|  |  |  |
| --- | --- | --- |
|  | | |
| **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 také 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 31.12.2021

*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)****[Seznámení s projektem](#_Toc370246087)** [6](#_Toc370246087)

1.1 Cíl projektu [6](#_Toc370246087)

1.2 Harmonogram projektu [6](#_Toc370246087)

**[2.](#_Toc370246087)****[Využité technologie](#_Toc370246087)**7

2.1 Heltec Esp32 LoRa V27

2.2 SPIFFS 7

2.3 LoRa 7

2.4 PlatformIO [7](#_Toc370246087)

2.5 Vývojové prostředí  [8](#_Toc370246087)

2.6 JQuery Update  [8](#_Toc370246087)

2.7 Tinkercad 9

2.8 GPS 9

**[3.](#_Toc370246088)****[Způsoby řešení a použité postupy](#_Toc370246088)**10

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

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

3.3 Spuštění SPIFFS a serveru  [1](#_Toc370246087)2

3.4 Design webového serveru  [1](#_Toc370246087)3

**[4.](#_Toc370246089)****[Výsledky řešení A testy](#_Toc370246089)** [1](#_Toc370246089)6

4.1 Shrnutí funkcí zařízení  [1](#_Toc370246087)6

4.2 Testy funkcí a dosahu zařízení  [1](#_Toc370246087)6

4.3 Porovnání s konkurencí [1](#_Toc370246087)6

**[Závěr](#_Toc370246090)**17

**[Seznam INFORMAČNÍCH ZDROJů a odkazy](#_Toc370246091)**18

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.

Jako první vás seznámím s tím, co jsem dělal a jak jsem to dělal, tak abyste dále mohli lépe porozumět principům projektu a použitým technologiím. Dále v dokumentaci popisuji významné technologie, které jsem při tvoření projektu potřeboval a co dané technologie dělají v praxi. Po seznámení s cílem a technologiemi projektu vám popíšu, jak jsem se s jednotlivými částmi projektu popral, a popřípadě na jaké problémy jsem narazil, tak abyste popřípadě neopakovali stejné chyby jako já. Před koncem najdete úsek, ve kterém se věnuji tomu, jak projekt dohromady funguje, popíšu jednotlivé testy, které jsem prováděl a porovnám moji GPS s konkurencí. Na konci najdete už jen uživatelský manuál k projektu společně se závěrem a odkazy, na všechny zdroje, které jsem použil.

# Seznámení s projektem

## Cíl projektu

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 dal, vytvořit dvě zařízení, první Esp32 “Sender”, který má za úkol získávat pozici z GPS modulu (toto zařízení je na psím obojku a sleduje, kde se pes nachází), kterou následovně pomocí LoRa sítě ve formě packetů s informacemi posílá, na druhé Esp32 “Reciever”, který tyto packety přijímá, parsuje a následovně vypisuje na display, který je součástí tohoto Esp a na webový server, tak aby se uživateli stačilo pouze připojit na adresu, a hned by viděl kde se “Sender” (zařízení na obojku psa) nachází.

## Harmonogram projektu

**- Září:** Seznámení se s technologiemi, objednání součástek

**- 1. polovina Říjnu:** První testy webového serveru, první testy s GPS modulem

**- 2. polovina Říjnu:** Řešení komunikace pomocí LoRa sítě

**- 1. polovina Listopadu:** Získávání GPS souřadnic a jejich posílání

**- 2. polovina Listopadu:** Spuštění webového serveru, a úpravy kódu

**- 1. polovina Prosince:** Práce na webové stránce a dokončení projektu

**- 2. polovina Prosince:** Dokumentace a vytváření obalu pro zařízení

# Využité technologie

## Heltec Esp32 LoRa V2

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.

