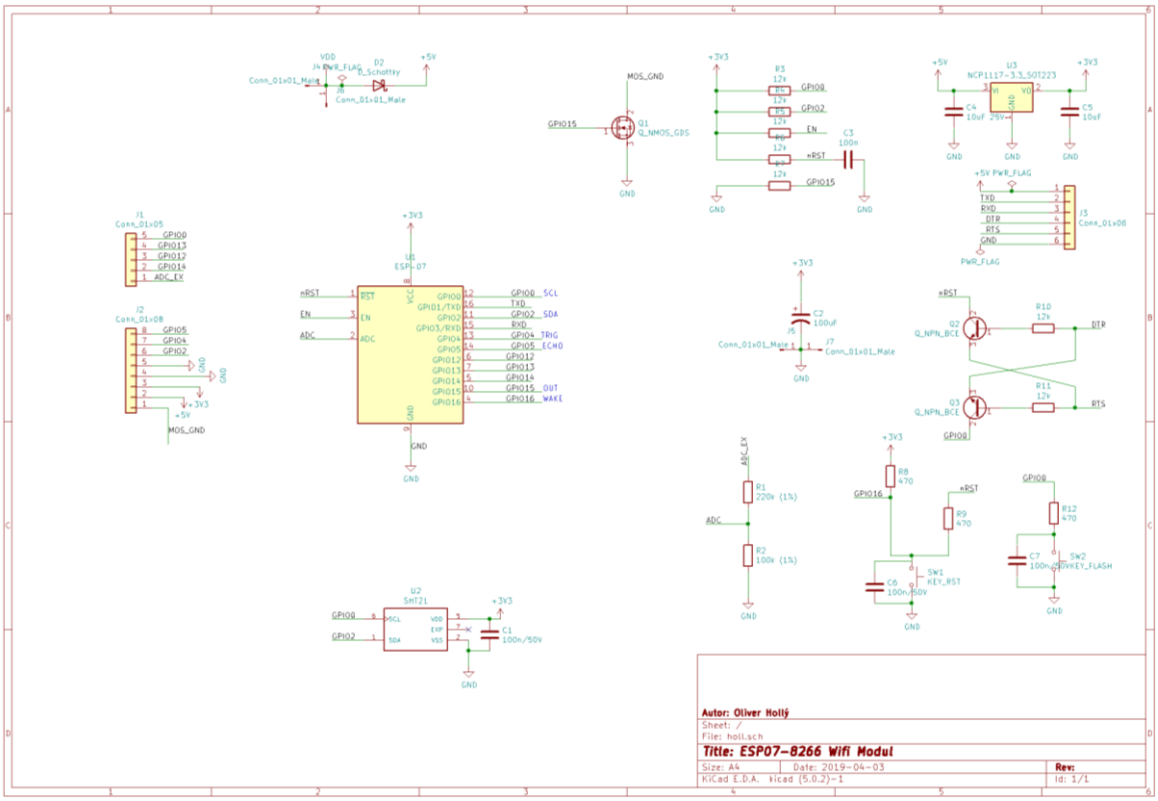
Zmenu definovaných stavov pinov procesora, ktorým meníme módy bootovania a resetujeme MCU vykonávame implementovaním tlačidlových spínačov podľa Obr. 1.7.



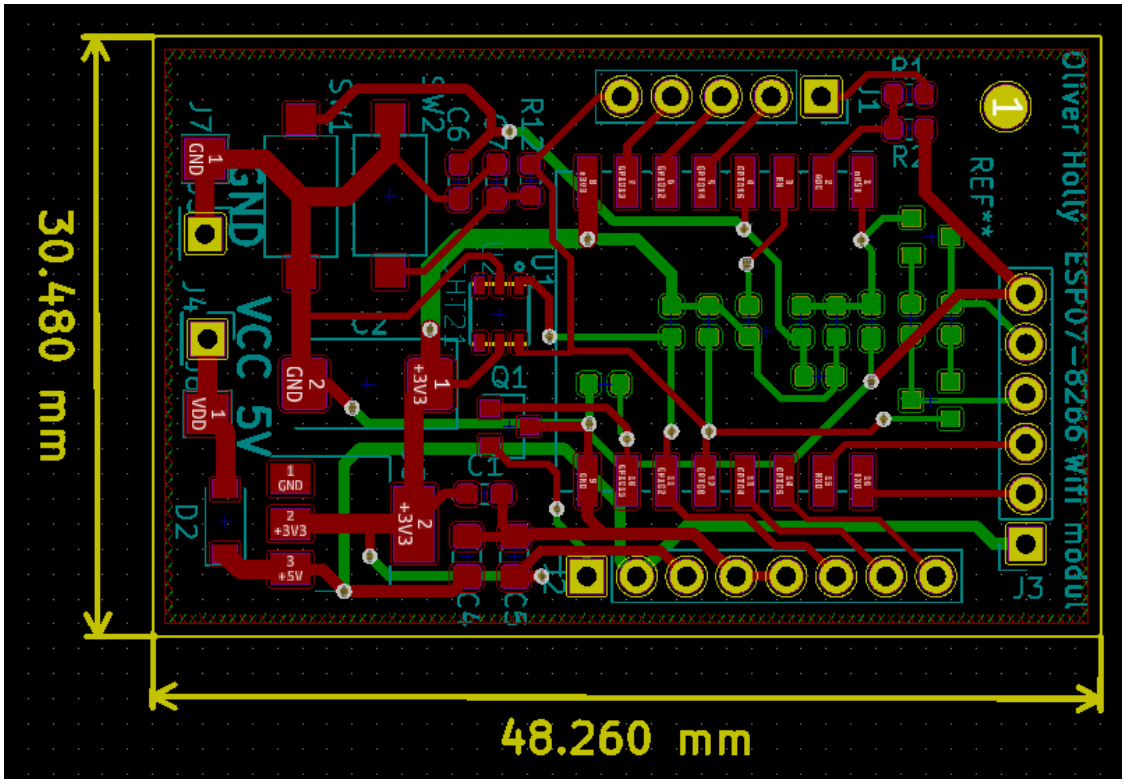
Obr. 1.7. Manuálne nastavenie definovaných úrovni

