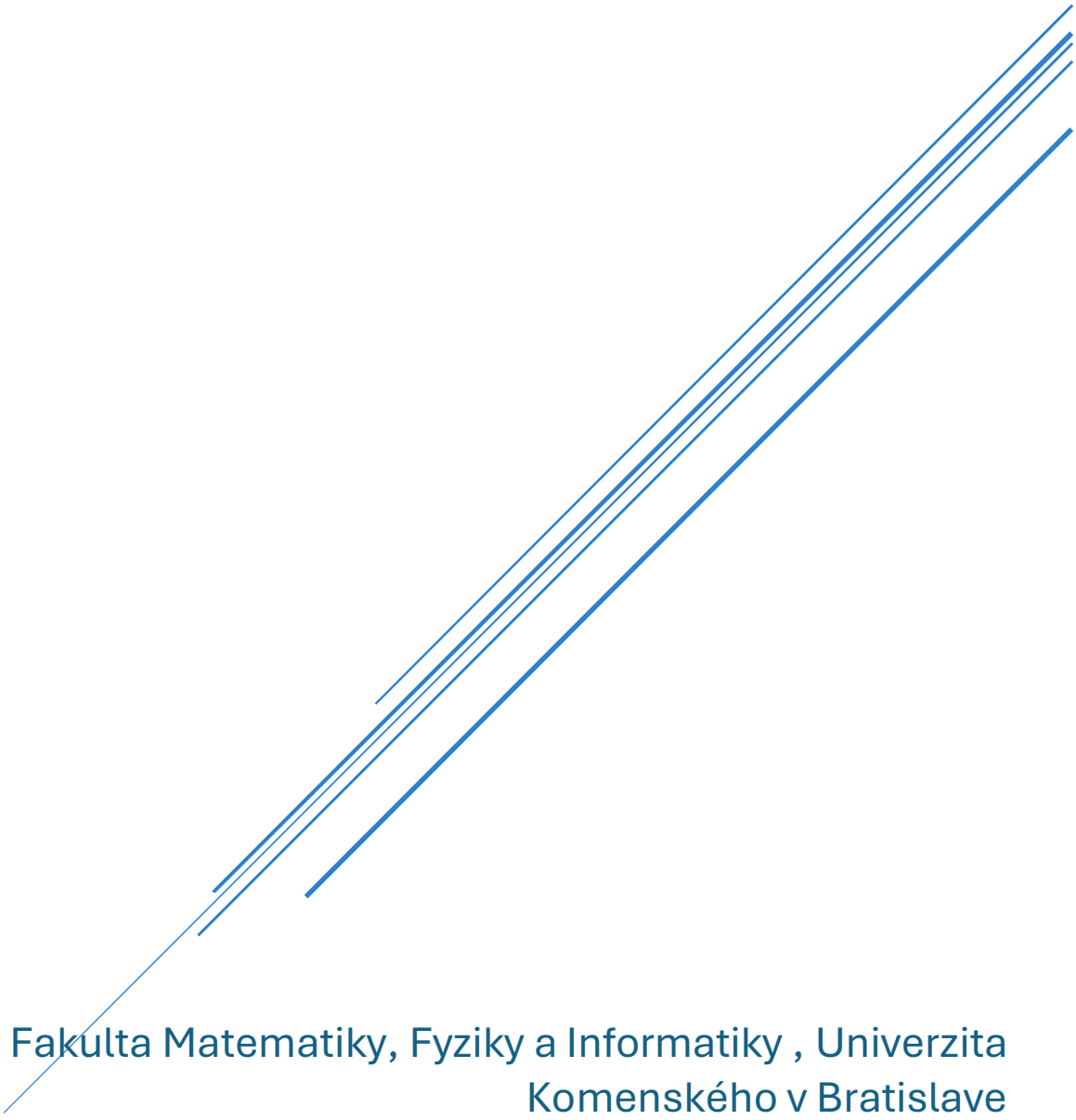
MONITOROVANIE OKAMŽITEJ POLOHY ĎALEKOHĽADU

Mikuláš Zelenák, Ivana Zeleňáková, Ján Radovan Zubo, Mariia Lementa



Fakulta Matematiky, Fyziky a Informatiky , Univerzita Komenského v Bratislave
Katedra aplikovanej informatiky

Obsah

1. Úvod

1.1 Účel dokumentu

Tento dokument popisuje presné požiadavky na systém kontroly polohy teleskopu. Systém je vyvíjaný pre Astronomické a geofyzikálne observatórium (ďalej iba AGO) Modra-Piesok Fakulty matematiky, fyziky a informatiky Univerzity Komenského v Bratislave (ďalej iba FMFI UK). Tento dokument je záväznou dohodou medzi zadávateľom a riešiteľmi projektu.

1.2 Rozsah produktu

Cieľom projektu je vyvinúť systém na monitorovanie a zabezpečenie polohy teleskopu v AGO Modra-Piesok, ktorý bude fungovať nezávisle od existujúceho systému. Systém bude pomocou snímačov sledovať polohu teleskopu a zabezpečí, aby sa nedostal do nebezpečných polôh. Pri prekročení stanovených limitov automaticky spustí varovné signály/alarm.

1.3 Slovník pojmov

SW – software
HW – hardware
Riadiaca jednotka – mikropočítač, ktorý rádiovo komunikuje s jednočipovým mikropočítačom namontovanom na teleskope a snímajúcom IMU, má jednoduchý displej a riadiaci panel a zároveň komunikuje s PC buď cez sériovú linku, alebo ethernet
Relé – súčiastka, ktorá v jednom obvode zapne alebo preruší elektrický prúd v druhom elektrickom obvode
Tubus – trubica ďalekohľadu alebo mikroskopu
Akcelerometer – súčiastka, ktorá meria lineárne zrýchlenie vrátane tiažového
Gyroskop – zariadenie na meranie uhlového zrýchlenia
IMU – inerciálna jednotka, ktorá kombinuje snímanie z akcelerometra, gyroskopu a prípadne kompasu a vypočítava relatívnu orientáciu voči Zemi a iniciálnej polohe
Eulerove uhly – všeobecný pojem pre možné druhy reprezentácie otočenia v 3D priestore - sú buď “intrinsic” - osi sa otáčajú spolu s objektom v každom kroku, alebo “extrinsic” - osi sa neotáčajú, zostávajú stále rovnaké (world coordinate system)
Roll-Pitch-Yaw – konkrétny prípad Eulerovských uhlov v danom poradí (Roll – os v smere pohybu lietadla, Pitch - vodorovná os kolmá na smer pohybu lietadla, Yaw - zvislá os)
Gimbal lock – strata jedného stupňa voľnosti v niektorých uhlových súradnicových systémoch pri zarovnaní dvoch osí otáčania
Kvaterniony - najvšeobecnejšia reprezentácia otočenia v 3D priestore, ktorá netrpí gimbal lockom vo forme $a + b*i + c*j + d*k$

1.4 Odkazy

Github repozitár projektu: https://github.com/TIS2024-FMFI/telescope-safety
Inerciálna jednotka ICM20948, https://www.waveshare.com/wiki/10_DOF_IMU_Sensor_(D)
Inerciálna jednotka s MPU9255 a BMP280,
https://www.waveshare.com/wiki/10_DOF_IMU_Sensor_(C)
Inerciálna jednotka ISM330DHCX, https://www.elecom.sk/sparkfun-6dof-imu-breakout-ism330dhcx--qwiic--2/
Inerciálna jednotka s ICM20948, https://www.elecom.sk/sparkfun-9dof-imu-breakout-icm-20948--qwiic--2/
Inerciálna jednotka s MPU6050, https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf
Inerciálna jednotka DFfRobot Fermion s Adxl345, ITG3200 a HMC5883L,
https://techfun.sk/produkt/dfrobot-fermion-10-dof-imu-senzor/
Interciálna jednotka s BNO055, https://www.aliexpress.com/item/1005005494956365.html
Frekvencie používané na rádiové spojenie pre IoT a ich vlastnosti, https://www.data-alliance.net/blog/iot-internet-of-things-wireless-protocols-and-their-frequency-bands/
Obojsmerný komunikačný rádiový modem 868 MHz LoRa,
https://techfun.sk/produkt/komunikacny-modul-sx1276-lora-433-868-915-mhz/?attribute_pa_variant=868mhz-rfm95w
Modul s relé, https://techfun.sk/produkt/rele-modul-1-kanal-5v/

1.5 Prehľad nasledujúcich kapitol

Kapitola číslo 2 opisuje vyvíjaný systém. Popisuje jeho funkcionalitu, všeobecné vlastnosti a správanie.
Kapitola číslo 3 obsahuje podrobné požiadavky, ktoré boli stanovené zadávateľom.

2. Všeobecný opis

2.1 Perspektíva

Systém bude slúžiť na kontrolu a bezpečnosť teleskopu. Na základe jeho aktuálnych priestorových a fyzikálnych parametrov bude vyhodnocovať, či by bol ďalší jeho pohyb bezpečný a nehrozí riziko havarijného stavu. Tento systém bude úplne nezávislý od softvérových programov, siete a napájania, čo zabezpečí jeho nepretržitú prevádzku aj v prípade výpadkov týchto zdrojov. V prípade vyhodnotenia rizika havárie systém podnikne kroky k jej zabráneniu.

