|  |  |  |
| --- | --- | --- |
|  | | |
| **Závěrečná studijní práce**  **dokumentace** | | |
| **GPS sledovač pro psy** | | |
| Michal Kurečka | | |
|  | | |
|  | |  |
| **Obor:** | 18-20-M/01 INFORMAČNÍ TECHNOLOGIE  se zaměřením na počítačové sítě a programování | |
| **Třída:**  **Školní rok:** | IT4  2021/2022 | |

#### Poděkování

*Jako první bych chtěl poděkovat svému pejskovi, který mě dovedl k tomuto projektu, dále bych chtěl podělkovat panu Ing. Petru Grusmannovi za pomoc s komunikační a hardwarovou částí projektu a nakonec bych chtěl poděkovat panu Mgr. Marku Lučnému, za rady v oblasti webu a webových stránek.*

Prohlašuji, že jsem závěrečnou práci vypracoval samostatně a uvedl veškeré použité   
informační zdroje.

Souhlasím, aby tato studijní práce byla použita k výukovým účelům na Střední průmyslové   
a umělecké škole v Opavě, Praskova 399/8.

V Opavě 13.12.2021

*podpis autora práce*

**ANOTACE**

Účelem tohoto GPS sledovače je pomocí zařízení připevněného na obojku získávat polohu psa a následně tuto polohu poslat na druhé zařízení (příjemce), které tyto data zpracuje a následně zobrazí na webovém serveru, tak aby uživatel jednoduše mohl k informacím přistoupit.

Projekt se skládá ze tří hlavních částí. Hardwarová stránka, kterou je esp32 zařízení, které je, jak u uživatele, tak na obojku psa. Zároveň zajištuje spolu s GPS modulem získávání, odesílání a příjem informací. Druhá část je softwarový kód, který pomocí převážně LoRa sítě zajišťuje komunikaci mezi zařízeními, spolu se získáváním informací o poloze, které je zařízeno modulem GPS. Poslední část je web, na kterém se informace zobrazují v přehledném uživatelském rozhraní.

Klíčová slova: uživatel, webový server, LoRa síť, GPS modul, uživatelské rozhraní

OBSAH

**[Úvod](#_Toc370246085)** [5](#_Toc370246085)

**[1.](#_Toc370246087)****[Využité technologie](#_Toc370246087)** [6](#_Toc370246087)

1.1 Heltec Esp32 LoRa V2 [6](#_Toc370246087)

1.2 SPIFFS  [6](#_Toc370246087)

1.3 LoRa  [6](#_Toc370246087)

1.4 JQUERY Update [7](#_Toc370246087)

1.5 Visual Studio Code  [7](#_Toc370246087)

1.6 Tinkercad  [8](#_Toc370246087)

1.7 PlatformIO  [8](#_Toc370246087)

1.8 GPS  [8](#_Toc370246087)

**[2.](#_Toc370246088)****[Způsoby řešení a použité postupy](#_Toc370246088)** [9](#_Toc370246088)

2.1 Vývojové prostředí 9

2.2 Řešení LoRa komunikace  [1](#_Toc370246087)0

2.3 Získávání GPS souřadnic  [1](#_Toc370246087)1

2.4 Spuštění SPIFFS a serveru  [7](#_Toc370246087)

2.5 Webový server [7](#_Toc370246087)

**[3.](#_Toc370246089)****[Výsledky řešení A testy](#_Toc370246089)** [9](#_Toc370246089)

3.1 Komunikace mezi Esp  [7](#_Toc370246087)

3.2 Komunikace mezi Esp  [7](#_Toc370246087)

3.3 Testy funkce a dosahu zařízení  [7](#_Toc370246087)

3.4 Porovnání s konkurencí [7](#_Toc370246087)

**[4.](#_Toc370246086)****[ČASOVÝ HARMONOGRAM](#_Toc370246086)** [6](#_Toc370246086)

**[5.](#_Toc370246089)****[uživatelský manuál](#_Toc370246089)** [9](#_Toc370246089)

**[Závěr](#_Toc370246090)** [10](#_Toc370246090)

**[Seznam INFORMAČNÍCH ZDROJů a odkazy](#_Toc370246091)** [11](#_Toc370246091)

Úvod

Nápad na tento konkrétní projekt vzniknul, když jsem byl jeden den na procházce s mým psem, který se zběhl za srnkou a já ho pak musel půl hodiny hledat. Jelikož s ním chodím na procházky často, a výskyt srnek je v mojí oblasti docela hojný, řekl jsem si, že mu udělám nějaký GPS lokátor, který by mi posílal jeho polohu, a já tak nemusel mít strach, že se zase ztratí.