1.7 Výsledný návrh

Na Obr. 1.8 je znázornená kompletná schéma zapojenia v programe KiCad. Z netlistu schémy vygenerujeme súbor footprintov, ktoré následne vhodne rozložíme na dosku plošného spoja podľa Obr. 1.9. Vidíme použitie hrubších ciest na napájacie vedenia a vývody nepoužitých pinov na pinové zbernice pre potenciálne využitie.



Obr. 1.8. Kompletná schéma obvodu z programu KiCad



Obr. 1.9. Návrh plošného spoja obvodu bez rozliatej zeme

Tab. 1. Zoznam súčiastok č.1

Amount	Reference	Desciption	Type	Value	Price
1	U1	ESP07	SMD	8266	5,03
1	U3	LDO	SMD	3.3 V	0,37
2	SW1-2	Switch	SMD		0,24
1	C2	Capacitor	SMD	100uF	0,83
2	Q2-3	Transistor	SMD	NPN	0,5
1	Q1	MosFET	SMD		0,09
1	D1	Schottky	SMD		0,09
20	R	Resistor	SMD	12k	0,02
4	C1, C3, C6, C7	Capacitor	SMD	100n	0,08
2	C4, C5	Capacitor	SMD	10u	0,07

Spolu: 7,32 eur

Tab. 2. Zoznam súčiastok č. 2

Amount	Reference	Desciption	Type	Value	Price
1	Q1	SHT21	SMD		5
1	USB-TTL	prevodník	modul		0.5
1	Anténa	Externá		2.4 GHz	1