2.2 Funkcionalita

Základom funkčnosti systému je monitorovanie polohy tubusu ďalekohľadu pomocou snímačov (inerciálna jednotka), ktoré sú prepojené s riadiacou jednotkou prostredníctvom rádiovej komunikácie. Systém nielenže sleduje polohu, ale umožňuje aj automatické vypnutie pohonu ďalekohľadu pri prekročení zakázaných polôh a zobrazuje výstražné signály (akustické, vizuálne) pre operátora. Medzi kľúčové vlastnosti patrí možnosť konfigurácie zakázaných polôh, ako aj pripojenie systému k lokálnej sieti, čím sa zabezpečí vzdialený prístup k údajom o polohe a konfigurácii systému. Tieto funkcie sú navrhnuté s dôrazom na spoľahlivosť a možnosť ďalšej rozšíriteľnosti systému. Systém bude monitorovať polohu a pohyb teleskopu. Bude neustále sledovať a zbierať dáta ohľadom aktuálnej polohy tubusu ďalekohľadu, jeho orientácie a parametrov ako sú rýchlosť otáčania či sklon, azimut, aby sa v prípade potreby včas zistili nežiadúce potenciálne odchýlky od očakávaných/povolených hodnôt. Systém bude analyzovať monitorované dáta a umožní porovnávať aktuálne údaje s predošlými a predpokladanými, ak tie budú k dispozícii a detegovať odchýlky na základe tejto analýzy. Systém sa bude dať nastaviť tak, aby na potenciálne riziko reagoval niektorým z nasledujúcich spôsobov (alebo ich kombináciou): akustickým signálom, vizuálnym signálom, výstupom na rozpínací kontakt relé - vypnutie napájania motorov ďalekohľadu, notifikáciou e-mailovou správou kompetentným. Po signalizovaní rizika je systém možné znovu aktivovať a obnoviť prerušené napájanie, či na mieste, alebo softvérovo na diaľku. Systém bude vytvárať podrobné záznamy o pohyboch teleskopu a bude sa snažiť priebežne diagnostikovať svoj vlastný stav, aby sa zabezpečila kontrola funkčnosti samotného systému - včasné odhaľovanie softvérových a hardvérových problémov, ktoré by mohli viesť k opačnému efektu systému. Systém bude vedieť zobraziť vizualizáciu aktuálnej polohy teleskopu v 3D zobrazení, v ktorom je možné pohybovať virtuálnou kamerou, t.j zobrazovať teleskop z ľubovoľnej strany (bude realizované iba v prípade dostatku ľudských zdrojov a času).

2.3 Používateľské roly

Používatelia budú astronomickí pracovníci a odborníci, obsluhujúci ďalekohľad. Môžu to byť astronómovia, astrofyzici alebo technici so špecializáciou na prácu s pokročilými optickými a elektronickými systémami. Okrem vedeckých pracovníkov budú súčasťou používateľov aj technickí operátori a údržbári zodpovední za správnu funkčnosť a kalibráciu prístrojov.

Používatelia sa budú líšiť v úrovni skúseností a znalostí softvérových nástrojov, pričom je dôležité zabezpečiť prístupné rozhranie, ktoré zjednoduší ich prácu a podporí efektívnu spoluprácu medzi

vedeckými tímami. Očakáva sa, že väčšina používateľov bude pracovať s dátami v reálnom čase, analyzovať ich a optimalizovať pozorovacie metódy v závislosti od aktuálnych potrieb výskumu.

Okrem iného to budú aj študenti, ktorí budú ovládať menší ďalekohľad, kde budú využívať dáta systému.

2.4 Všeobecné obmedzenia

Kvôli obmedzenej dostupnosti ďalekohľadu sa predpokladá, že sa na testovanie využije model, ktorý si tím zostrojí.

Potenciálnemu vzájomnému vlnovému narúšaniu sa medzi rádiovou komunikáciou a inými zdrojmi vlnenia (wifi, senzory pohybu,...) z dôvodu podobných vlnových frekvencií sa zamedzí využitím odlišných frekvencií (868 MHz), ale aj tak je potrebné skontrolovať, či prenos signálu je v danom priestore spoľahlivý.

Exaktnosť gyroskopov a akcelerometrov – je tu možnosť že sa nebudú kontrolovať len odchýlky ktoré má systém riešiť, ale aj odchýlky zariadení ktoré tieto odchýlky zisťujú – potrebná kvalitná kalibrácia systémových zariadení ako aj ich správny výber.

2.5 Predpoklady a závislosti

Predpokladá sa, že vývojový tím sa hladko adaptuje na všetky výzvy, hardvérové a softvérové prostriedky. Predpokladá sa, že zariadenia budú napájané cez powerbanku, ktorá bude fungovať ako záložný zdroj v prípade výpadku napájania a nepretržite bude pripojená na externý zdroj. Softvér bude navrhnutý tak, aby bol jednoducho modifikovateľný pre pripojenie viac ako jednej inerciálnej jednotky súčasne.

3. Konkrétne požiadavky

3.1 Požiadavky na funkcionalitu

1. Systém musí fungovať nezávisle. Nezávisle na elektrickej sieti, internetovej sieti, internetovej infraštruktúre, existujúcich programoch.
2. Systém bude tvorený nasledujúcimi komponentami: 1) modulom s inerciálnou jednotkou; 2) riadiacou jednotkou; 3) softvérom na PC.
3. Modul s inerciálnou jednotkou bude primontovaný priamo na teleskope a bude pozostávať z inerciálnej jednotky, jednoduchého jednočipového mikropočítača a rádiového modemu, ktorý tento modul bude spájať s riadiacou jednotkou.
4. K riadiacej jednotke bude pripojený jednoduchý displej, jednoduchý riadiaci panel (tlačidlá), vizuálny a zvukový alarm, HW rozhranie pre odpojenie napájania (relé) a bude ju možné pripojiť cez ethernet na softvér na PC.
5. Softvér na PC dokáže vizualizovať údaje prenášané z modulu z inerciálnej jednotky v reálnom čase a konfigurovať celý systém.
6. Systém (modul s inerciálnou jednotkou) bude neustále snímať okamžitú polohu teleskopu v priestore.
7. Po vstúpení teleskopu do súradníc, ktoré sú predkonfigurované ako “nebezpečné“, systém spustí konfigurovateľný alarm.
8. Spustenie alarmu môže (podľa jeho konfigurácie) spôsobiť nasledujúce akcie (alebo ich kombináciu):
    a. Odpojenie systému ovládania polohy ďalekohľadu od elektrickej siete, pre zabránenie zrážke ďalekohľadu s inou časťou prostredia
    b. Audiovizuálnu signalizáciu problému
9. Používateľ môže nakonfigurovať súradnice pre spustenie alarmu z administratívneho rozhrania softvéru na PC prostredníctvom nahratia konfiguračného súboru, alebo ručného zadania súradníc.
10. Súradnice konkrétnej polohy (jedného bodu) teleskopu sa vyjadrujú dvojicou azimut (0° – 360°), elevácia *(-90° – 90°)* (sklon, výška).
11. Hodnota súradnice azimut sa dá nastaviť ručne cez displej na riadiacej jednotke.
12. Jedna zakázaná oblasť je definovaná ako postupnosť bodov b_i=(azimut_i, elevácia_i), ktoré tvoria uzavretý “polygón” na guľovej ploche v smere hodinových ručičiek pri pohľade zo stredu guľovej plochy.
13. Súradnice, ktoré sú predkonfigurované ako nebezpečné (pozri 9) tvorí zoznam zakázaných oblastí (pozri 14).
14. Hodnoty azimutu a elevácie, predkonfigurované nebezpečné súradnice a akákoľvek iná konfigurácia sa bude ukladať do permanentnej pamäte riadiacej jednotky.
15. Používateľské rozhranie pre zobrazenie okamžitých súradníc teleskopu zobrazuje okamžitý azimut a eleváciu tubusu a to aj na riadiacej jednotke aj v softvéri na PC.
16. Systém bude pracovať aj keď softvér na PC nie je práve k riadiacej jednotke pripojený.
17. Riadiaca jednotka a modul s inerciálnou jednotkou budú pracovať ďalej aj pri výpadku/odpojení od siete s rozvodom napätia.

18. Systém bude v nejakom dobre definovanom softvérovom rozhraní poskytovať informácie o polohe teleskopu pre ďalší externý softvér, ktorý ich bude môcť napr. porovnávať s aktuálnou želanou polohou teleskopu v riadiacom softvéri.
19. Systém bude zaznamenávať aktuálnu polohu teleskopu a všetky mimoriadne i konfiguračné udalosti do záznamu s frekvenciou, ktorú bude možné nejakým spôsobom nastaviť.
20. Softvér na PC umožní získať súbor so záznamami (pozri 21) do súborového systému na PC.
21. Softvér automaticky zabezpečí, aby bol z hľadiska miesta na svojom lokálnom úložisku bezúdržbový, t.j. staré záznamy sa môžu automaticky vymazať.

3.2 Doplnkové požiadavky*

1. Systém bude vedieť zobrazovať/zaznamenávať aktuálnu polohu v rôznych súradnicových systémoch: Eulerove uhly, Roll-Pitch-Yaw, kvaterniony.
2. Zobrazenie polohy teleskopu v softvéri na PC bude aj grafickej podobe v 3D zobrazení, ktoré sa dá otáčať posúvaním/otáčaním virtuálnej kamery.