## LoRa

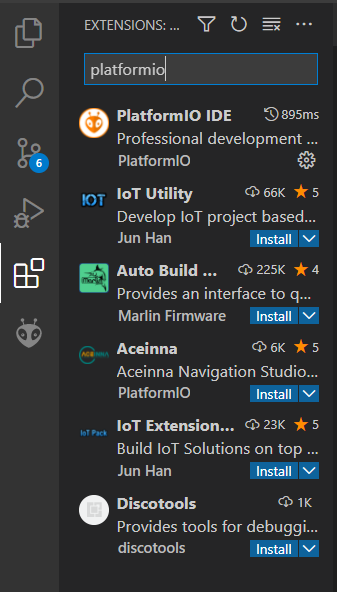
LoRa je řešení pro bezdrátový přenos dat, kdy je hlavním cílem co nejnižší spotřeba energie při malých pořizovacích nákladech a postačí malý datový tok. Vzhledem k vysílacímu výkonu v řádu jednotek až desítek miliwattů a občasnému vysílání třeba jen několikrát denně může být životnost baterií zařízení 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 poslat malé množství dat co nejefektivněji a v tom LoRa exceluje.

## 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.

## 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 baudu

## Ukázka kódu ze souboru “/Reciever/platformio.ini” (nastavení počtu baudů pro Esp)

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.

## JQuery Update

JQuery je javascriptová knihovna s širokou podporou prohlížečů, která klade důraz na interakci mezi JavaScriptem a HTML. JQuery je open source software. JQuery jsem využil na automatické obnovování stránky, byla to nejefektivnější metoda.

## Tinkercad

Tinkercad je 3D modelovací program, který je dostupný ve webovém prohlížeči. Je to často používaný nástroj, pro vytváření modelů pro 3D tisk. Tento program jsem využil proto, protože jsem potřeboval vymodelovat nějakou “krabičku”, do které bych mohl dát své esp zařízení. V této aplikaci modeluji už nějakou dobu, takže se v ní vyznám a umím ji používat

## 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).  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 přídavný modul pro Esp zařízení.

# Způsoby řešení a použité postupy

## 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, který pak bude odeslán pomocí LoRy

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

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

## Ukázka kódu ze souboru “/Sender/src/main.cpp” (vytváření packetu pro odeslání LoRa)

“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());

      }

    }

## Ukázka kódu ze souboru “/Reciever/src/main.cpp” (přijímání packetu LoRa)

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 //LoRa pasmo pro Evropu

void setup()

{

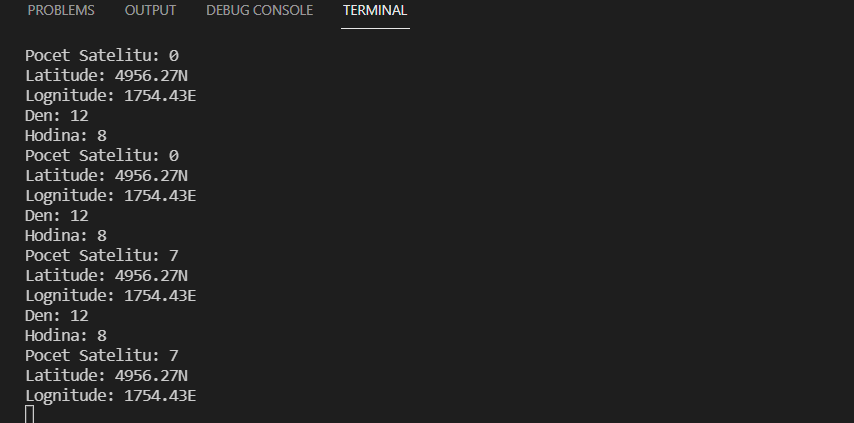
  Heltec.begin(false, true, true, true, PASMO /\*pasmo pro Evropu\*/); //nastaveni hardwaru