Spolu: 6.5 eur

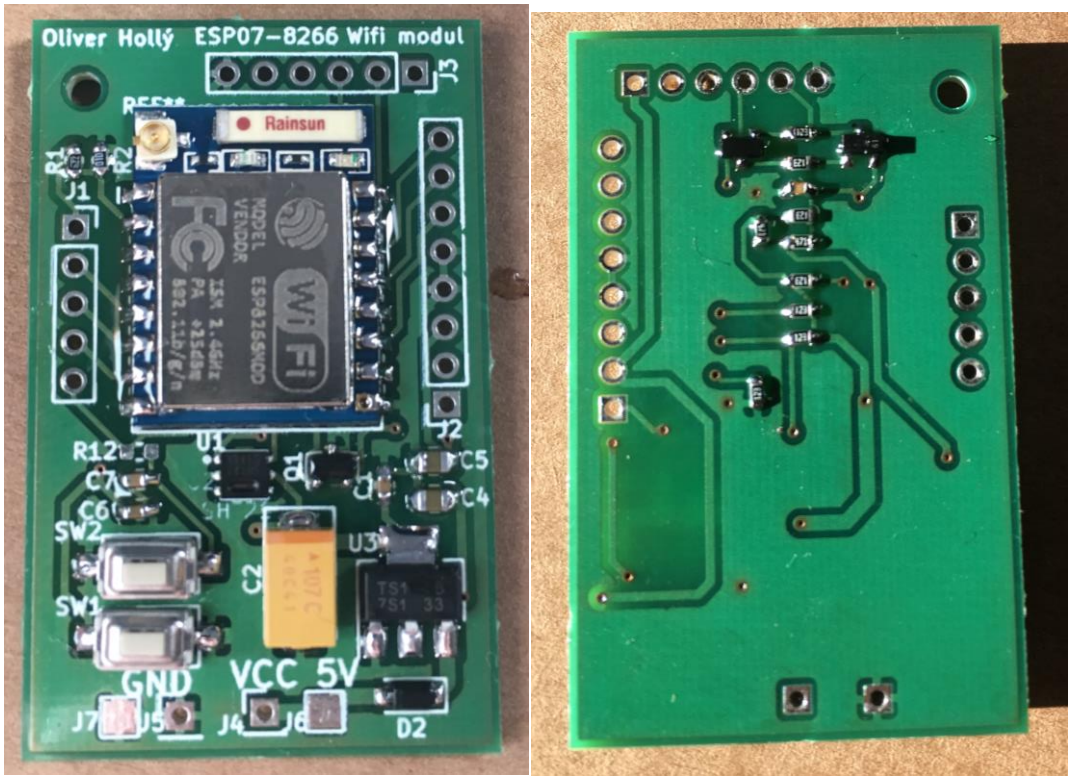
1.8 Zhodnotenie návrhu

Návrh zapojenia a osadenia DPS zabezpečuje funkčnosť, univerzálnosť a energetickú stabilitu pre modul ESP-07. Schéma zapojenia obsahuje minimálne množstvo potrebných komponentov pre možnosť programovania a bootovania. Výsledný návrh bol vo forme gerberu poslaný na výrobu. Plošný spoj prišiel vyrobený podľa žiadosti, čím sa overila vyrobiteľnosť návrhu.

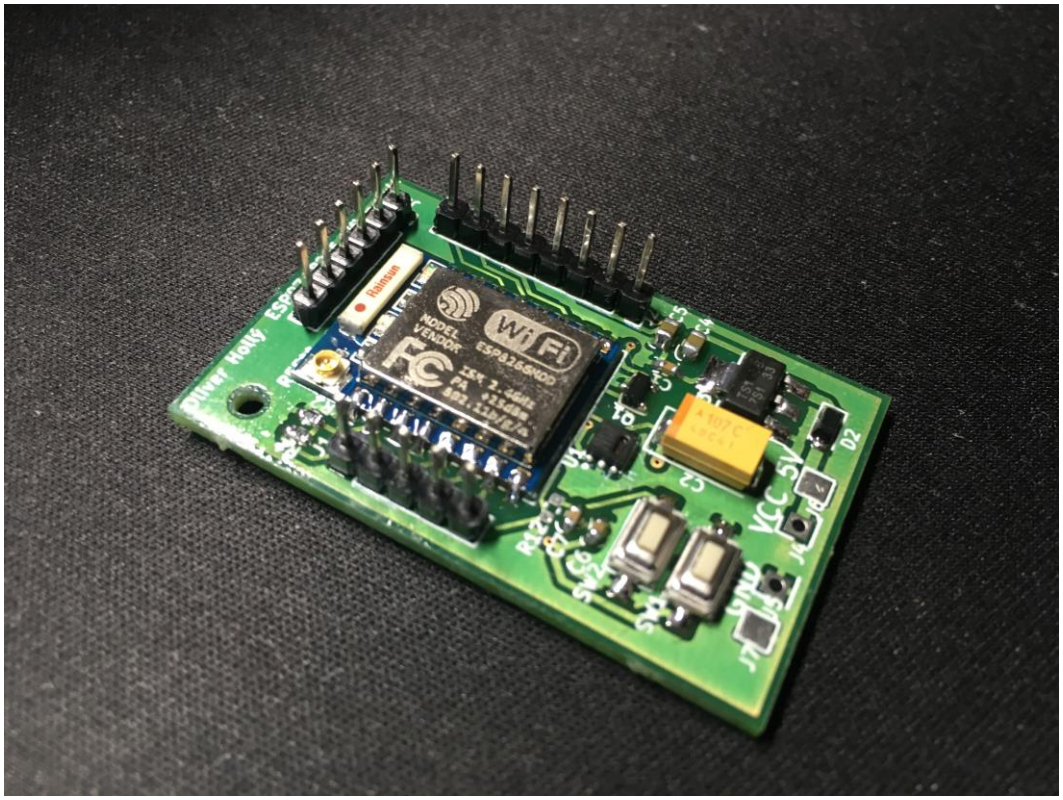
V Tab. 1 je zoznam základných súčiastok hradených školou. V Tab. 2 sú prídavné komponenty, ktoré som zabezpečil sám z dôvodu finančnej náročnosti a rovnosti. S prihliadnutím na ďalšie finančné náklady vyhradené na výrobu DPS je výsledná suma návrhu veľmi vysoká v porovnaní s konkurenciou z Číny. Tento fakt je spôsobený cenovou maržou lokálnych obchodníkov, odkiaľ boli súčiastky kupované. Ďalším faktorom vysokej ceny je prototypová výroba bez cenových zrážok objednávania malých počtov komponentov. Výsledná cena zariadenia teda vychádza približne na 15 eur. Bez kvalitného snímača teploty SHT21 by bola cena cca 10 eur, podobný modul je možné objednať z Číny za 2 eura.