* budú realizované iba v prípade dostatku ľudských zdrojov

Návrh

Úvod

Tento dokument popisuje návrh pre vývoj systému na monitorovanie a kontrolu polohy teleskopu pre Astronomické a geofyzikálne observatórium Modra-Piesok. Kľúčovými prvkami systému sú modul s inerciálnou jednotkou, riadiaca jednotka a webové rozhranie, ktoré umožňuje vzdialený prístup k údajom a konfigurácii. Tento návrh dokumentuje komunikáciu medzi jednotlivými komponentmi hardvéru a softvéru, špecifikáciu funkcií jednotlivých modulov, podrobnosti o logovaní a správe dát, návrh používateľského rozhrania a plán implementácie.

Podrobná špecifikácia vonkajších rozhraní systému na kontrolu polohy teleskopu:

Definícia komunikácie medzi jednotlivými komponentmi hardvéru (merač/vysielač a prijímač/zobrazovač):

V moduli s inerciálnou jednotkou bude komunikácia medzi senzorom a mikropočítačom prebiehať pomocou I^2C rozhrania. Modul s inerciálnou jednotkou bude vybavený rádiovým modemom, ktorý zabezpečuje reálnu komunikáciu so systémom riadiacej jednotky. Na komunikáciu mikropočítača s modemom sa využije SPI - Serial Peripheral Interface. Modem bude poskytovať bezdrátovú komunikáciu medzi modulom s inerciálnou jednotkou a riadiacou jednotkou pomocou protokolu LoRa.

Riadiaca jednotka (prijímač/zobrazovač): Má schopnosť prijímať údaje o polohe teleskopu z modulu s inerciálnou jednotkou prostredníctvom rádiového signálu. Obsahuje jednoduchý displej na zobrazenie aktuálnych súradníc azimutu a elevácia. Riadiaca jednotka vie komunikovať s počítačom cez ethernetové rozhranie (Pomocou protokolov HTTP a WebSocket).

Riadiaca jednotka, po prijatí dát o polohe ďalekohľadu z inerciálnej jednotky rádiovým signálom, zaloguje prijaté dáta do súboru (pre každý deň je vytvorený nový súbor, po určitom čase sa logy automaticky mažú) a odošle dáta vo formáte JSON web klientom pripojeným na WebSocket (na WebSocket sa pripoja pomocou nacitania webu pomocou HTTP a o WS spojenie sa uz postara JS).

Riadiaca jednotka vyhodnocuje prijaté data (zisťuje, či sa náhodou teleskop nedostal do niektorej zo zakázaných zón). Pokiaľ sa dostal teleskop do zakázanej zóny spustí fyzický alarm

(audiovizuálny) a zaloguje spustenie alarmu, aktuálnu polohu, pri ktorej došlo ku kolízii (spusteniu alarmu) a dátum a čas (do špeciálneho logu kolízii, ktoré sa nemažú).

Prehliadač po prijatí JSON súboru pomocou JavaScript-u JSON rozbalí a zobrazí (optionaly: vykreslí ďalekohľad a jeho aktuálnu polohu).

Používatelia sa cez prehliadač pripoja na web server cez HTTP[S], obdržia od servera HTML, CSS a JS súbory, následne JavaScript na stránke čaká na prijatie dát cez WebSocket.

Návrh jednotlivých modulov

Modul s inerciálnou jednotkou (MIMU):

Inerciálna jednotka umiestnená na teleskope bude monitorovať jeho polohu a pohyb pomocou komponentov Roll-Pitch-Yaw, Eulerových uhlov alebo kvaterniónov. Vstavaný mikropočítač spracováva tieto informácie, preráta ich do uhlov azimut a elevácia. Tieto údaje posiela riadiacej jednotke. V prípade potreby zo strany používateľa dostáva správu od riadiacej jednotky, že modul treba resetnúť a nanovo nakalibrovať.

Riadiaca jednotka (RJ):

Riadiaca jednotka má ručne vytvorenú MAC adreru, ktora spĺňa špecifikáciu pre takto generovane MAC adresy. Od DHCP servera nasledne získavá IP adresu. K servru sa klienti budu pripajat cez lokalnu adresu something.local (názov domény bude upresnený neskôr) ktorú bude server distribuovať pomocou mDNS protokolu (alternatívne by tento server updatoval svoju lokalnu IP na nejakej dostupnej stránke (najlepšie lokálnej pre menšiu závislosť na sieti), ktorú užívateľ zadá a automaticky ho presmeruje na lokálnu IP RJ).

Perzistentné údaje (RJ)

Všeobecné nastavenia:

RJ si pamätá nastavene frekvencie logovania do súboru a frekvencie zasielania Azimutu a Elevácie prihláseným klientom cez WS.

Tiež si pamätá nastavenia o tom, aké alarmy má spúšťať.

Log súbory:

Všetku logové súbory sú vo formáte CSV

Priečinok "Positions" obsahuje súbory s logmi. Súbory sa volajú "Log_" + dátum vzniku súboru, t.j. prvého záznamu v ňom. Každý záznam obsahuje timestamp (dátum a čas), polohu Azimut a Elevácia.

Súbor "Log_collisions" obsahuje informáciu o vstupe teleskopu do zakázaných zón. Záznam obsahuje timestamp, polohu do ktorej vošiel a informáciu o tom, či odpojilo motory (či bolo zapnuté odpájanie motorov).

Súbor "Log_configuration" obsahuje informáciu o každej zmene konfigurácie. Záznam obsahuje timestamp, typ zmeny konfigurácie, IP zariadenia, ktoré menilo toto nastavenie.

Konfiguračné súbory pre zakázané zóny:

Formát súboru:

```
1. line: Azimuth Elevation
2. line: Azimuth Elevation
3. line: Azimuth Elevation
...
n. line: \n (empty line)
n+1. line: Azimuth Elevation
...
```

* Azimut a Elevácia sú oddelené medzerou

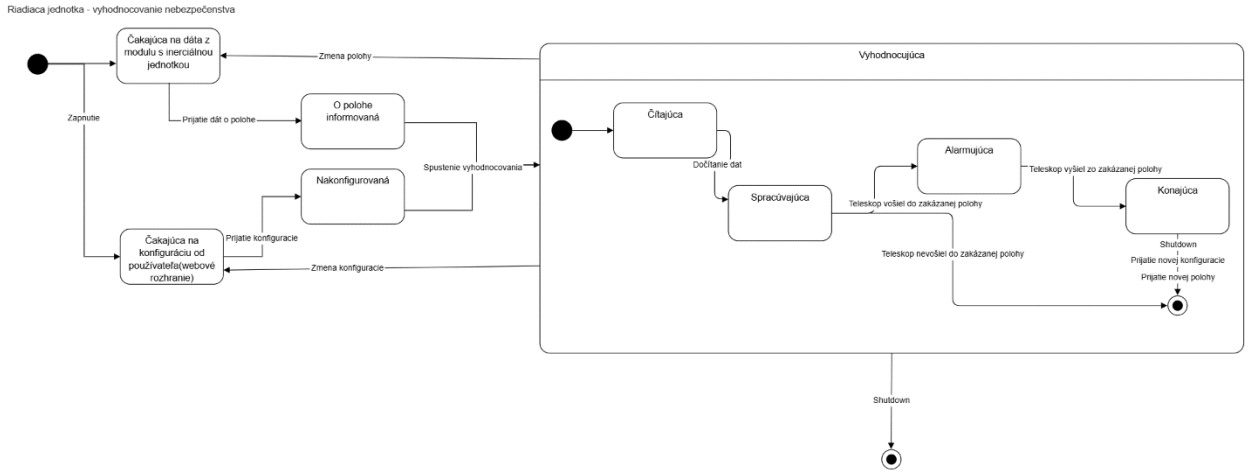
* Každá dvojica Azimut a Elevácia sú v inom riadku

* Rôzne zakázané zóny sú oddelene jednym prázdnym riadkom

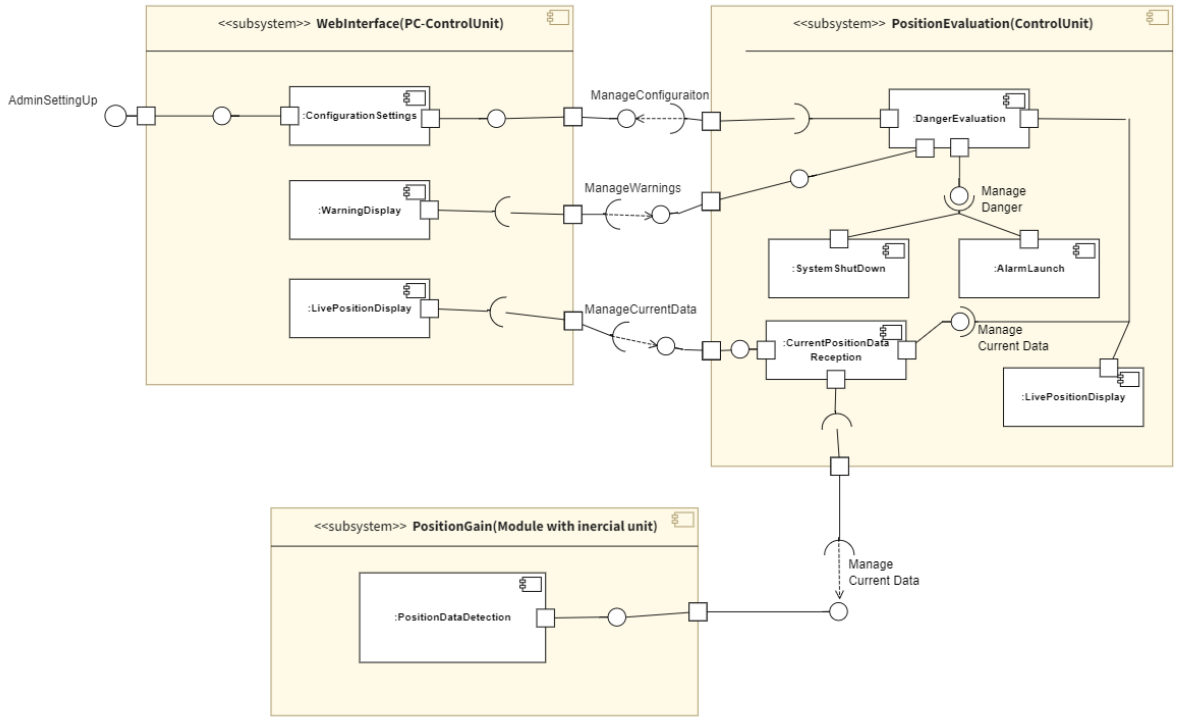
* Každá zakázaná zóna obdahuje aspon 3 body (3 dvojice Azimut a Elevácia)