}

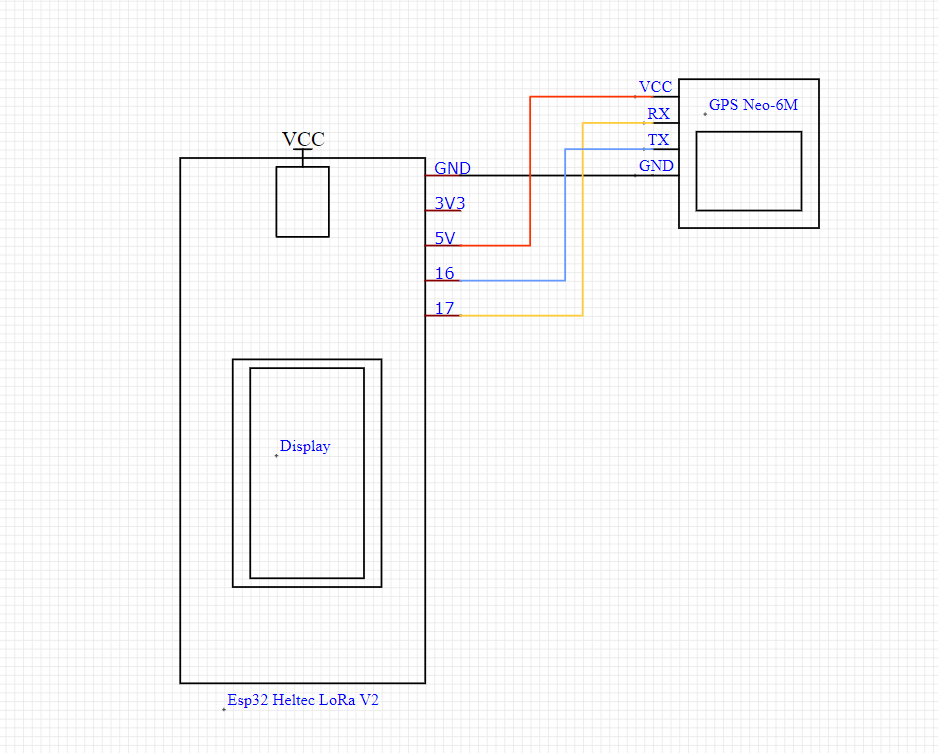
Ukázka kódu ze sozboru “/Reciever/src/main.cpp” (vybrání pásma pro LoRa)

## 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.



GPS signál dovoluje získávat 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 projektu 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.

Schéma zapojení GPS modulu

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

    {

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

    }

  }

## Ukázka kódu ze souboru “/Sender/src/main.cpp” (získávání GPS souřadnic)

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

      LoRa.print(":");

    }

    else //jinak

    {

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

      LoRa.print(":");

    }

## Ukázka kódu ze souboru “/Sender/src/main.cpp” (podmínka pro odesílání času LoRa)

## Spuštění SPIFFS a serveru

Předposledním úkolem bylo, vymyslet nějaký způsob, jak by se mohl uživatel připojit na Esp zařízení a přistoupit k datům získaným z GPS. Proto jsem vytvořil webový server pomocí SPIFFSu. Jako první byla verze, která na samotném Esp zařízení vytvořila vlastní wifi, uživatel se na ní připojil a do vyhledávače zadal IP adresu [IP 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 ], která ho přesměrovala na webovou stránku.

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

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

  });

## Ukázka kódu ze souboru “/Reciever/src/main.cpp” (odesílání dat na webserver)

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 vytvořenou wifi. Nevýhoda tohoto způsobu je taková, že uživatel musí mít vytvořenou wifi s údaji zadanými na Esp.

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

char \*password = "heslo";  //heslo

AsyncWebServer server(80); //vytvoření serveru

void setup()

{

  Heltec.begin(false, true, true, true, PASMO /\*nastaveni pasma\*/); //nastaveni hardwaru

  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...