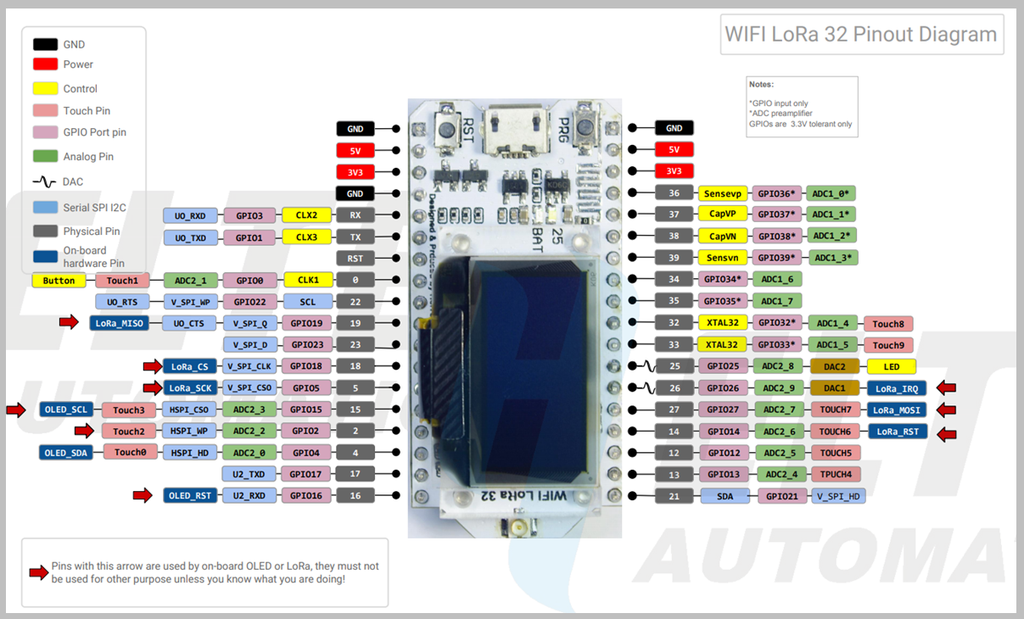
První návrh, byl ve stylu arduina s GPS modulem, který by data posílal pomocí SMS zpráv přímo do telefonu, avšak to se po pár konzultacích změnilo. Nakonec jsem si za cíl, jsem si dal vytvořit dvě zařízení, první Esp32 “Sender”, který má za úkol získávat pozici z GPS modulu, kterou následovně pomocí LoRa sítě posílat packety s informacemi o této poloze, a druhý Esp32 “Reciever”, který bude tyto packety přijímat, parsovat a následovně vypisovat na webový server, tak aby se uživateli stačilo pouze připojit na adresu, a hned by viděl kde se “sender” nachazí.

Tato dokumentace nejdříve popisuje nejdůležitější technologie, které jsem při tvorbě tohoto projektu využil, dále jak jsem postupoval a jak jsem jednotlivé technologie použil. Třetí část se skládá z finálního výsledku komunikace zařízení a porovnání s kokurencí, na konec je k dispozici časový harmonogram a uživatelský manuál.

# Využité technologie

## Heltec Esp32 LoRa

Vývojová deska Esp32 je série mikrokontrolerů, pracujících s napětím 3-7V, které společně se zabudovaným Wifi a Bluetooth modulem fungují jako malý počítač. Zvolil jsem tuto vývojovou desku, protože má v sobě společně s Wifi a Bluetooth zabudovaný také LoRa modul a 0,96-palcový OLED display.



## SPIFFS

Je souborový a webový systém, dokáže ukládat soubory do paměti Flash. SPIFFS používám, kvůli vytvoření webového serveru na Esp zařízení a dotazům, které jsou posílány z webového serveru do Esp. Jednoduše se může posílat odpovědi na dotazy z webu.

server.on("/jquery-3.6.0.min.js", HTTP\_GET, [](AsyncWebServerRequest \*request) { //cesta pro jquery

    request->send(SPIFFS, "/jquery-3.6.0.min.js", "text/javascript");

  });

  server.begin(); //zapne server

## LoRa

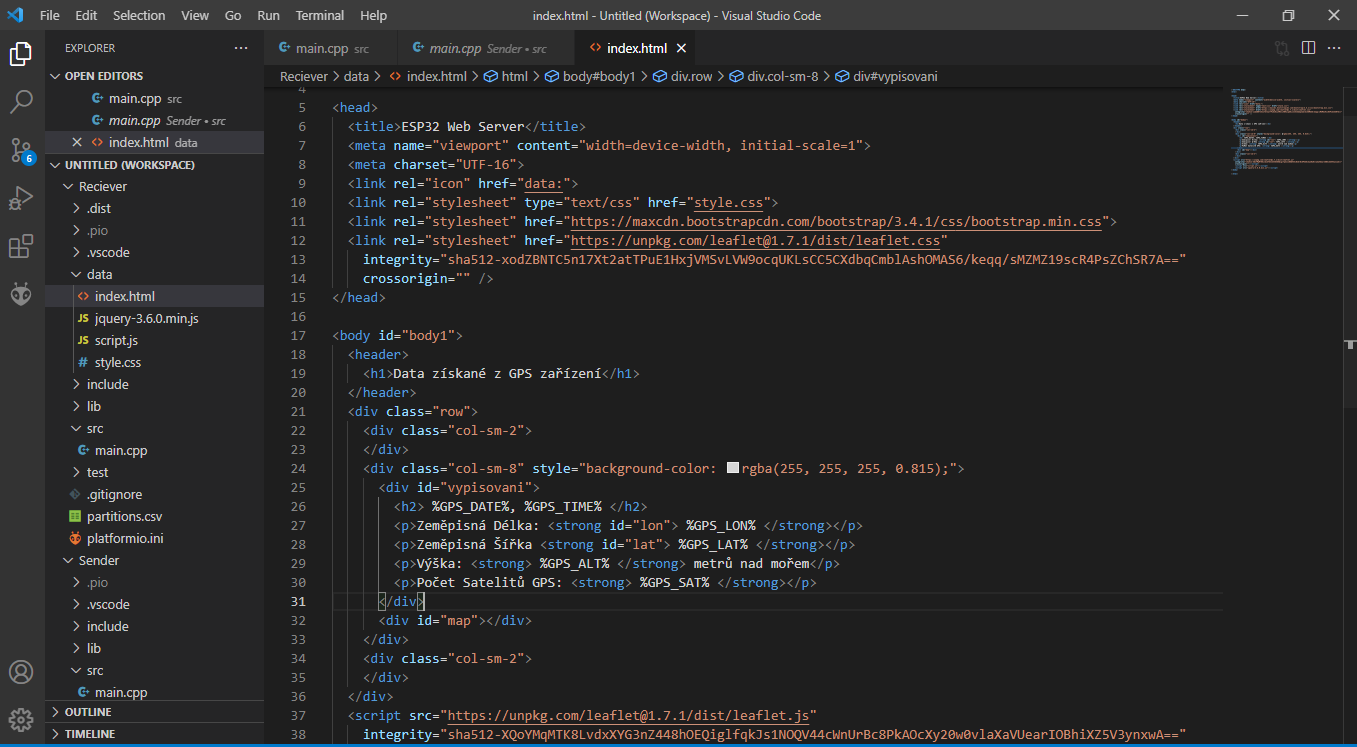
LoRa je řešení pro bezdrátový přenos dat, kdy je hlavním cílem co nejnižší spotřeba energie při malých pořizovacích nákladech a postačí malý datový tok. Vzhledem k vysílacímu výkonu v řádu jednotek až desítek miliwattů a občasnému vysílání třeba jen několikrát denně může být životnost baterií zařízení mnoho let. Díky využití tzv. Bezlicenčních pásem v rozsahu metrových, resp. decimetrových vln může být v příznivém terénu dosah přes 10 km. LoRa síť jsem zvolil z jednoduchého důvodu, potřebuju postal malé množství dat co nejefektivněji a v tom LoRa exceluje.