2 Výroba a osadenie DPS

Výsledný návrh dvojvrstvového plošného spoja som osadil navrhnutými súčiastkami. Snažil som sa modul navrhnúť, čo najmenší, preto som používal iba SMD komponenty. Všetky súčiastky som osádzal ručne mikrospájkou, okrem snímača SHT21, ktorého puzdro bolo lepšie osadiť fúkaním. Výsledný osadený DPS je na Obr. 2.1. a Obr. 2.2.



Obr. 2.1. Osadený plošný spoj



Obr. 2.2. Osadený plošný spoj

3 Softvérová implementácia

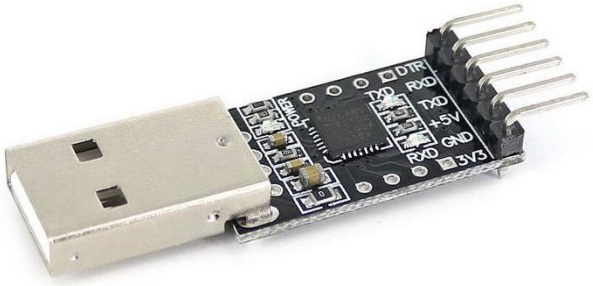
Požadovanú funkčnosť a vlastnosti modulu budeme nastavovať programovaním procesoru ESP8266. Programovanie ako činnosť abstrakcie spracovania dát pomocou algoritmov a funkcií realizujeme všeobecne definovanými textovými metodikami v programovacom jazyku. Všeobecne každý program sa vykonáva v procesore na úrovni tranzistorov a elektrických signálov. Programovací prístup sa odlišuje spomínanou abstrakciou nad hardvérom.

Ako som spomenul ESP platforma nadobúda na obľúbenosti a používateľnosti, vďaka čomu sa abstrakcia nad procesorom ESP8266 preniesla až na portfólio Arduina. Pri programovaní na takejto relatívne vysokej úrovni dochádza k stretu medzi kvantitou a kvalitou softvérového výstupu. Vzhľadom na nekritickosť a rôznorodosť IoT zariadení je rýchlosť a jednoduchosť vývoja vítaná.

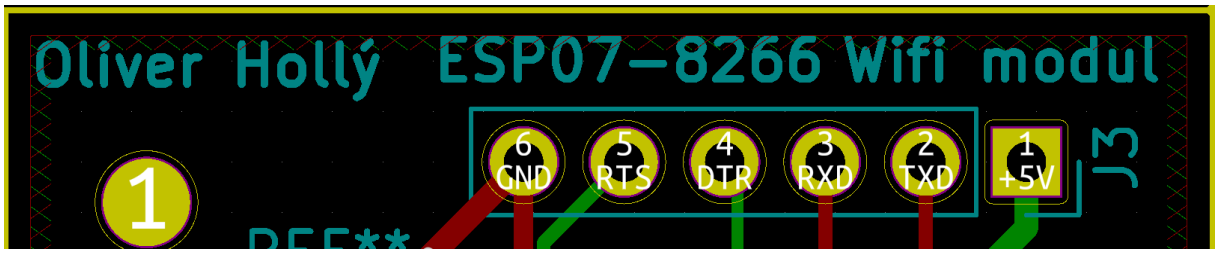
3.1 Programovanie modulu a vývojové prostredie

Zdrojový kód píšem teda v Arduino IDE s ESP8266 package podporou. Skompilovaný binárny program je nutné nahrať do pamäte procesora. Dátový prenos sa realizuje po sériovej linke z PC v podobe USB a pre MCU v podobe UART protokolu. Preto je nutné použiť USB-TTL prevodník na Obr. 3.1. S protokolom UART pracujeme s dvomi dátovými linkami RX a TX.

Na Obr. 3.2 vidíme usporiadanie vývodov používaných na falshovanie programu. Dátové linky prepojíme s USB-TTL prevodníkom do kríža a piny RTS a DTR priamo.