  }

}

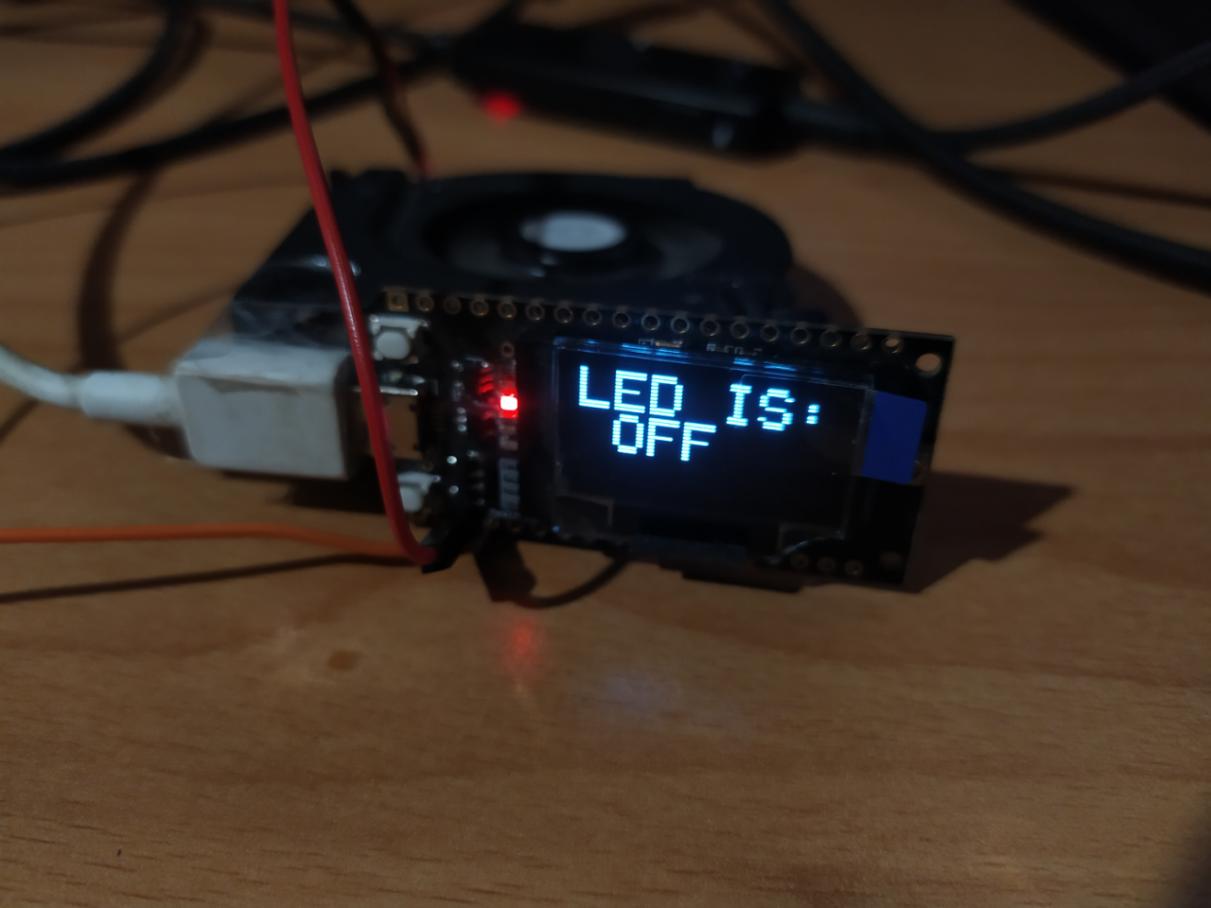
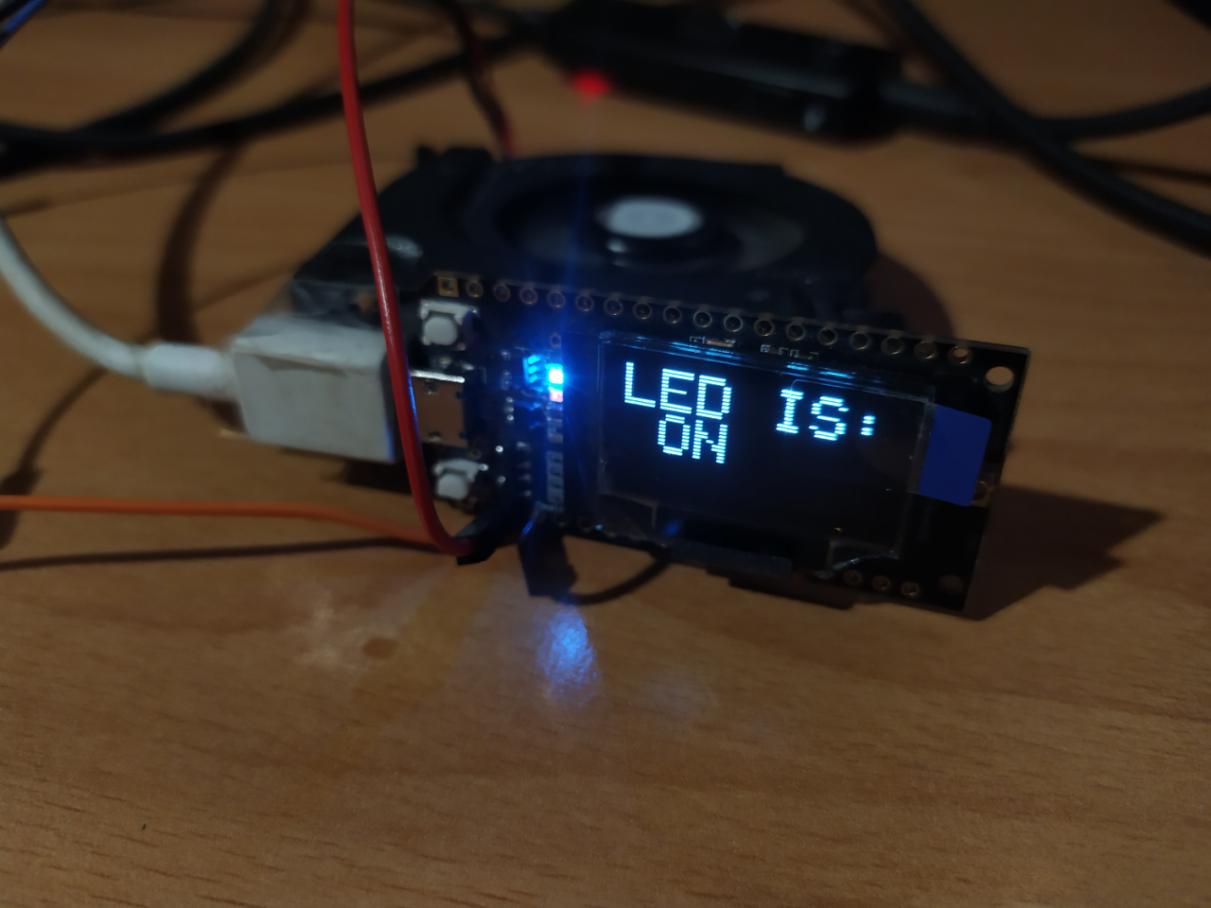
## Ukázka kódu ze souboru “/Reciever/src/main.cpp” (zapínání webserveru a připojení na wifi)