## JQUERY Update

JQuery je javascriptová knihovna s širkou podporou prohlížečů, která klade důraz na interakci mezi JavaScriptem a HTML. Byla vydána Johnem Resingem v lednu 2006. JQuery je open source software pod licencí MIT. JQuery jsem využil na automatické obnovování stránky, byla to nejefektivnější metoda.

## Visual Studio Code

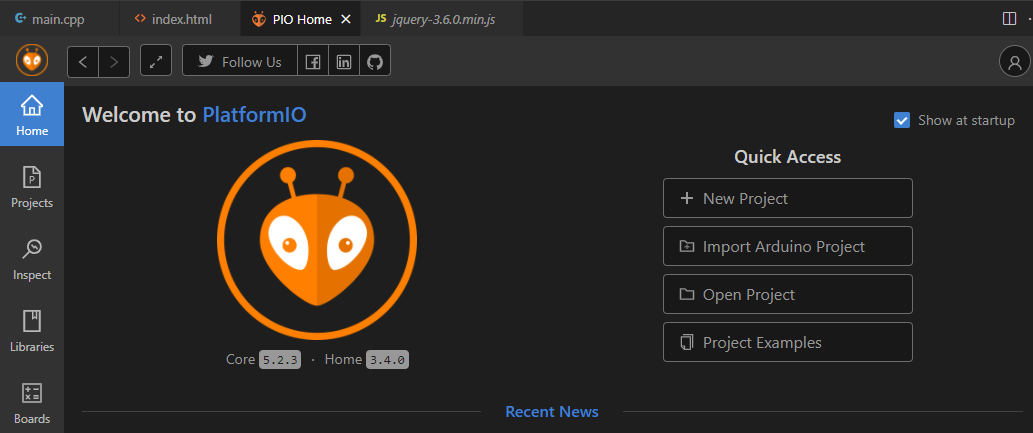
Visual Studio Code je [editor zdrojového kódu](https://cs.wikipedia.org/w/index.php?title=Editor_zdrojov%C3%A9ho_k%C3%B3du&action=edit&redlink=1" \o "Editor zdrojového kódu (stránka neexistuje)) vyvíjený společností [Microsoft](https://cs.wikipedia.org/wiki/Microsoft" \o "Microsoft) pro [operační systémy](https://cs.wikipedia.org/wiki/Opera%C4%8Dn%C3%AD_syst%C3%A9m" \o "Operační systém) [Windows](https://cs.wikipedia.org/wiki/Microsoft_Windows" \o "Microsoft Windows), [Linux](https://cs.wikipedia.org/wiki/Linux" \o "Linux) a [macOS](https://cs.wikipedia.org/wiki/MacOS" \o "MacOS). Obsahuje podporu pro [Git](https://cs.wikipedia.org/wiki/Git" \o "Git) (a pro [GitHub](https://cs.wikipedia.org/wiki/GitHub" \o "GitHub)), [zvýraznění syntaxe](https://cs.wikipedia.org/wiki/Zv%C3%BDrazn%C4%9Bn%C3%AD_syntaxe" \o "Zvýraznění syntaxe), kontextový [našeptávač](https://cs.wikipedia.org/wiki/Na%C5%A1ept%C3%A1va%C4%8D" \o "Našeptávač) a podporu pro [ladění](https://cs.wikipedia.org/wiki/Lad%C4%9Bn%C3%AD_(programov%C3%A1n%C3%AD)" \o "Ladění (programování)) a [refaktorizaci](https://cs.wikipedia.org/wiki/Refaktorizace" \o "Refaktorizace). Zdrojový kód je [svobodný software](https://cs.wikipedia.org/wiki/Svobodn%C3%BD_software" \o "Svobodný software) pod [licencí MIT](https://cs.wikipedia.org/wiki/Licence_MIT" \o "Licence MIT). Sestavené verze nabízené přímo Microsoftem jsou [freewarem](https://cs.wikipedia.org/wiki/Freeware" \o "Freeware) obsahujícím [telemetrii](https://cs.wikipedia.org/wiki/Telemetrie" \o "Telemetrie), ale existuje i komunitně sestavovaná varianta VSCodium. Editor je naprogramovaný v [JavaScriptu](https://cs.wikipedia.org/wiki/JavaScript" \o "JavaScript) a [TypeScriptu](https://cs.wikipedia.org/wiki/TypeScript" \o "). Visual Studio Code jsem zvolil, protože je jednoduché na používání a mám všechny potřebné programy na jednom místě, dále také kvůli tomu, že dokáže otevřít různé druhy souborů, takže nemusím využívat více aplikací.