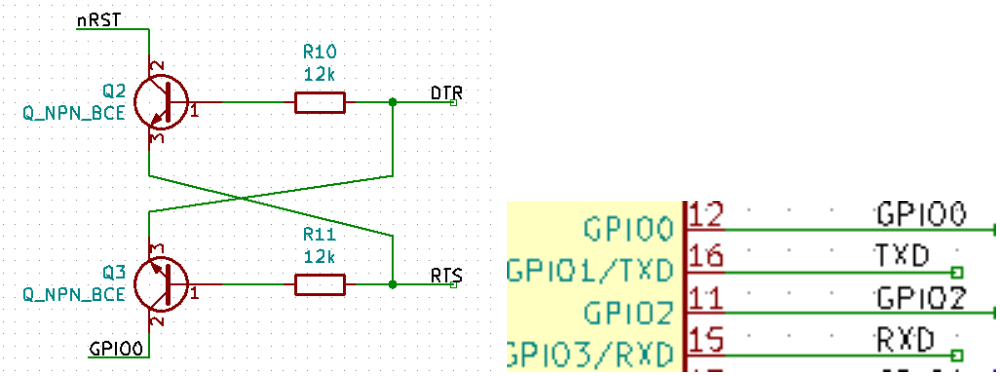


Obr. 3.1. USB-UART prevodník



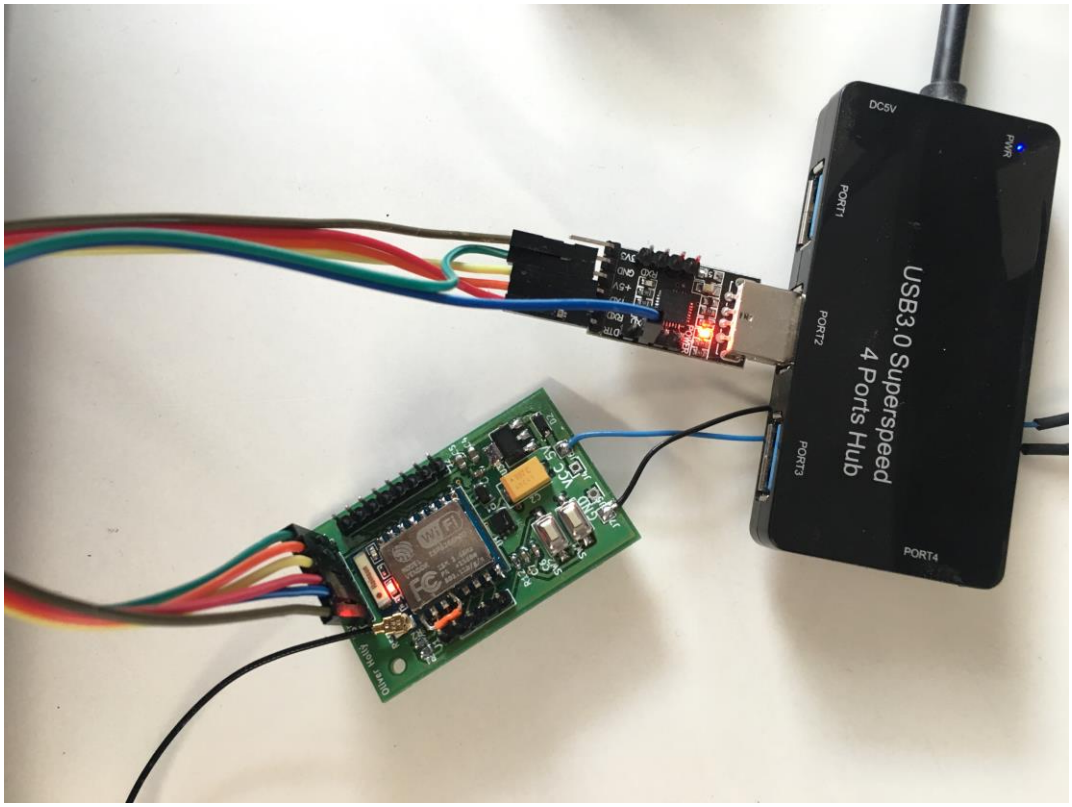
Obr. 3.2. Programovacie vývody vrchnej lišty na DPS

Pinmi RTS a DTR sme zabezpečili automatickú informáciu pre procesor, že sa chystá nahrávanie nového programu. Princíp vychádza zo zapojenia na Obr. 3.3 a definovaných požiadaviek na stavy pinov z kapitoly 1.



Obr. 3.3. Programovacie piny a zapojenie pre automatické flashovanie

Na Obr. 3.4 je vidieť zapojenie sústavy pripravenej na flashovanie programu do pamäte mikroprocesora. Dôležitá informácia pred prvými pokusmi s programovaním ESP modulov je fakt, že ich prúdový odber prevyšuje výkon LDO regulátora 3.3V na UART prevodníku. Z toho je zrejmé, že nemôžeme modul napájať z regulovaného 3.3 V výstupu prevodníka, ale musíme využiť 5V a regulovať na 3.3V regulátorom osadeným na module. Ak by sme postupovali inak odpálili by sme regulátor na prevodníku, prípadne celý poškodili. Aj preto som v návrhu celú pinovú hornú lištu (Obr. 3.2) pripravil na programovanie aj s 5V vstupom.