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”.

## Design webového serveru

Poslední část projetu, byl design webu tak, aby byl jednoduchý na pochopení a využití v praxi. Základem je “index.html”, který slouží jako zobrazovaná stránka a společně s “style.css” tvoří to, co uživatel vidí. Jako další je javascript jako “script.js“ sloužící převážně k funkčnosti stránky (například vytváří mapu a automaticky obnovuje informace na stránce).

Úplně první designovou část jsem vytvořil už, při spuštění SPIFFS webserveru, kde však na stránce byly pouze dvě tlačítka, která vypínala a zapínala ledku, stránka také postrádala většinu CSS [*Kaskádové styly (v anglickém originále Cascading Style Sheets se zkratkou CSS) je v informatice jazyk pro popis způsobu zobrazení elementů na stránkách napsaných v jazycích [HTML](https://cs.wikipedia.org/wiki/HyperText_Markup_Language" \o "HyperText Markup Language), [XHTML](https://cs.wikipedia.org/wiki/Extensible_HyperText_Markup_Language" \o "Extensible HyperText Markup Language) nebo [XML](https://cs.wikipedia.org/wiki/Extensible_Markup_Language" \o "Extensible Markup Language).*] a JavaScriptu [*JavaScript, často zkráceně JS, je programovací jazyk, který je vedle HTML a CSS jednou ze základních technologií World Wide Webu*]. Po dokončení funkční části projektu, však přišel čas na dotázání se Esp na informace z gps, které se poté vypsaly na stránku.



<div id="vypisovani">

        <h2> %GPS\_DATE%, %GPS\_TIME% </h2>

        <p>Zeměpisná Délka: <strong id="lon"> %GPS\_LON% </strong></p>

        <p>Zeměpisná Šířka <strong id="lat"> %GPS\_LAT% </strong></p>

        <p>Výška: <strong> %GPS\_ALT% </strong> metrů nad mořem</p>

        <p>Počet Satelitů GPS: <strong> %GPS\_SAT% </strong></p>

      </div>

## Ukázka kódu ze souboru “/Reciever/data/index.html” (výpis dat na webserveru)

Dále jsem musel implementovat do kódu základní zdroje jako Bootstrap, CSS a JavaScript, pro správné fungování stránky. V souboru “style.css” jsem pomocí tříd nadesignoval stránku, tak, že jsem změnil pozadí, velikost, či font textu. Poté jsem musel vyřešit to, že se stránka nechtěla sama obnovovat, což bylo klíčové pro správné zobrazování polohy druhého zařízení, takže jsem pomocí JavaScriptu a JQuery napsal kód, který se za určitý čas znovu zeptal Esp na informace o poloze, které se znovu zaslaly.

function updateDiv() { // funkce pro obnovování textu a mapy

  lon = document.getElementById('lon').innerText;

  lat = document.getElementById('lat').innerText;

  map.removeLayer(marker);

  marker = L.marker([lat, lon]).addTo(map);

  $("#vypisovani").load(window.location.href + " #vypisovani");

}

setInterval(updateDiv(), 1000); //funkce se spustí každých 1000 milisekund

## Ukázka kódu ze souboru “/Reciever/data/script.js” (obnovování textu a mapy)

Na konec mi zbylo, ještě nějak přijaté informace zobrazit, tak aby je nemusel uživatel manuálně zadávat do Map (př. Mapy Google), toto byla pravděpodobně nejtěžší část designu. Zaprvé jsem implementoval zdroje pro knihovnu “leaflet”, která dokázala vložit na stánku mapy a ukázat na nich polohu, která se dynamicky vkládá pomocí funkce, která získá informaci o poloze z textu, který je vypsaný na html stránce, za druhé jsem musel zajistit způsob, kterým by se mapy jako takové zobrazovaly n a stránce, s tím mi pomohl “mapbox”, který na mapu pokládá jednotlivé “dlaždice” mapy (mapa je rozdělena do jednotlivých dlaždic, které se pak pokládají na mapu podle potřeby a velikosti zvolené oblasti) a tím vyplní určenou oblast.

// Vytvoření mapy

let lon = document.getElementById('lon').innerText; //získá souřadnice, které jsou již na stránce vypsané

let lat = document.getElementById('lat').innerText;

let map = L.map('map').setView([lat, lon], 12);

L.tileLayer('https://api.mapbox.com/styles/v1/{id}/tiles/{z}/{x}/{y}?access\_token=pk.eyJ1IjoibWNrdWt1cmljZWN6IiwiYSI6ImNreGJ3NDR1ejBwdjEyeHBnMmthamhsajgifQ.GAvd9JUztKQTiwEl5V2JIA', {

  attribution: 'Map data &copy; <a href="https://www.openstreetmap.org/copyright">OpenStreetMap</a> contributors, Imagery © <a href="https://www.mapbox.com/">Mapbox</a>',

  maxZoom: 18,

  id: 'mapbox/streets-v11',

  tileSize: 512,

  zoomOffset: -1,

}).addTo(map);

let marker = L.marker([lat, lon]).addTo(map);