* Riadky začínajúce znakom ‘#’ sú ignorované, slúžia na pipis zóny pre uživateľa+

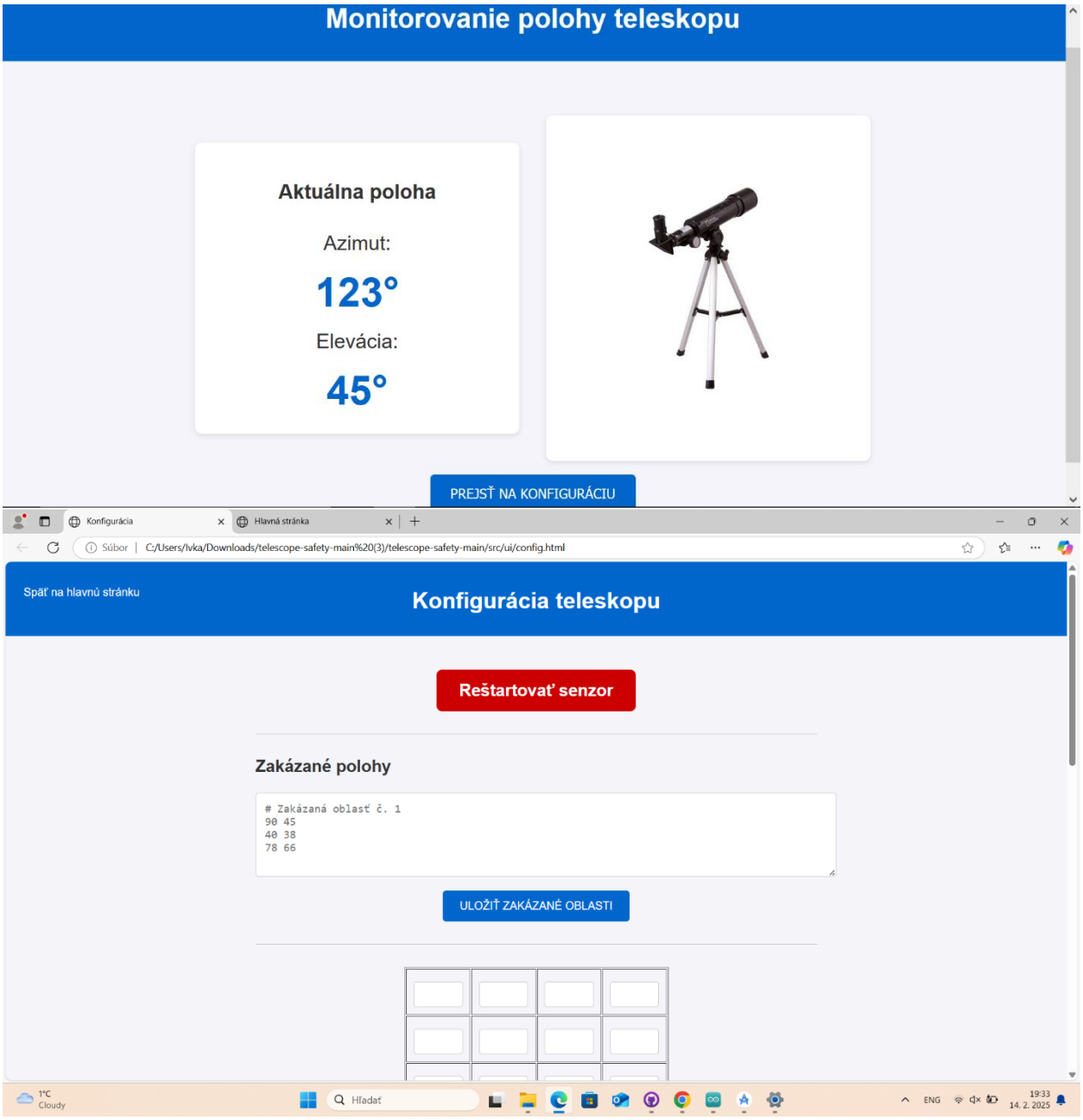
Diagramy:

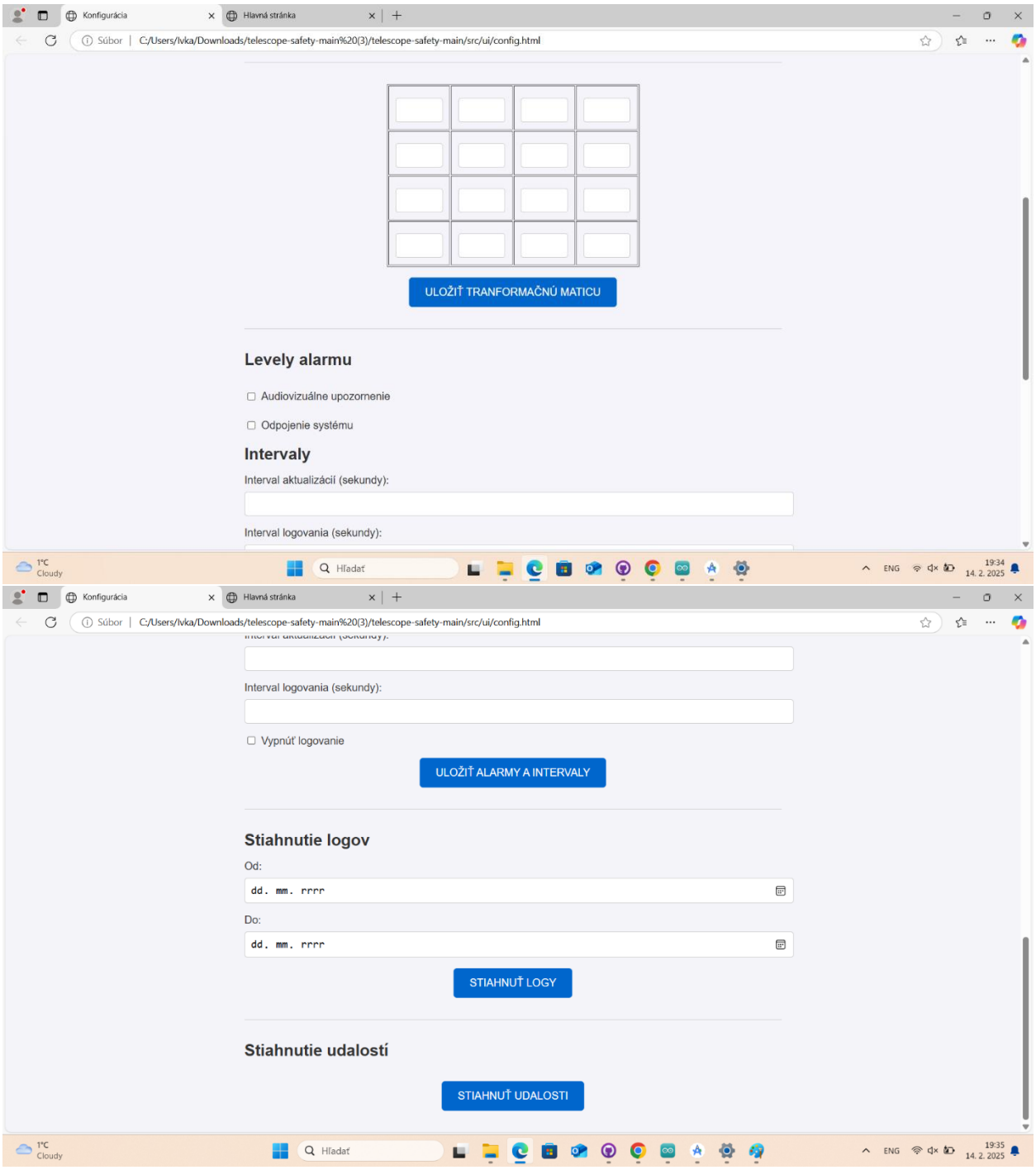


UML Component Diagram - kontrola polohy teleskopu



Návrh používateľského rozhrania:





Plán implementácie:

Modul s inerciálnou jednotkou

- program v jazyku C vytvorený vo vývojovom prostredí Arduino IDE pre mikropočítač Raspberry Pi Pico 2 na získanie údajov z inerciálnej jednotky 10 DOF IMU Sensor,ICM20948 vo forme Eulerových uhlov, výpočet kvaterniónov, prevod na azimut a eleváciu
- program pre vysielač modemu LoRa v jazyku C vytvorený vo vývojovom prostredí Arduino IDE na rádiové zasielanie údajov získaných zo senzoru
- program pre prijímač modemu LoRa na prijatie správy ohľadom resetovania a opätovnej kalibrácie modulu