Obr. 3.4. Programovanie modulu

3.2 Vizualizácia a monitorovanie dát na internetovom servery

Podstatou internetového pripojenia v IoT aplikáciách je okrem vzájomnej komunikácie zariadení aj možnosť zdieľania dát na webových serveroch. Internetová infraštruktúra je výhodná kvôli širokej prístupnosti z rôznych zariadení na rôznych miestach.

S modulom ESP-07 dokážeme posielať dáta na existujúci server alebo ho dokonca môžeme vytvoriť priamo v ňom. Avšak vytvorený server je bezpečne prístupný iba zariadeniam v lokálnej sieti LAN (váš domáci internet - router), keď chceme získať vzdialený WAN prístup k serveru na ESP musíme v nastaveniach routera povoliť otvorenie portov, čo je nebezpečné z hľadiska bezpečnosti a ochrany súkromia celej LAN siete.

Preto je lepšie využívať vizualizáciu dát na bezpečnom serveri. Ja som pri tejto práci využil platformu ThingSpeak, kde vizualizujem dáta to snímača teploty a vlhkosti. Táto platformu má navyše softvérovú podporu pre Arduino vývoj, čiže je to jedna z najrýchlejších implementácií IoT monitoringu.

3.3 Low power módy

Zariadenia internetu vecí je nutné občas umiestniť na miesta bez priameho prístupu elektrickej energie. Vtedy má zariadenie napájanie z baterky. Pre zabezpečenie dlhodobej funkčnosti je nutné minimalizovať spotrebu elektrickej energie. Idea je v tom, že vykonávame meranie a zdieľanie dát len v určitých časových intervaloch. Medzi jednotlivými meraniami uvedieme zariadenie do low-power módu alebo spánku. V týchto módoch má procesor minimálnu spotrebu (viď Obr. 3.5).

Table 1-1. Differences between 3 Sleep Modes

Item		Modem-sleep	Light-sleep	Deep-sleep
Wi-Fi		OFF	OFF	OFF
System clock		ON	OFF	OFF
RTC		ON	ON	ON
CPU		ON	Pending	OFF
Substrate current		15 mA	0.4 mA	~ 20 μA
Average current	DTIM = 1	16.2 mA	1.8 mA	-
	DTIM = 3	15.4 mA	0.9 mA	
	DTIM = 10	15.2 mA	0.55 mA	

Obr. 3.5. Low power prúdové odbery

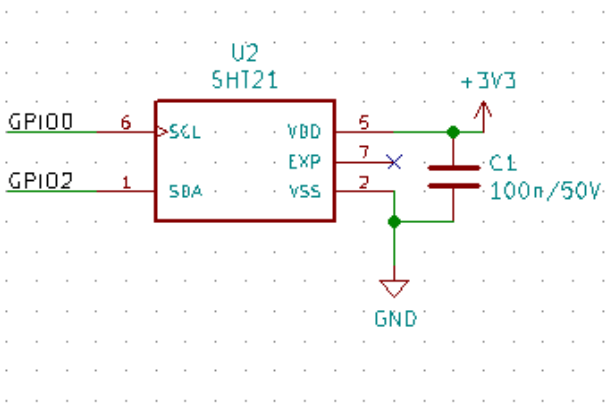
3.4 Spracovanie dát zo senzorov

Na modul som osadil aj kvalitný snímač teploz a vlhkosti SHT21. Komunikácia so snímačom prebieha cez I2C zbernicu. ESP8266 má možnosť variabilného konfigurovania pinov na alternatívnu funkciu I2C sériovej komunikácie. Ja som vybral ako komunikačné linky pre SDA a SCL piny GPIO0 a GPIO2, nakoľko už pri návrhu ošetrenia pinov som im musel pripojiť pull up rezistory, teda sú pripravené na túto alternatívnu funkciou a nemusím pridávať navyše dva rezistory. Kondenzátor pripojený paralelne k napájaniu snímača (Obr. 3.6) vychádza z odporúčaného zapojenia.

```
uint16_t readSensor(int reg)
{
        Wire.beginTransmission(SHT21::ADDR);
        Wire.write(reg);
        Wire.endTransmission(false);
        delay(100);     //sad but necessary
        Wire.requestFrom(SHT21::ADDR, 3);

        uint32_t timeout = millis() + 300;
        while (Wire.available() < 3) {
                if ((millis() - timeout) > 0) {
                        return 0;
                }
        }
        uint8_t H = Wire.read();
        uint8_t L = Wire.read();

        uint16_t result = (uint16_t)((H<<8) | L);
        result &= ~0x0003;   // clear two low bits (status bits)
        Wire.read();
        return (uint16_t)(result);
}
```