## Tinkercad

## PlatformIO

Rozšíření ve Visual Studiu, funguje jako spouštěč arduino kódu, je to Open Source uživatelské rozhraní a debugger pro arduino a esp zařízení. Platformio jsem použil, kvůli výhodě zobrazení všech knihoven na jednom místě a jednoduchému ovládání.

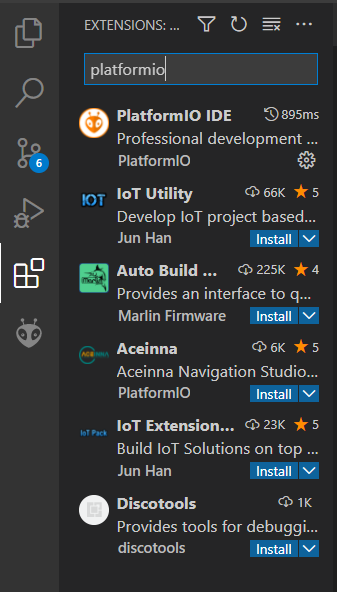


## GPS

GPS ([anglicky](https://cs.wikipedia.org/wiki/Angli%C4%8Dtina" \o "Angličtina) Global Positioning System, [česky](https://cs.wikipedia.org/wiki/%C4%8Ce%C5%A1tina" \o "Čeština) globální polohový systém, hovorově džípíeska) je [globální družicový polohový systém](https://cs.wikipedia.org/wiki/Glob%C3%A1ln%C3%AD_dru%C5%BEicov%C3%BD_polohov%C3%BD_syst%C3%A9m" \o "Globální družicový polohový systém) vlastněný [USA](https://cs.wikipedia.org/wiki/Spojen%C3%A9_st%C3%A1ty_americk%C3%A9" \o "Spojené státy americké) a provozovaný [Vesmírnými silami Spojených států amerických](https://cs.wikipedia.org/wiki/Vesm%C3%ADrn%C3%A9_s%C3%ADly_Spojen%C3%BDch_st%C3%A1t%C5%AF_americk%C3%BDch" \o "Vesmírné síly Spojených států amerických). GPS umožňuje pomocí elektronického přijímače určit přesnou polohu na povrchu země, nahrazuje tak starší metody založené na pozorování hvězd, Slunce, používání [sextantu](https://cs.wikipedia.org/wiki/Sextant" \o "Sextant) nebo [triangulace](https://cs.wikipedia.org/wiki/Triangulace" \o ").  V současné době je GPS systém využíván v mnoha oborech lidské činnosti. GPS jsem zvolil, protože je zdarma, je (skoro) všude a hlavně poskytuje velké množství informací, nejen polohu. Na komunikaci s GPS jsem zvolil modul GPS-NEO-6M, což je přídávný modul pro Esp zařízení.

# Způsoby řešení a použité postupy

## Vývojové prostředí



[*Vývojové prostředí (zkratka IDE, [anglicky](https://cs.wikipedia.org/wiki/Angli%C4%8Dtina" \o "Angličtina) Integrated Development Environment) je [software](https://cs.wikipedia.org/wiki/Software" \o "Software) usnadňující práci [programátorů](https://cs.wikipedia.org/wiki/Program%C3%A1tor" \o "Programátor), většinou zaměřené na jeden konkrétní [programovací jazyk](https://cs.wikipedia.org/wiki/Programovac%C3%AD_jazyk" \o "Programovací jazyk). Obsahuje [editor](https://cs.wikipedia.org/wiki/Textov%C3%BD_editor" \o "Textový editor) [zdrojového kódu](https://cs.wikipedia.org/wiki/Zdrojov%C3%BD_k%C3%B3d" \o "Zdrojový kód), [kompilátor](https://cs.wikipedia.org/wiki/P%C5%99eklada%C4%8D" \o "Překladač), případně [interpret](https://cs.wikipedia.org/wiki/Interpret_(software)" \o "Interpret (software)) a většinou také [debugger](https://cs.wikipedia.org/wiki/Debugger" \o "Debugger).*]Jako první věc, kterou jsem při dělání mého projektu, musel splnit, bylo vybrat si správné vývojové prostředí, ve kterém bych mohl bez problému pracovat na svém projektu. Jako první jsem chtěl využít aplikaci Arduino IDE, kterou jsem používal již nějakou dobu, avšak pro tento složitější projekt jsem nakonec zvolil Visual Studio Code s rozšířením PlatformIO, které jsem nainstaloval skrz možnost “extensions” ve Visual Studiu. První z mnoha problémů, které mě na mé cestě provázely, nastal po tom, co jsem zkusil na Esp vypsat jednoduchý text, ovšem místo textu se mi na obrazovce zobrazily jen podivné znaky. Po pár minutách jsem však zjistil, že šlo pouze o špatně nastavený počet baudů [*Baud (Bd) je [jednotka](https://cs.wikipedia.org/wiki/Fyzik%C3%A1ln%C3%AD_jednotka" \o "Fyzikální jednotka) [modulační rychlosti](https://cs.wikipedia.org/wiki/Modula%C4%8Dn%C3%AD_rychlost" \o "Modulační rychlost) (také symbolová nebo znaková rychlost, [anglicky](https://cs.wikipedia.org/wiki/Angli%C4%8Dtina" \o "Angličtina) baud rate) udávající počet změn stavu přenosového média za jednu [sekundu](https://cs.wikipedia.org/wiki/Sekunda" \o "Sekunda)*], což jsem vyřešil přepsáním informací v souboru “platformio.ini”.