## Ukázka kódu ze souboru “/Reciever/data/script.js” (funkce pro vytvoření mapy a ukazatele polohy)

# Výsledky řešení a testy

## Shrnutí funkcí zařízení

První zařízení Sender, dokáže získat informace z GPS satelitů a pomocí LoRa sítě je sdílí v předem určeném pásmu jako proud dat. Druhé zařízení na tyto data čeká, a pokud přijme některé z těchto dat, rozloží je na jednotlivé části (poloha, čas, počet satelitů,...) a pokud se na ně webový server zeptá, tak mu je pošle. Webový server přijme data z Esp, zobrazí je na webové stránce a podle pozice, kterou dostal vytvoří interaktivní mapu, po které se může uživatel volně posouvat, přibližovat a oddalovat. Tato stránka se každou sekundu obnoví a tím se obnoví i dostupné informace.

## Testy funkcí a dosahu zařízení

1. První test se konal v lese, terén byl převážně zarostlý křovím a stromy. LoRa zařízení se nedokázalo připojit k “Sender” zařízení, test byl **neúspěšný**.
2. Druhý test, se konal na okraji lesu, terén byl převážně prostupný, občasné stromy, musíme brát v potaz, že signál musel nejdříve projít přes vesnici, což znamená domy a různé typy elektromagnetického rušení. Test byl úspěšný s dosahem asi *800-900 metrů*.
3. Třetí test, se konal v obývané oblasti, v okolí byly budovy a ostatní zařízení, terén byl smíšený. Přesto byl test úspěšný s dosahem asi *1300 metrů*.

## Porovnání s konkurencí

1. Test dosahu - konkurenční zařízení byly lepší, s průměrným dosahem 15km, mé zařízení bylo schopné dosáhnout 1300 metrů v terénu.
2. Test zobrazení - konkurenčrí zařízení mají stejně jako já možnost obrazit pozici psa na mapě. Některé zařízení mají navíc i kompas, který ukazuje směrem k pozici.
3. Cena - v této části vyhrává moje zařízení s cenou (bez práce) necelých 1,000 korun, oproti konkurenčním zařízením, kde cena začíná okolo 10,000 korun.

Celkový výsledek: V porovnání cenové hranice s možnostmi a dosahem zařízení, bych usoudil, že za necelých 1000 korun je mé zařízení dobrou volbou.

# Závěr

Cílem projektu bylo, vytvořit malý GPS sledovač, který by získával informace o poloze zařízení a odesílal je na druhé zařízení. Toto zařízení by data zpracovalo a předalo uživateli v podobě webové stránky.

V průběhu práce mě napadlo spoustu vylepšení, které by se v budoucnu daly uplatnit, aby se i mé zařízení mohlo rovnat konkurenci na trhu, například kompas ukazující směr k zařízení, lepší zobrazení stránky nebo i vylepšení hardwaru, pro lepší dosah signálu.

Projekt v tomto stavu, umožňuje tyto informace získávat a také předávat uživateli v podobě, jakou jsem zamýšlel, navíc jsem ještě přidal možnost zobrazení na mapě, kterou beru jako velké vylepšení původního nápadu, a tím bych řekl, že cíl projektu byl jak naplněn, tak i překonán.

Hlavní však je, množství nových informací a zkušeností, které jsem za dobu projektu získal, a které můžu v budoucnu využít k dalším podobným projektům. Celý projekt bych za sebe hodnotil velice kladně a jsem rád, že jsem si vybral zrovna tento projekt. Dále se budu snažit projekt využít při mých pravidelných procházkách se psem.