Obr. 3.6. I2C komunikácia a zapojenie snímača SHT21

```
#include <Wire.h>
#include <ESP8266WiFi.h>
#include "ThingSpeak.h"
#include <Ticker.h>

namespace I2C {
        static const uint8_t SDA = 2;
        static const uint8_t SCL = 0;
}

namespace SHT21 {
        static const uint8_t ADDR = 0x40;
        static const uint8_t TRIGGER_TEMP_HM  =  0b11100011;
        static const uint8_t TRIGGER_RH_HM    =  0b11100101;
        static const uint8_t TRIGGER_TEMP_NHM =  0b11110011;
        static const uint8_t TRIGGER_RH_NHM   =  0b11110101;
}

static const unsigned long myChannelNumber = 783703;
static const char * myWriteAPIKey = "4FVWE2XRBN9KZQ7O";

static const char ssid[] = "xxx";   // your network SSID (name)
static const char pass[] = "xxx";   // your network password

WiFiClient  client;

static const uint8_t sleepTime = 20;
Ticker ticker;

void setup() {
        Serial.begin(256000);
        Wire.begin(I2C::SDA, I2C::SCL);
        WiFi.mode(WIFI_STA);
        ThingSpeak.begin(client);
        ticker.attach(0.5, tick);

        if (WiFi.status() != WL_CONNECTED) {
                while (WiFi.status() != WL_CONNECTED) {
                        WiFi.begin(ssid, pass);
                        Serial.print(".");
                        delay(5000);
                }
                Serial.println("\nConnected.");
        }
}

void loop() {

        shareTemperature();
        ESP.sleep(20 * 1000000);
        shareHumidity();
        ESP.deepSleep(sleepTime * 1000000);
}
```

```
void tick()
{
  int state = digitalRead(BUILTIN_LED);  // get the current state of GPIO1
  digitalWrite(BUILTIN_LED, !state);     // set pin to the opposite state
}

void shareTemperature()
{
     long temperature = (long)getTemeperature();
     ThingSpeak.writeField(myChannelNumber, 1, temperature, myWriteAPIKey)
}

void shareHumidity()
{
       long humidity = (long)getHumidity();
       ThingSpeak.writeField(myChannelNumber, 2, humidity, myWriteAPIKey);
}

float getTemeperature()
{
 return (float)(-52.5+(175.72*readSensor(SHT21::TRIGGER_TEMP_NHM))/65536.0)
}

float getHumidity()
{
 return (float)(-6 + (125 * readSensor(SHT21::TRIGGER_RH_NHM))/65536.0);
}

uint16_t readSensor(int reg)
{
       Wire.beginTransmission(SHT21::ADDR);
       Wire.write(reg);
       Wire.endTransmission(false);
       delay(100);     //sad but necessary
       Wire.requestFrom(SHT21::ADDR, 3);

       uint32_t timeout = millis() + 300;
       while (Wire.available() < 3) {
              if ((millis() - timeout) > 0) {
                     return 0;
              }
       }
       uint8_t H = Wire.read();
       uint8_t L = Wire.read();

       uint16_t result = (uint16_t)((H<<8) | L);
       result &= ~0x0003;   // clear two low bits (status bits)
       Wire.read();
       return (uint16_t)(result);
}
```

Obr. 3.7. Zdrojový kód

Zdrojový kód Obr. 3.7 z Arduino IDE, využíva veľa komplexných knižníc vďaka čomu návrh rýchli , jednoduchý a intuitívny. Avšak v prípade problému by som len ťažko hľadal principiálnu chybu v knižniciach. Našťastie všetko fungovalo. Vidíme implementáciu čítania dát zo senzora cez I2C, ďalej komunikáciu so serverom ThingSpeak a taktiež prechody do low power módov (ESP.deepSleep(x)).

4 Testovanie dosahu prijímania wifi signálu

Mojím cieľom bolo vďaka externej anténe dosiahnuť lepší dosah komunikácie wifi oproti modulu ESP-12 s internou anténou na DPS. Potreboval som totiž pripojenie na wifi na dvore, vo vzdialenosti od domu cca 30 metrov, pričom router je na druhej strane domu a v ceste sú teda 3 betónové steny. Čo už je dokopy cca 40 metrov plus tienenie stenami.