[env:heltec\_wifi\_lora\_32\_V2]

platform = espressif32

board = heltec\_wifi\_lora\_32\_V2

framework = arduino

monitor\_speed = 115200 ;nastavení počtu baudů

Kromě výpisu na obrazovku, jsem také spustil program na vyhledání wifi sítí, úspěšně jsem se připojil na mou domácí síť a ujistil se, že wifi modul na zařízení funguje tak, jak má.

## Řešení LoRa komunikace

Jako další přišlo na řadu zjistit, jak funguje LoRa a jak posílat určité informace z jednoho, na druhé zařízení. Celý princip komunikace mezi zařízeními je takový, že “Sender”, neboli odesílač začne tvořit packet [*Paket označuje v [informatice](https://cs.wikipedia.org/wiki/Informatika" \o "Informatika) blok [dat](https://cs.wikipedia.org/wiki/Data" \o "Data) přenášený v [počítačových sítích](https://cs.wikipedia.org/wiki/Po%C4%8D%C3%ADta%C4%8Dov%C3%A1_s%C3%AD%C5%A5" \o "Počítačová síť) založených na [přepojování paketů](https://cs.wikipedia.org/wiki/P%C5%99epojov%C3%A1n%C3%AD_paket%C5%AF" \o "Přepojování paketů), kde je možné přenášet data i při výpadcích některých spojů.*],

do kterého uloží informace, které má obdržet druhé zařízení, ukončí tvoření packetu a dále pořád do kola vysílá dané packety po LoRa pásmě.

LoRa.beginPacket(); //začíná tvořit jeden packet

LoRa.print(gps.altitude.meters()); //uloží do packetu informaci o nadmořské výšce

LoRa.endPacket(); //ukončí a odešle packet

“Reciever” neboli přijímač má za úkol naopak čekat a skenovat dané pásmo, dokud nezachytí packet vyslaný “Senderem”. Pokud se tak stane, “Reciever” tento packet uloží a je možné s ním provádět jiné operace.

if (velikostPaketu)

    { //pokud se nejaký packet přijme spustí se funkce

      while (LoRa.available())

      { //prevedeni textu do jednoho stringu

        VypisovanyText += (VyslednyText + strlen(VyslednyText), "%c", (char)LoRa.read());

      }

Problém, se kterým jsem se v této oblasti setkal, byl ten, že Esp zařízení komunikovaly pouze, pokud se jejich antény dotýkaly, což se nakonec vyřešilo změněním LoRa pásma, které bylo pro jiný model Esp.

#define PASMO 434E6 // definuje LoRa pasmo pro Evropu

void setup()

{

Heltec.begin(false, true, true, true, PASMO); // na posledním místě se nastaví // předem zvolené pásmo “PASMO”

}

## Získávání GPS souřadnic

Po vyřešení komunikace mezi zařízeními, přišlo na řadu zajistit informace, které by se odesílaly, k tomu jsem použil GPS modul, který je schopný komunikovat se satelity. Pro jednodušší programování, jsem vybral knihovnu “TinyGps++.h”, která jak spolehlivá, tak jednoduchá na používání. Celý GPS kód funguje podobně, jako LoRa kód, zařízení skenuje prostor, ze kterého má GPS signál přijít a pokud je signál přijat, přeloží ho a dále je možné s ním pracovat.

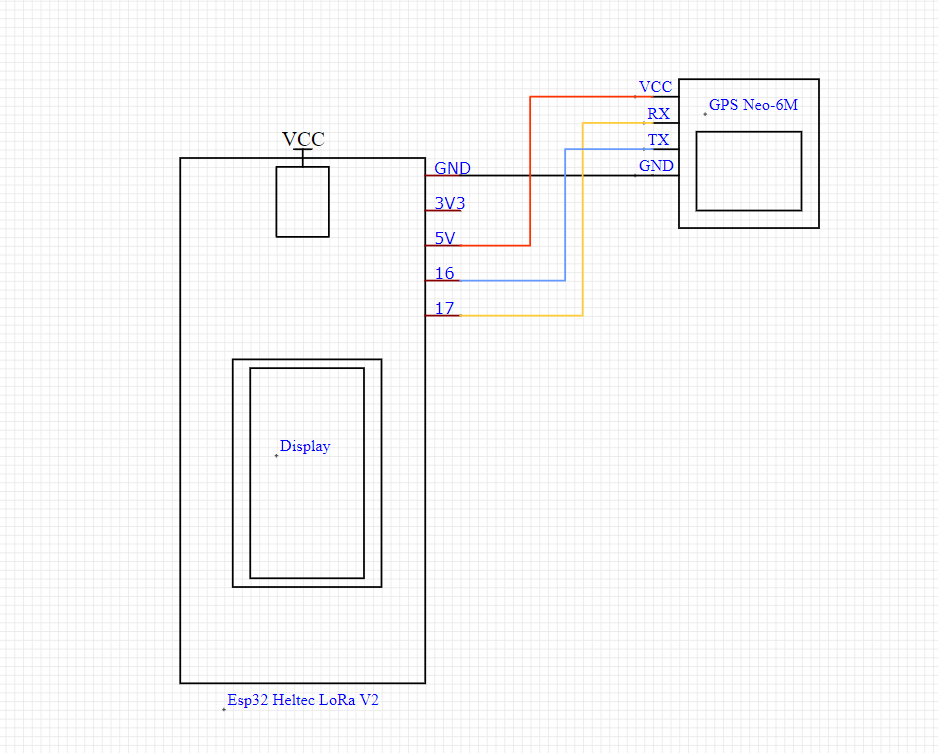
while (SerialGPS.available() > 0) //pokud je GPS signál dostupný