Riadiaca jednotka

- program v jazyku C vytvorený vo vývojom prostredí Arduino IDE pre spracovanie údajov prijatých u prijímača modemu LoRa
- program v jazyku C vytvorený vo vývojom prostredí Arduino IDE pre posielanie správy modulu s inerciálnou jednotkou na jeho opätovnú kalibráciu a resetovanie
- program v jazyku C na vyhodnotenie nebezpečenstva podľa získaných a konfiguračných údajov, spúšťanie alarmu, vypnutie systému
- program v jazyku C na spracovanie konfiguračného súboru a jeho naparsovanie do dátovej štruktúry

- program v jazyku C pre príjem HTTP requestov a distribúciu statických súborov
- program v jazyku C pre príjem HTTP POST requestov na konfiguráciu systému
- program v jazyku C pre príjem HTTP POST requestov na zipovanie a stiahnutie logovych dat
- program v jazyku C pre upgrade HTTP na WebSocket a naslednu distribuciu aktuálnych dát z inerciálnej jednotky

- program pre logovanie dát, posielanie na webový server
- program na vypísanie dát na displej

Webové rozhranie

- program v html, css a javascript jazyku pre stránky používateľského rozhrania(konfiguračná stránka, informačná stránka)

Návrh implementácie:

```
struct Time{
  int year;
  int month;
```

```
  int day;
  int hours;
  int minutes;
  int seconds;
};

struct RollPitchYaw{
    float roll;
    float pitch;
    float yaw;
};

struct AzimuthElevation{
    float azimuth;
    float elevation;
};

struct DegreesMinutesSeconds {
    int degrees;
    int minutes;
    int seconds;
};

enum ChangeType{
    FORBIDDEN_ZONE_CHANGED,
    LOG_FREQUENCY_ALARM_CHANGED,
    RESTRT
};

// Inercialna jednotka

// Reads data from sensor and stores them in RollPitchYaw structure
// @return pointer to RollPitchYaw structure
RollPitchYaw* readFromSensor();

// Converts RollPitchYaw structure to AzimuthElevation structure
// @param rollPitchYaw pointer to RollPitchYaw structure
// @return pointer to AzimuthElevation structure
AzimuthElevation* fromRPYtoAzimuthElevation(RollPitchYaw*
rollPitchYaw);

// Sends AzimuthElevation structure to control unit
// @param azimutElevation pointer to AzimuthElevation structure
// @return 0 if success, -1 if error
int sendToControlUnit(AzimuthElevation* azimutElevation);

// Receiving message from control unit to restart
```

```
// @return 0 if success, -1 if error
int readFromControlUnit();

// Riadiaca jednotka

// Reads data from inertial unit and stores them in AzimuthElevation
structure
// @return pointer to AzimuthElevation structure
AzimuthElevation* readFromInertialUnit();

// Checks if AzimuthElevation structure is in forbidden zone
// @param azimutElevation pointer to AzimuthElevation structure
// @return 0 if not in forbidden zone, -1 if in forbidden zone
int checkForbiddenZone(AzimuthElevation* azimutElevation);

// Sends AzimuthElevation structure to all conected clients via WS
// @param azimutElevation pointer to AzimuthElevation structure
// @return 0 if success, -1 if error
int sendToClients(AzimuthElevation* azimutElevation);

// Dispalays AzimuthElevation structure on screen
// @param azimutElevation pointer to AzimuthElevation structure
// @return 0 if success, -1 if error
void displayAE(AzimuthElevation* azimutElevation);

// Writes AzimuthElevation structure to log file, with timestamp in
CSV format
// @param azimutElevation pointer to AzimuthElevation structure
// @return 0 if success, -1 if error
int writeAEtoLog(AzimuthElevation* azimutElevation);

// Checks if file format is correct
// also serves for parsing the forbidden zones into data structure
// @param newConfiguration from user, or loaded from file
// @return 0 if correct, -1 if zone has less than three points, -2 if
bad line format
// @note Different error codes can be added
// The format of the file is as follows:
// 1. line: Azimuth Elevation
// 2. line: Azimuth Elevation
// 3. line: Azimuth Elevation
// ...
// n. line: \n (empty line)
// n+1. line: Azimuth Elevation
// ...
// Azimuth and Elevation are separated by space
// Each Azimuth Elevation is separated by new line
```

```
// Each forbidden zone is separated by empty line
// Each forbidden zone contains at least 3 points
// Lines starting with # are ignored
int setUpZones(const char* newConfiguration);

// Parsing alarm type, logging status, log frequency and update
frequency from file to global variable settings
// @param newConfiguration loaded from file
// returns -1 if unsuccessful and 0 if successful
int setUpAlarmAndIntervals(const char* newConfiguration);

// Writes new forbidden zones configuration
// @param zones as written by user as const char* type
// @return 0 if success, -1 if error
int writeNewForbiddenConfig(const char* zones);

// Writes new log frequency configuration and alarm settings
// @param data has info on alarm and storing intervals settings
// @return 0 if success, -1 if error
int writeConfigAlarmAndIntervals(const char* data);

// Reads forbidden zones configuration
// stores forbidden zones to global variable
"settings.systemForbiddenZones"
// Reads log frequency configuration
// stores log frequency to global variable "settings.log_frequency"
// Reads alarm type configuration
// stores alarm type configuration to global variables
"settings.alarm" and "settings.rele"
// Reads logging configuration
// stores logging configuration to global variable "settings.logging"
// Reads update frequency configuration
// stores update frequency configuration to global variable
"settings.update_frequency"
// @return 0 if success, -1 if error
int loadSettings();

// Writes change of configuration to log file, with timestamp in CSV
format
// @param changeType type of change
// @return 0 if success, -1 if error
int writeChangeToLog(ChangeType changeType);

// Writes in witch position the telescope entered the forbidden zone
to log file, with timestamp in CSV format
// @param azimutElevation pointer to AzimuthElevation structure
int writeAlarmToLog(AzimuthElevation* azimutElevation);
```

```
// Starts alarm
// @return 0 if success, -1 if error
int startAlarm();

// Stops alarm
// @return 0 if success, -1 if error
int stopAlarm();

// Disables motors
// @return 0 if success, -1 if error
int disableMotors();

// Enables motors
// @return 0 if success, -1 if error
int enableMotors();

// Starts alarm and disables motors
// @return 0 if success, -1 if error
int enteredForbidenZone(AzimuthElevation* azimutElevation);

// Stops alarm and enables motors
// @return 0 if success, -1 if error
int reenable();

// Retrievs real time from Ethernet
// @return time in seconds
Time getRealTime();

// Restarts system
// @return 0 if success, -1 if error
int restart();

// Restarts inertial unit
// @return 0 if success, -1 if error
// @param azimuth value indicates calibration of azimuth on inertial
unit
    // if its -1 it stays the same and sensor just resets
int restartInertialUnit(double azimuth = -1);

// Setup HTTP server
// @return 0 if success, -1 if error
int setupHTTPServer();

// Setup WebSocket server
// @return 0 if success, -1 if error
int setupWebSocketServer();
```

```
// Setup mDNS server
// @return 0 if success, -1 if error
int setupMDNSServer();

// Starts all servers
// @return 0 if success, -1 if error
int startServers();


int enteredForbidenZone(AzimuthElevation* azimutElevation){
    writeAlarmToLog(azimutElevation);
    startAlarm();
    disableMotors();
    return 0;
}

int reenable(){
    stopAlarm();
    enableMotors();
    return 0;
}
struct Time{
  int year;
  int month;
  int day;
  int hours;
  int minutes;
  int seconds;
};

struct RollPitchYaw{
    float roll;
    float pitch;
    float yaw;
};

struct AzimuthElevation{
    float azimuth;
    float elevation;
};

struct DegreesMinutesSeconds {
    int degrees;
    int minutes;
    int seconds;
};
```

```
enum ChangeType{
    FORBIDDEN_ZONE_CHANGED,
    LOG_FREQUENCY_ALARM_CHANGED,
    RESTRT
};

// Inercialna jednotka

// Reads data from sensor and stores them in RollPitchYaw structure
// @return pointer to RollPitchYaw structure
RollPitchYaw* readFromSensor();

// Converts RollPitchYaw structure to AzimuthElevation structure
// @param rollPitchYaw pointer to RollPitchYaw structure
// @return pointer to AzimuthElevation structure
AzimuthElevation* fromRPYtoAzimuthElevation(RollPitchYaw*
rollPitchYaw);

// Sends AzimuthElevation structure to control unit
// @param azimutElevation pointer to AzimuthElevation structure
// @return 0 if success, -1 if error
int sendToControlUnit(AzimuthElevation* azimutElevation);

// Receiving message from control unit to restart
// @return 0 if success, -1 if error
int readFromControlUnit();

// Riadiaca jednotka

// Reads data from inertial unit and stores them in AzimuthElevation
structure
// @return pointer to AzimuthElevation structure
AzimuthElevation* readFromInertialUnit();

// Checks if AzimuthElevation structure is in forbidden zone
// @param azimutElevation pointer to AzimuthElevation structure
// @return 0 if not in forbidden zone, -1 if in forbidden zone
int checkForbiddenZone(AzimuthElevation* azimutElevation);

// Sends AzimuthElevation structure to all conected clients via WS
// @param azimutElevation pointer to AzimuthElevation structure
// @return 0 if success, -1 if error
int sendToClients(AzimuthElevation* azimutElevation);

// Dispalays AzimuthElevation structure on screen
// @param azimutElevation pointer to AzimuthElevation structure
```

```
// @return 0 if success, -1 if error
void displayAE(AzimuthElevation* azimutElevation);

// Writes AzimuthElevation structure to log file, with timestamp in
CSV format
// @param azimutElevation pointer to AzimuthElevation structure
// @return 0 if success, -1 if error
int writeAEtoLog(AzimuthElevation* azimutElevation);

// Checks if file format is correct
// also serves for parsing the forbidden zones into data structure
// @param newConfiguration from user, or loaded from file
// @return 0 if correct, -1 if zone has less than three points, -2 if
bad line format
// @note Different error codes can be added
// The format of the file is as follows:
// 1. line: Azimuth Elevation
// 2. line: Azimuth Elevation
// 3. line: Azimuth Elevation
// ...
// n. line: \n (empty line)
// n+1. line: Azimuth Elevation
// ...
// Azimuth and Elevation are separated by space
// Each Azimuth Elevation is separated by new line
// Each forbidden zone is separated by empty line
// Each forbidden zone contains at least 3 points
// Lines starting with # are ignored
int setUpZones(const char* newConfiguration);

// Parsing alarm type, logging status, log frequency and update
frequency from file to global variable settings
// @param newConfiguration loaded from file
// returns -1 if unsuccessful and 0 if successful
int setUpAlarmAndIntervals(const char* newConfiguration);

// Writes new forbidden zones configuration
// @param zones as written by user as const char* type
// @return 0 if success, -1 if error
int writeNewForbiddenConfig(const char* zones);

// Writes new log frequency configuration and alarm settings
// @param data has info on alarm and storing intervals settings
// @return 0 if success, -1 if error
int writeConfigAlarmAndIntervals(const char* data);

// Reads forbidden zones configuration
```

```
// stores forbidden zones to global variable
"settings.systemForbiddenZones"
// Reads log frequency configuration
// stores log frequency to global variable "settings.log_frequency"
// Reads alarm type configuration
// stores alarm type configuration to global variables
"settings.alarm" and "settings.rele"
// Reads logging configuration
// stores logging configuration to global variable "settings.logging"
// Reads update frequency configuration
// stores update frequency configuration to global variable
"settings.update_frequency"
// @return 0 if success, -1 if error
int loadSettings();

// Writes change of configuration to log file, with timestamp in CSV
format
// @param changeType type of change
// @return 0 if success, -1 if error
int writeChangeToLog(ChangeType changeType);

// Writes in witch position the telescope entered the forbidden zone
to log file, with timestamp in CSV format
// @param azimutElevation pointer to AzimuthElevation structure
int writeAlarmToLog(AzimuthElevation* azimutElevation);

// Starts alarm
// @return 0 if success, -1 if error
int startAlarm();

// Stops alarm
// @return 0 if success, -1 if error
int stopAlarm();

// Disables motors
// @return 0 if success, -1 if error
int disableMotors();

// Enables motors
// @return 0 if success, -1 if error
int enableMotors();

// Starts alarm and disables motors
// @return 0 if success, -1 if error
int enteredForbidenZone(AzimuthElevation* azimutElevation);

// Stops alarm and enables motors
```

```
// @return 0 if success, -1 if error
int reenable();

// Retrievs real time from Ethernet
// @return time in seconds
Time getRealTime();

// Restarts system
// @return 0 if success, -1 if error
int restart();

// Restarts inertial unit
// @return 0 if success, -1 if error
// @param azimuth value indicates calibration of azimuth on inertial
unit
    // if its -1 it stays the same and sensor just resets
int restartInertialUnit(double azimuth = -1);

// Setup HTTP server
// @return 0 if success, -1 if error
int setupHTTPServer();

// Setup WebSocket server
// @return 0 if success, -1 if error
int setupWebSocketServer();

// Setup mDNS server
// @return 0 if success, -1 if error
int setupMDNSServer();

// Starts all servers
// @return 0 if success, -1 if error
int startServers();


int enteredForbidenZone(AzimuthElevation* azimutElevation){
    writeAlarmToLog(azimutElevation);
    startAlarm();
    disableMotors();
    return 0;
}

int reenable(){
    stopAlarm();
    enableMotors();
    return 0;
}
```