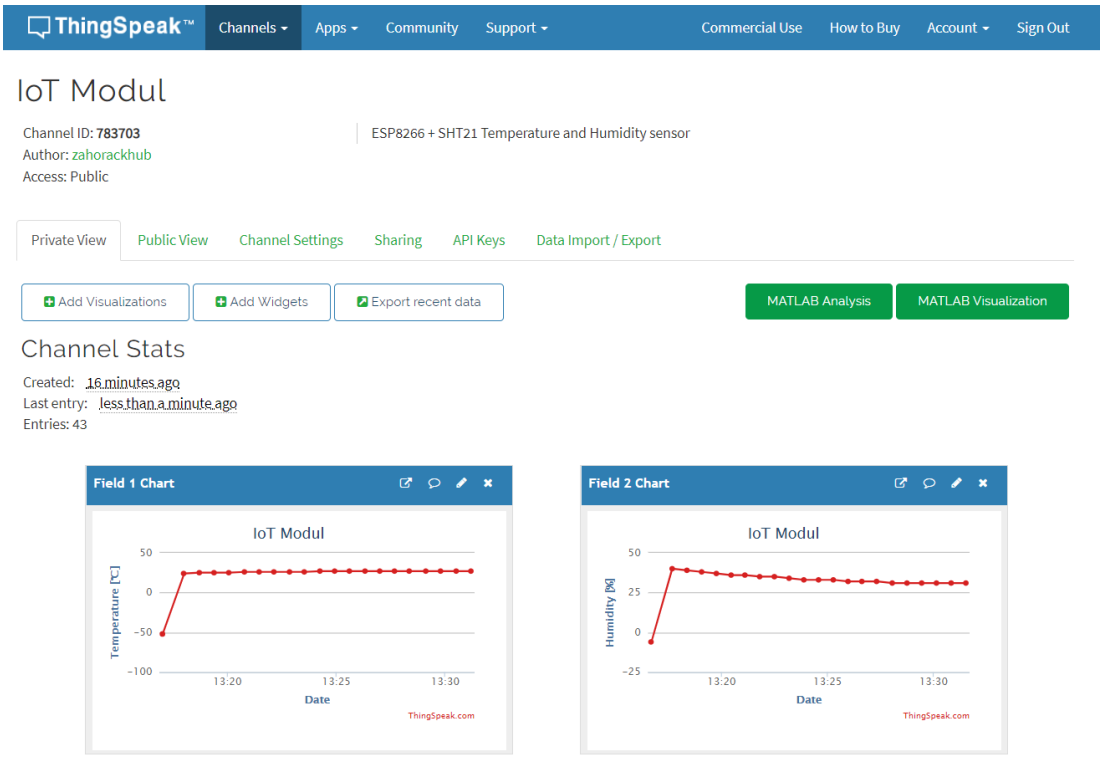
Pri pokusoch s ESP-12 som dokázal komunikovať cca 5 metrov od domu, pre porovnanie s mobilným telefónom bol výsledok podobný. Avšak s novým modulom na báze ESP-07 a externou anténou som dosiahol komunikáciu na požadovanú vzdialenosť bez problémov. Na Obr. 4.1 je fotka komunikujúceho modulu počas testovania 40 metrov od routera. Výsledkom je teda možnosť využívať IoT zariadenie a monitorovanie prostredia aj v značnej vzdialenosti od routera v exteriéry.

Obr. 4.1. Testovanie dosahu signálu Wifi modulu

5 Vzdialený prístup a vizualizácia dát

Využitie modulu môže byť napríklad zdieľanie informácií o teplote a vlhkosti z interného senzora, alebo iné informácie z pridaných senzorov, či dokonca riadenie aktuátorov cez internet.

Ako príklad využitia a vizualizácie dát som použil zdieľanie teploty a vlhkosti cez platformu ThingSpeak na Obr. 5.1. Tieto dáta v podobe grafov sú prístupne cez každý internetový prehliadač od všadiaľ. Platforma ponúka aj mobilnú aplikáciu na monitorovanie. Nevýhodu platformy pociťujem v obmedzenosti rýchlosti a objemu aktualizovania dát pri neplatenej verzii. A teda do budúcna budem hľadať inú platformu, prípadne vlastný server.



Obr. 5.1. Vizualizácia dát

6 Zhodnotenie dosiahnutých výsledkov

Výsledkom práce je funkčný univerzálny wifi modul s dlhým dosahom pripojenia Wifi. Vývojom a návrhom som sa podrobnejšie zoznámil s hardvérovou úrovňou IoT zariadení. Výsledná cena návrhu modulu prevyšuje optimálne hodnoty, preto výroba sa oplatí len pri sériových množstvách. Naprogramoval som modul tak, aby spracovával informácie zo snímačov a zdieľal ich na internetovom serveri. Pričom modul vďaka dlhému dosahu môže byť v exteriéry vzdialený od routera aj 40 metrov. Na otvorenom priestranstve (bez stien) by dokázal komunikovať na oveľa dlhšie vzdialenosti. Zmena využitia modulu je ovplyvnená len jednoduchým preprogramovaním, na čo je modul pripravený. Vďaka low power managementu je možné modul použiť aj pri batériových aplikáciách.