    {

      gps.encode(SerialGPS.read()); //přelož GPS signál

    }

GPS signál dovoluje získávát zařízení, nejen polohu a nadmořskou výšku, ale také čas, datum a počet satelitů a mnoho jiných, čehož jsem v mém projektu rád využil. Jeden ze dvou problémů, v této fázi projetku byl, že pozice, kterou jsem dostával měla často odchylku až 1km, což určitě nebylo vhodné. Tento problém byl vyřešen posláním delší informace o poloze (z 50.09 se stalo 50.099999) což silně ovlivnilo pozdější určování polohy.



Druhý problém byl, že pokaždé, když bylo například dopoledne, tak se místo času 09:53:02 zobrazovalo 9:53:2. Toto byl problém, hlavně kvůli pozdějšímu odesílání v packetu LoRy, kde to dělalo problémy. Výsledným řešením byla podmínková funkce “if”, která pokud bylo číslo menší jak 10, automaticky napsala před číslo 0.

if (gps.time.minute() < 10) //pokud je čas menší než 10

    {

      LoRa.print("0"); //napíše se 0

      LoRa.print(gps.time.minute()); //napíše se čas

    }

    else //jinak

    {

      LoRa.print(gps.time.minute()); //se vypíše celý čas

    }

## Spuštění SPIFFS a serveru

Předposledním úkolem bylo, vymylset nějaký způsob, jak by se mohl uživatel připojit na Esp zařízení a přistoupit k datům zíkaným z GPS. Proto jsem vytvořil webový server pomocí SPIFFSu. Jako první byla verze, která na samotném Esp zařízení vytvořila vlastní wifi, uživatel se připojil na a do vyhledávače zadal ip adresu [P adresa slouží k rozlišení síťových rozhraní připojených k [počítačové síti](https://cs.wikipedia.org/wiki/Po%C4%8D%C3%ADta%C4%8Dov%C3%A1_s%C3%AD%C5%A5" \o "Počítačová síť). Síťovým rozhraním může být [síťová karta](https://cs.wikipedia.org/wiki/S%C3%AD%C5%A5ov%C3%A1_karta" \o "Síťová karta) ([Ethernet](https://cs.wikipedia.org/wiki/Ethernet" \o "Ethernet), [Wi-Fi](https://cs.wikipedia.org/wiki/Wi-Fi" \o "Wi-Fi)), [IrDA](https://cs.wikipedia.org/wiki/IrDA" \o "IrDA) port, ale může se jednat i o virtuální zařízení ([loopback](https://cs.wikipedia.org/wiki/Loopback" \o "Loopback), rozhraní pro [virtuální počítač](https://cs.wikipedia.org/wiki/Virtu%C3%A1ln%C3%AD_stroj" \o "Virtuální stroj) a podobně)], která ho přesměrovala na webovou stránku. Problém ale nastal, když jsem potřeboval využít služby jako mapy, jquery nebo bootstrap, ke kterým byl nejjednodušší přístup přes internet, který Esp nemá, proto jsem se rohodnul spíše pro druhou variantu, která zahrnuje připojení se na již vyvořenou wifi, místo vytváření wifi. Nevýhoda tohoto způsobu je taková, že uživatel musí mít vytvořenou wifi s údaji zadanými na Esp, které by nebylo zrovna lehké změnit.

const char \*ssid = "Doma-wifi"; //nazev wifi, na kterou se zařízení připojí

char \*password = "heslo";  //heslo

AsyncWebServer server(80); //vytvoření serveru

void setup()

{

WiFi.begin(ssid, password); //zapnutí wifi (připojení se na wifi pomocí údajů)

  while (WiFi.status() != WL\_CONNECTED) //dokud se wifi nepřipojí

  {

    delay(1000);

    Serial.println("Connecting to WiFi.."); //vypiš connecting...