Testovacie scenáre

1. Overenie základnej funkcionality: Monitorovanie polohy Zámer: Otestovať schopnosť systému presne monitorovať aktuálnu polohu teleskopu. Scenár: Nastavte teleskop do rôznych polôh v priestore (s bezpečnými a zakázanými súradnicami). Sledujte hodnoty azimutu a výšky na riadiacej jednotke a softvéri na PC. Porovnajte ich s manuálne zaznamenanými údajmi. Očakávaný výstup: Údaje o azimute a výške sa zobrazujú správne a sú zhodné s manuálne určenými hodnotami. Testované body: 3.1.8, 3.1.17, 3.1.18

2. Detekcia zakázaných oblastí Zámer: Overiť, či systém správne deteguje zakázané oblasti a spustí alarm. Scenár: Nastavte zakázané oblasti cez softvér na PC. Posuňte teleskop do súradníc v zakázanej oblasti. Sledujte správanie systému. Očakávaný výstup: Systém spustí vizuálny a akustický alarm a podľa konfigurácie vykoná ďalšie akcie (napr. odpojenie pohonu teleskopu). Testované body: 3.1.9, 3.1.10, 3.1.15

3. Konfigurácia zakázaných oblastí Zámer: Otestovať funkčnosť konfigurácie zakázaných oblastí. Scenár: Používateľ nakonfiguruje zakázané oblasti cez administratívne rozhranie softvéru na PC nahraním konfiguračného súboru a manuálnym zadaním súradníc. Následne skontroluje, či sa konfigurácia uložila a správne zobrazuje. Očakávaný výstup: Zakázané oblasti sú správne uložené a zobrazované v administratívnom rozhraní. Testované body: 3.1.11, 3.1.14, 3.1.15

4. Kalibrácia nulovej hodnoty azimutu Zámer: Overiť funkčnosť kalibrácie nulovej hodnoty azimutu. Scenár: Kalibrujte nulovú hodnotu azimutu manuálne na riadiacej jednotke a cez softvér na PC. Skontrolujte, či sa zmeny správne prejavili v systéme a údajoch o polohe. Očakávaný výstup: Kalibrácia je úspešná, nulová hodnota azimutu sa správne nastaví a prejaví v údajoch. Testované body: 3.1.13, 3.1.16

5. Diagnostika a zaznamenávanie udalostí Zámer: Otestovať schopnosť systému zaznamenávať polohu a udalosti. Scenár: Simulujte pohyb teleskopu vrátane vstupu do zakázaných oblastí. Skontrolujte záznamy v systéme na riadiacej jednotke a softvéri na PC. Exportujte ich do súboru. Očakávaný výstup: Záznamy obsahujú všetky polohy a udalosti vrátane alarmov. Export je úspešný. Testované body: 3.1.21, 3.1.22

6. Funkcia pri výpadku napájania Zámer: Overiť, či systém funguje pri výpadku hlavného napájania. Scenár: Odpojte systém od hlavného napájania a skontrolujte, či sa prepol na záložný zdroj (powerbanku). Sledujte, či všetky funkcie zostávajú aktívne. Očakávaný výstup: Systém pokračuje v plnej prevádzke bez výpadkov. Testované body: 3.1.19, 2.5

7. Komunikácia medzi modulmi Zámer: Overiť stabilitu rádiovej komunikácie medzi modulmi. Scenár: Testujte komunikáciu medzi inerciálnou jednotkou, riadiacou jednotkou a softvérom na PC v prítomnosti iných rádiových zariadení (napr. WiFi). Očakávaný výstup: Komunikácia je stabilná bez výpadkov a chýb. Testované body: 3.1.3, 2.4

8. Nezávislosť systému Zámer: Overiť schopnosť systému fungovať nezávisle od elektrickej siete, internetovej siete a iných programov. Scenár: Odpojte systém od elektrickej siete, internetovej siete a softvéru na PC. Skontrolujte, či všetky funkcie (monitorovanie polohy, alarmy, zaznamenávanie udalostí) zostávajú aktívne. Očakávaný výstup: Systém pokračuje v plnej prevádzke a nezávisí od externých sietí alebo softvéru. Testované body: 3.1.1

9. Zostava systému Zámer: Overiť, či všetky komponenty systému fungujú podľa špecifikácie a sú správne prepojené. Scenár: Zapojte modul s inerciálnou jednotkou, riadiacu jednotku a softvér na PC. Overte, či sú všetky komponenty schopné komunikovať a systém poskytuje správne údaje o polohe. Očakávaný výstup: Všetky komponenty sú správne prepojené a funkčné, údaje o polohe sa zobrazujú bez chýb. Testované body: 3.1.2

10. Kvalita senzorov a presnosť údajov Zámer: Overiť presnosť údajov získaných z inerciálnej jednotky (akcelerometra a gyroskopu). Scenár: Nastavte teleskop do známych referenčných polôh v rôznych osiach. Porovnajte údaje z inerciálnej jednotky s údajmi z presného referenčného zariadenia. Očakávaný výstup: Údaje z inerciálnej jednotky sú v tolerancii stanovených chýb. Testované body: 3.1.4, 2.4

11. Integrácia modulu s riadiacou jednotkou Zámer: Overiť, či modul s inerciálnou jednotkou správne komunikuje s riadiacou jednotkou cez rádiový modem. Scenár: Simulujte rôzne pohyby teleskopu a sledujte, či riadiaca jednotka prijíma údaje z modulu. Očakávaný výstup: Riadiaca jednotka prijíma všetky údaje presne a bez výpadkov. Testované body: 3.1.3