K možnosti uplatnění projektu, bych navrhnul ještě chvíli počkat, projekt není perfektní a má stále nějaké chyby, avšak v budoucnu, kdyby byl plně hotový společně s opravenými chybami, bych viděl uplatnění, jak v sektoru sledovaní mazlíčků, tak například sledování vozidel či zavazadel.

Seznam INFORMAČNÍCH ZDROJů a odkazy

**Odkaz na Github s projektem:**

**https://github.com/NeViMjAkSeJmEnOvAt/rocnikovy\_projekt\_V2**

**Použité zařízení:**

Heltec Esp32 LoRa V2 - **<https://www.ebay.com/itm/173624285281>**

GPS modul - **https://www.ebay.com/itm/174930619300**

1. NMEA - https://en.wikipedia.org/wiki/NMEA\_2000
2. spiffs - https://randomnerdtutorials.com/esp32-vs-code-platformio-spiffs/
3. spiffs webserver - https://randomnerdtutorials.com/esp32-web-server-spiffs-spi-flash-file-system/
4. LoRa komunikace - https://github.com/HelTecAutomation/Heltec\_ESP32/tree/master/examples/LoRa
5. LoRa komunikace 2 - https://randomnerdtutorials.com/esp32-lora-rfm95-transceiver-arduino-ide/
6. LoRa prikazy - https://github.com/sandeepmistry/arduino-LoRa/blob/master/API.md
7. Heltec knihovna-priklady - https://github.com/HelTecAutomation/Heltec\_ESP32/tree/master/examples/LoRa
8. GPS data - https://ww2.mathworks.cn/help/supportpkg/arduino/ref/read-serial-data-from-a-gps-shield-using-arduino-hardware.html //nakonec použitý jiný program
9. GPS - https://github.com/Heltec-Aaron-Lee/WiFi\_Kit\_series/issues/116 //nakonec použitý jiný program
10. GPS2 - https://learn.adafruit.com/adafruit-ultimate-gps/arduino-wiring //nakonec použitý jiný program
11. GPS3 - https://randomnerdtutorials.com/guide-to-neo-6m-gps-module-with-arduino/ //nakonec použitý jiný program
12. parsovani dat GPS - https://learn.adafruit.com/adafruit-ultimate-gps/parsed-data-output
13. Tiny GPS kód - https://github.com/DzikuVx/esp32\_gps\_thingy/blob/master/gps\_logger.ino
14. rtos vtasks - https://www.youtube.com/watch?v=95yUbClyf3E
15. arduino wifi - https://randomnerdtutorials.com/esp32-access-point-ap-web-server/
16. esp delay nahrada - https://www.arduino.cc/en/Tutorial/BuiltInExamples/BlinkWithoutDelay
17. nastaveni wifi - https://gist.github.com/futechiot/ee0223dd269cbe7d8605ce97d120d7d2
18. ESP rest api endpoints - https://www.mischianti.org/2020/05/16/how-to-create-a-rest-server-on-esp8266-and-esp32-startup-part-1/
19. jquery help - https://javarevisited.blogspot.com/2016/04/3-ways-to-solve-jquery-uncaught-reference-error-is-not-defined.html
20. jquery update - https://stackoverflow.com/questions/33801650/how-do-i-refresh-a-div-content
21. rest api - https://github.com/superucitelka/SmartWatch-REST
22. partitions - https://github.com/espressif/arduino-esp32/tree/master/tools/partitions
23. Partitions2 - https://docs.platformio.org/en/latest/platforms/espressif32.html#partition-tables
24. mapa1 - https://stackoverflow.com/questions/9912145/leaflet-how-to-find-existing-markers-and-delete-markers
25. mapy2 - https://www.mapbox.com
26. leaflet - https://leafletjs.com/examples/quick-start/
27. leaflet2 - https://stackoverflow.com/questions/43743097/showing-an-offline-osm-map-file-suggestion-an-mb-tiles-file-with-js-library
28. size of partitions - <https://stackoverflow.com/questions/36413423/mapbox-offline-storage-size>
29. Images - https://pixabay.com