  }

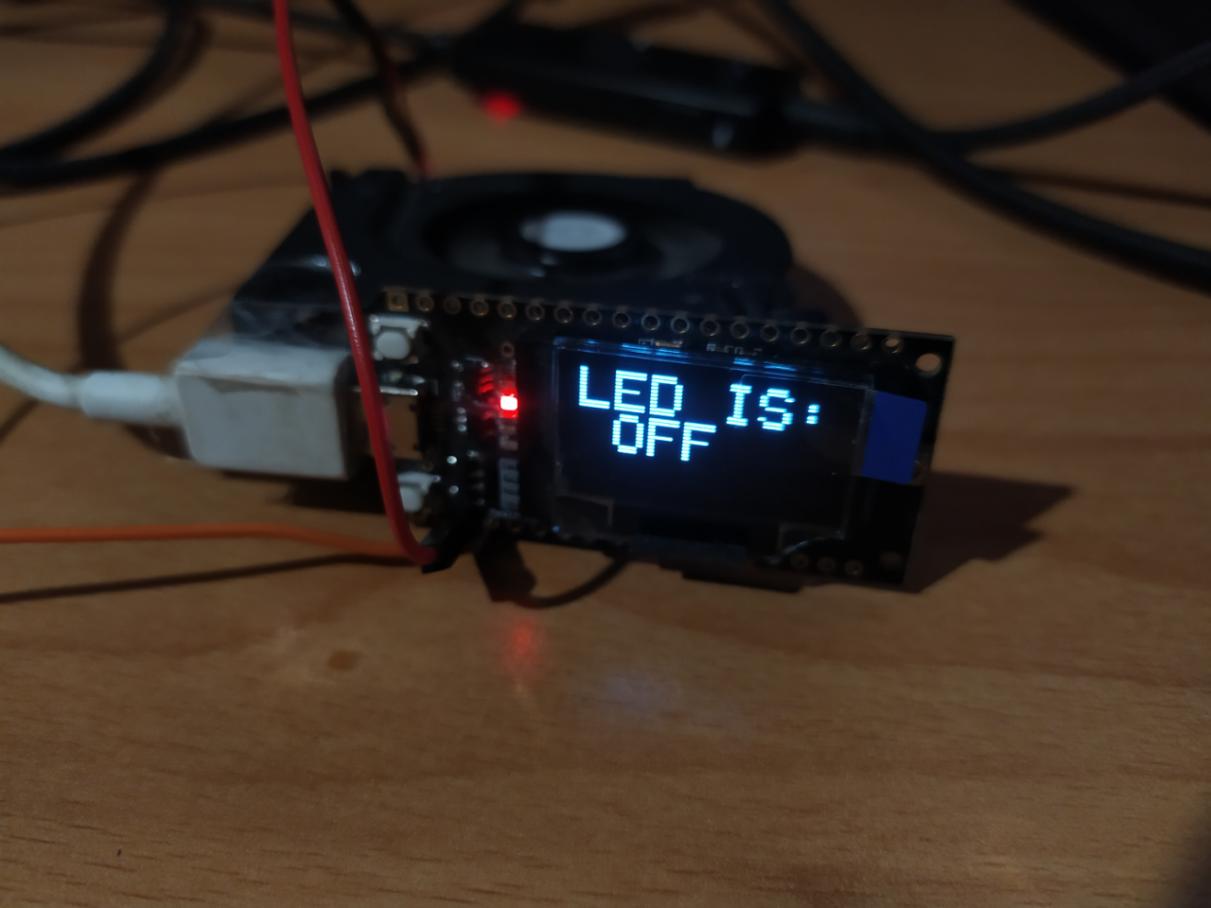
}

Ukázka využití SPIFFSu pro zaslání informací na webový server:

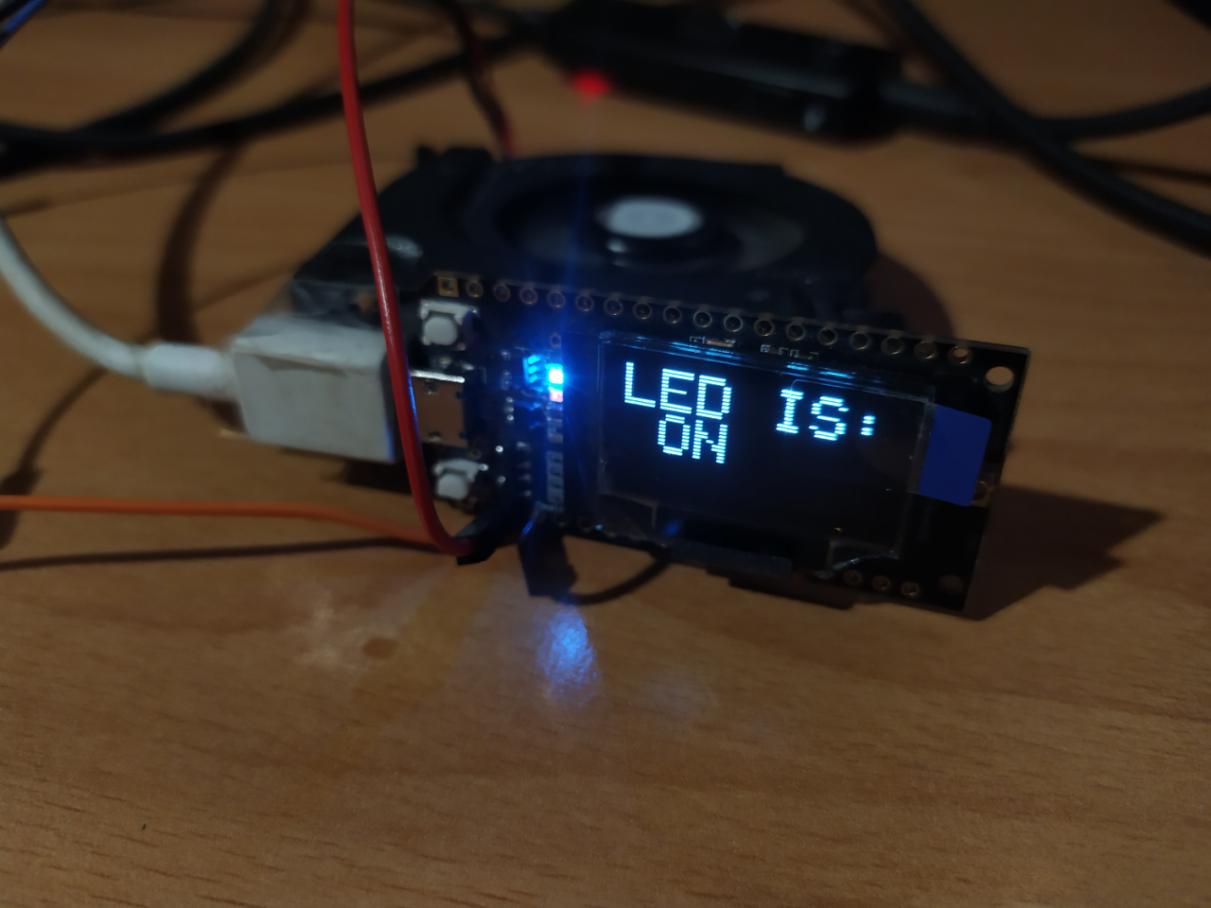
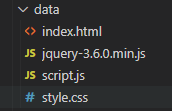
server.on("/style.css", HTTP\_GET, [](AsyncWebServerRequest \*request) { //cesta pro styly pro .css