12. Presnosť súradníc Zámer: Overiť správne zobrazenie aktuálnej polohy teleskopu pomocou dvojice azimutvýška. Scenár: Nastavte teleskop do známych referenčných bodov s konkrétnymi hodnotami azimutu a výšky. Sledujte zobrazenie údajov na riadiacej jednotke a porovnajte ich s manuálne zistenými hodnotami. Očakávaný výstup: Zobrazené hodnoty azimutu a výšky sa zhodujú s manuálne určenými údajmi. Testované body: 3.1.12

13. Poskytovanie údajov externému softvéru Zámer: Otestovať správne poskytovanie údajov o polohe teleskopu externému softvéru. Scenár: Pripojte externý softvér k systému a sledujte, či získava údaje o polohe teleskopu v reálnom čase. Overte, či sú údaje správne interpretované. Očakávaný výstup: Externý softvér prijíma presné a aktuálne údaje o polohe teleskopu. Testované body: 3.1.20

14. Automatická správa úložiska Zámer: Overiť schopnosť systému automaticky spravovať úložisko pre záznamy udalostí. Scenár: Uložte veľké množstvo záznamov o pohyboch a udalostiach, kým sa úložisko naplní. Sledujte, či systém automaticky maže staršie záznamy a pokračuje v zaznamenávaní nových udalostí. Očakávaný výstup: Systém maže staré záznamy a spravuje úložisko bez potreby manuálneho zásahu. Testované body: 3.1.23

Testovanie senzorov(inerciálnych jednotiek) a ich výber

- spomedzi senzorov sa najviac v testovacej fáze osvedčili tri:
- Waveshare 10 DOF IMU Sensor (D), ICM20948 Onboard
- SparkFun 6DoF IMU Breakout - ISM330DHCX
- SparkFun 9DoF IMU Breakout - ICM-20948

SparkFun 6DoF IMU Breakout – ISM330DHCX senzor je obzvlášť dobrý vďaka kalibrácii offsetov pri inicializácii (teoreticky možno inicializovať ako sa ďalekohľad hýbe) a taktiež vďaka Mahonny filtru, ktorý na úrovni microsekúnd odstraňuje odchýlku medzi uhlami roll, pitch, yaw. Jeho presnosť si však žiada daň v tom, že sa musí inicializovať vo vodorovnej polohe.

Waveshare 10 DOF IMU Sensor (D), ICM20948 Onboard senzor je spoľahlivý, avšak po resete a kalibrácii azimutu užívateľom cez tlačítka klesá hodnota azimutu zrejme kvôli jeho prístupu rátania uhlov roll pitch, yaw v dmp. Taktiež pri inicializácii by sa nemal hýbať nakoľko sa potom trošku menia hodnoty aj keď sa ďaekohľad nehýbe.

SparkFun 9DoF IMU Breakout - ICM-20948 senzor je veľmi podobný waveshare senzoru. Jediná vec čo sa pri ňom v kóde mení je adresa s ktorou funguje v I2C komunikácii s mikrokontrolérom.

Senzory možno dynamicky meniť za behu systému avšak potom by sa mal reštartovať senzor cez webstránku. Zmena senzora nie je zavislá na kompilácii.

Po inicializácii všetky senzory nastavia hodnotu azimutu na nulu, to znamená že nie je to automaticky podľa severu. V tomto prípade vie užívateľ nastaviť azimut senzora cez tlačítka a displej na riadiacej jednotke. Jedným sa dostane do nastavení, druhým sa presúva medzi hodnotami, tretím nastavuje hodnoty a štvrtým odosiela príkaz inerciálnej jednotke na reset s nastaveným azimutom.

Ovládanie systému

Webstránka

Hlavná stránka

Na hlavnej stránke (teleskop.local) sa nachádza informácia o aktuálnej polohe teleskopu. Dole pod polohou sa zobrazuje dátum a čas poslednej prijatej správy o aktuálnej polohe.
Nad touto polohou sa zobrazuje sprava o prípadnom spustení alarmu alebo pripájaní k serveru ak by z nejakého dôvodu nebol dostupný.
Pod zobrazením polohy sa nachádza tlačidlo, ktorým užívateľ prejde na nastavenie konfigurácie (link: teleskop.local/config)

Konfiguračná stránka

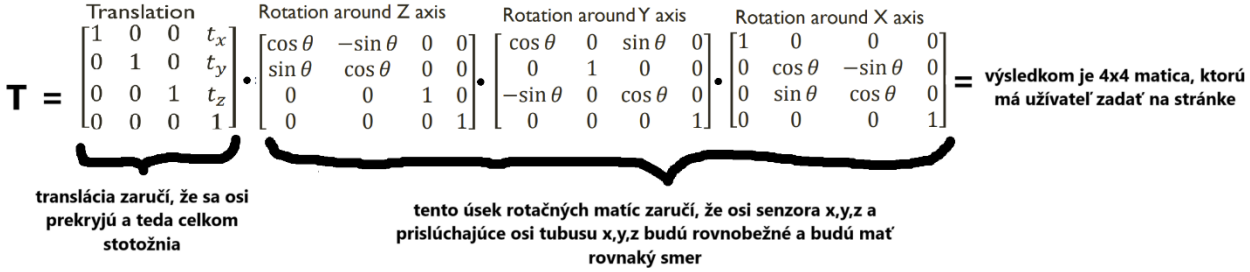
Na konfiguračnej stránke sa dá celý systém nastavovať. Na vrchu vľavo sa nachádza tlačidlo, ktorým užívateľ prejde naspäť na hlavnú stránku.
Prvé nastavenie je veľké červené tlačidlo, ktoré vyšle správu pre reštart inerciálneho senzoru. Reštart treba vykonávať za pokojného stavu teleskopu v polohe zvislej (inerciálny senzor musí byť vodorovne), čiže pri elevácii 90.
Ďalej si vie užívateľ navoliť zakázané zóny (poligóny), do ktorých teleskop nesmie vojsť (z bezpečnostných dôvodov). Každú zónu musia tvoriť aspoň 3 body. Každý bod je tvorený z azimutu a elevácie a ja v samostatnom riadku. Riadky začínajúce znakom "#" sú ignorované vnútri systému. Slúžia pre užívateľa, aby si dokázal pomenovať jednotlivé zóny. Zóny musia byť oddelené jedným prázdnym riadkom.

Príklad takéhoto nastavenia 2 zón:

```
# zóna brániaca príliš nízkej elevácii
0 45
360 45
0 -90
365 -90

#druha zóna popisujúca nohu ďalekohľadu
58 60
64 60
60 50
```

Ďalej si vie užívateľ nastaviť 4x4 transformačnú maticu, ktorá slúži na prerátanie azimutu a elevácie senzora na azimut a eleváciu strednej osi tubusu. Matica sa odvíja od toho kde je senzor na teleskope pripojený(matica posunu/translácie- posun vyjadrený v stupňoch) a takisto ako je lokálne voči osiam tubusu natočený(súčin rotačných matíc okolo osí x,y,z). Opisný obrázok:



V ďalšej časti sa dá nastaviť, či ma systém spúšťať alarm pri vstupe do zakázanej zóny, či má odpojiť napájanie motorov, ktoré ovládajú teleskop, ako často sa majú posielať údaje o akatuálnej polohe z riadiacej jednotky na web, ako často sa ma aktuálna (okamžitá) poloha teleskopu zapisovať do súboru s logmi, či sa majú takéto logy vôbec zapisovať podľa tohto nastavenia (ak sú vypnuté logy nezapisujú sa. Logy o vstupe do zakázanej zóny a konfiguráciách sa vždy zapíšu).
V ďalšej sekcii si vieme navoliť 2 dátumy od kedy do kedy si chceme stiahnuť logy. Po potvrdení sa JavaScript na stránke postará o postupne stiahnutie všetkých existujúcich súborov s logmi, ktoré vyhovujú intervalu.
V poslednej sekcii si rovnako vieme stiahnuť Logy, tentokrát však udalosti. Udalosťou chápeme vstup do zakázanej zóny alebo zmenu konfigurácie. Pre oba tieto druhy udalosti je zvlášť súbor.

Displej

Displej zobrazuje aktuálnu polohu teleskopu. Poloha je reprezentovaná azimutom a elevációu, ktoré sú vo forme “stupne minúty sekundy”. Poloha sa aktualizuje každých 10 sekúnd. Po inicializácii všetky senzory nastavia hodnotu azimutu na nulu, to znamená, že nie je to automaticky podľa severu. V tomto prípade vie užívateľ nastaviť azimut senzora cez tlačidlá a displej. Do nastavení sa užívateľ dostane stlačením 1. tlačidla (naspäť sa dostane tiež stlačením 1. tlačidla). V nastaveniach je vizuálne vyznačené, v ktorej časti azimutu sa užívateľ nachádza (stupne, minúty, sekundy). Užívateľ sa dokáže medzi týmito časťami prepínať stačením 2. tlačidla. Pomocou 3. a 4. tlačidla vie dané číslo zvyšovať a znižovať o 1. Keď je nastavený požadovaný azimut, treba ho ešte potvrdiť. Potvrdí sa podržaním 1. tlačidla minimálne 2 sekundy. Nový azimut sa odošle inerciálnej jednotke a na displeji sa zobrazí obrazovka s aktuálnou polohou teleskopu. Funkcie tlačidiel sú znázornené vizuálne na displeji.