    request->send(SPIFFS, "/style.css", "text/css");

  });



Jako první jsem vytvořil jednoduchý webový server, který po stisknutí tlačítka na webové stránce přepnul stav ledky. Účelem bylo zjistit, jak fungují jednotlivé části kódu a hlavně, jak mezi sebou Esp a Spiffs komunikují. Všechny soubory, které byly použité při tvorbě webu jsou uložené ve složce “data” a musejí se do Esp nahrát pomocí možnosti “Upload Filesystem Image”.



# Způsoby řešení a použité postupy

Text třetí kapitoly

* popis řešení úkolu včetně, použité postupy a jejich vysvětlení, způsoby testování funkčnosti, parametry výrobku (programu, hotového řešení), schémata, obrázky z tvorby a finálního provedení, výpočty, použité příkazy…

# Výsledky řešení, výstupy, uživatelský manuál

Text čtvrté kapitoly

* výčet splněných a nesplněných cílů, obrázky (schémata, vzorce apod.) z finálního provedení, prokázání funkčnosti, výsledné parametry výrobku apod.
* podle zaměření a charakteru práce je třeba volit vhodný nadpis pro tuto kapitolu, je samozřejmě možné i rozdělení na více kapitol (např. Uživatelské rozhraní internetové aplikace; Administrace internetové aplikace…)

# Závěr

Text závěru

* povinná část,
* shrnuje výsledky, hodnotí splnění cíle práce, uvádí možnost uplatnění řešení v praxi a nastínění případných dalších budoucích vylepšení
* kapitola se nečísluje (stejné jako úvod)

Seznam použitýCH INFORMAČNÍCH ZDROJů

[1] BOHMAN, Ludvík. Zákon o pojistné smlouvě. Praha: Linde Praha a. s., 2004. 381 s. ISBN80-7201-504-4

[2] DUCHÁČKOVÁ, Eva. Principy pojištění a pojišťovnictví. 3. aktualizované vydání. Praha: Ekopress 2009. 224 s. ISBN 978-80-86929-51-4

[3] KUBALA, Petr. Planetární dvojcata - Věda a technika (Český rozhlas) [online].   
Č. 2000-2008, poslední revize 19. 3. 2008 [cit. 2008-03-20].  
<http://www.rozhlas.cz/veda/vesmir/\_zprava/435849>.

[4] KULDOVÁ, O., FLEISCHMANNOVÁ, E. Metodická příručka k technice administrativy a obchodní korespondence. 1.vyd. Praha: Fortuna 1998. 111 s.   
ISBN 80-7168-574-7. Kapitola 6, Metody nácviku psaní hmatovou metodou,   
s. 28-29.

[5] VLACH, J. JE Temelín a zásobování teplem. Energetika, 2001, roč. 51, č. 3, s. 84 -85. ISSN 0375-8842.

* musí zahrnovat všechny prameny, knihy, internetové odkazy a další studijní podklady, z nichž jsme čerpali;
* kapitola se nečísluje a zde končí číslování stránek práce;
* jednotlivé publikace se uvádějí v abecedním pořadí podle příjmení autorů a iniciál jeho jména, který se píše za čárkou;
* příjmení autora se píše velkými písmeny;
* název publikace se zvýrazňuje kurzívou;
* jestliže jsou uvedeni více než tři autoři, je možné vypsat hlavního autora s poznámkou „a kol.“(a kolektiv).

Seznam příloh

č. 1 Titulní list

č. 2 Čestné prohlášení

č. 3 Poděkování

Nepovinná část – pokud nemáte žádné přílohy ke své práci, tuto část odstraňte!

* Přílohy se zařazují na konec práce.
* Jsou to texty, obrázky, grafy, tabulky, které by přímo v textu byly zbytečně detailní, ale mají být po ruce k dokreslení východisek i výsledku řešení.
* Jsou číslovány a v textu se na ně může odkazovat.
* Před první přílohu se umisťuje seznam příloh.
* Každá příloha je označena číslem - např. Tabulka č.. 1, Schéma č. 2, Obrázek č. 3.
* Každá tabulka by měla mít i vlastní název, který stručně vystihuje její obsah.
* (Tabulka č. 1 Zakázky stavebních prací v roce 2009-2010).
* Pokud je z tabulky vytvořen graf, umístíme jej na stejné stránce jako tabulku.

**Příloha č. 1: Titulní list**