Ako na úpravu kódu

Pre úpravu kódu si treba stiahnuť Arduino IDE. Nainštalovať všetky potrebné (nižšie spomenuté) knižnice. Urobiť fork z github repozitára, zakúpiť si potrebný HW a zapojiť ho podľa priložených schém.

Použité knižnice

Aby programy fungovali, je potrebné stiahnuť nasledovné knižnice buď priamo pomocou Arduino IDE Library managera alebo ich uložiť ich do Dokumenty/Arduino/libraries

Adafruit_BusIO - https://github.com/adafruit/Adafruit_BusIO

Adafruit_GFX_Library - https://github.com/adafruit/Adafruit-GFX-Library

MCUFRIEND_kbv - https://github.com/prenticedavid/MCUFRIEND_kbv

NTPClient - https://github.com/arduino-libraries/NTPClient

SimpleHOTP - https://github.com/jlusPrivat/SimpleHOTP

Time-master - https://github.com/PaulStoffregen/Time

UTFT - http://www.rinkydinkelectronics.com/library.php?id=51#google_vignette

arduino-LoRa-master - https://github.com/sandeepmistry/arduino-LoRa

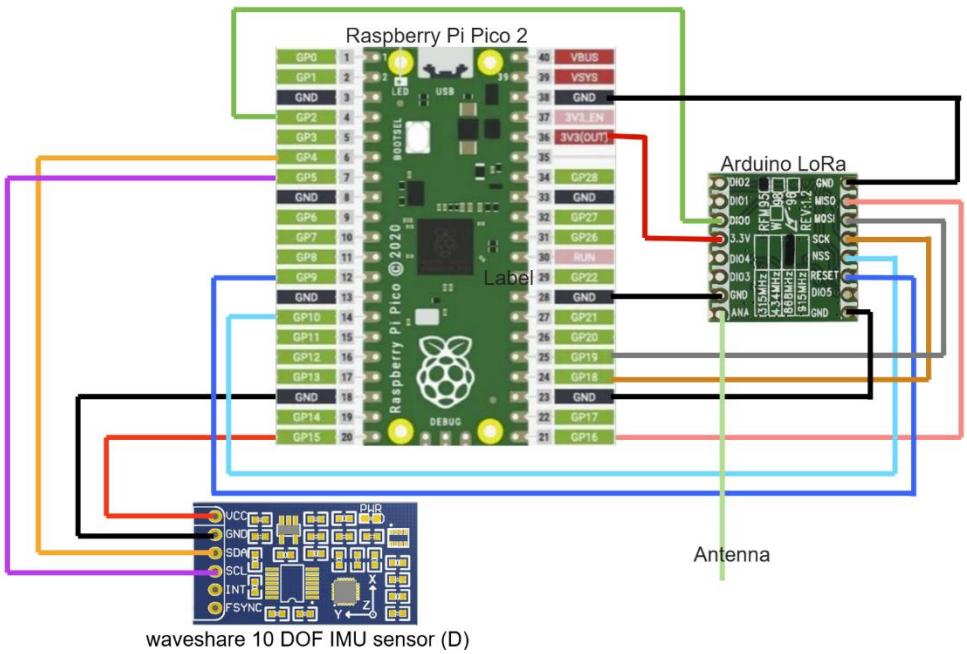
SparkFun_6DoF_ISM330DHCX_Arduino_Library-main - https://github.com/sparkfun/SparkFun_6DoF_ISM330DHCX_Arduino_Library

SparkFun_ICM-20948_ArduinoLibrary-main - https://github.com/sparkfun/SparkFun_ICM-20948_ArduinoLibrary

Hardvér

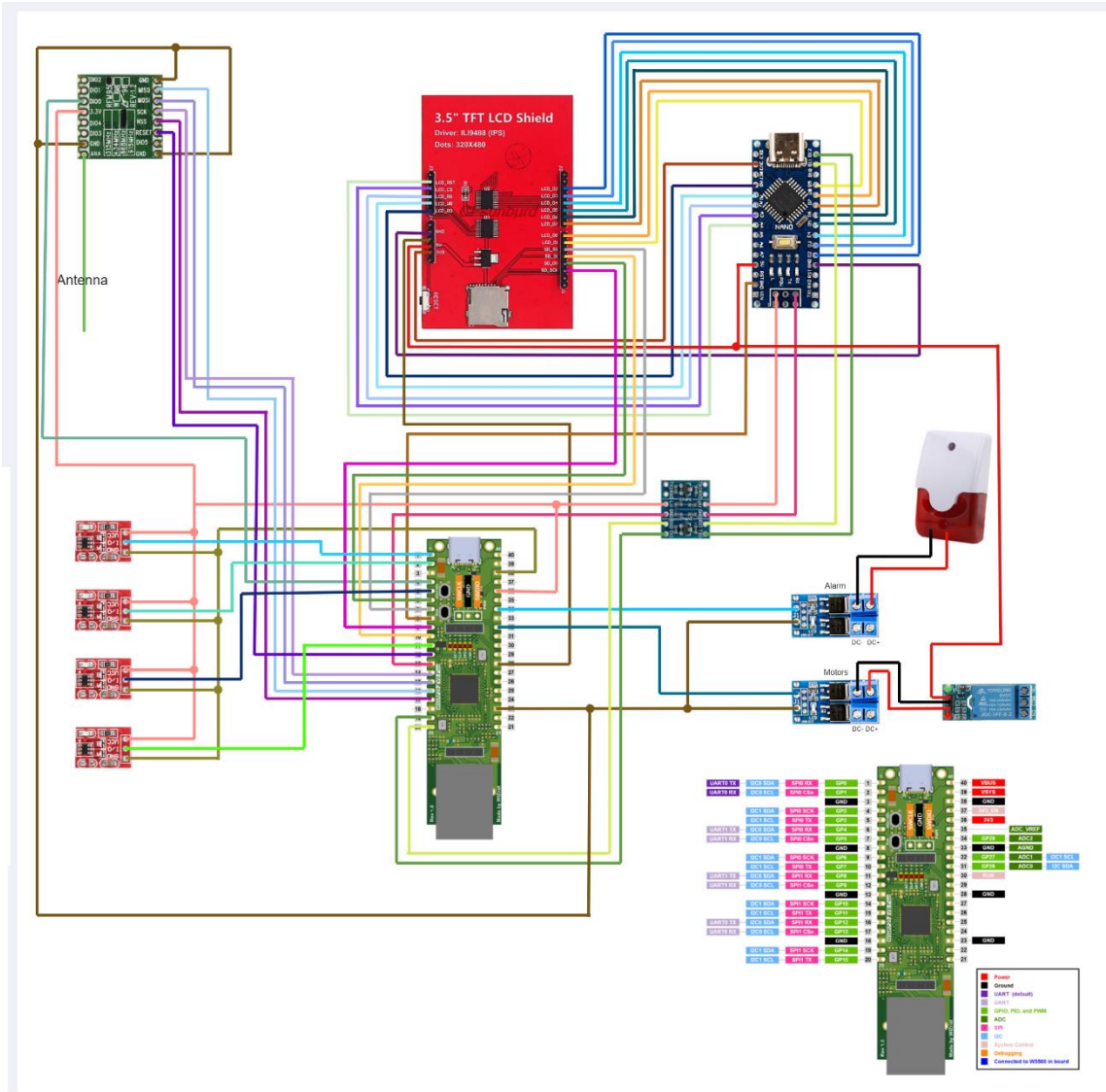
Inerciálna jednotka

- Raspberry Pico 2 - https://www.raspberrypi.com/products/raspberry-pi-pico-2/
- Senzor (alternatívy)
    - Waveshare 10 DOF IMU Sensor (D), ICM20948 Onboard - https://www.sparkfun.com/sparkfun-6dof-imu-breakout-ism330dhcx-qwiic.html
    - SparkFun 6DoF IMU Breakout – ISM330DHCX - https://www.sparkfun.com/sparkfun-6dof-imu-breakout-ism330dhcx-qwiic.html (najlepšia voľba)
    - SparkFun 9DoF IMU Breakout - ICM-20948 - https://www.sparkfun.com/sparkfun-9dof-imu-breakout-icm-20948-qwiic.html
- LoRa 868MHz SX1276 RF Transreceiver Module RFM96W - https://techfun.sk/produkt/komunikacny-modul-sx1276-lora-433-868-915-mhz/
- Zdroj 5V4A - https://chipskey.en.made-in-china.com/product/ATJRiUelIGWN/China-47126-Output-5V-97-Mini-DC-DC-15V-12V-9V-7-4V-to-5V-4A-Step-Down-Buck-Power-Converter-Charging-Voltage-Regulator-Module.html
- Powerbanka - https://www.alza.sk/spello-vreckova-powerbanka-5000-mah-cierna-d12422928.htm
- Krabička v ktorej bude hardvér uložený - https://www.tinkercad.com/things/2qtOncCJI9N-inertial-unit



Riadiaca jednotka

- W55RP20-EVB-PICO - https://docs.wiznet.io/Product/ioNIC/W55RP20/w55rp20-evb-pico
- Arduino NANO - https://store.arduino.cc/en-sk/products/arduino-nano
- 3.5 TLT LCD Shield Display - https://techfun.sk/produkt/tft-display-shield-3-5-pre-arduino-uno-s-sd-kartou
- Micro SD Card
- LoRa 868MHz SX1276 RF Transreceiver Module RFM96W
- 4x Kapacitné dotykové tlačidlo TTP223
- 2x MosfetDual mosfet PWM regulátor AOD4184A - https://techfun.sk/produkt/dual-mosfet-pwm-regulator-aod4184a/
- Relé modul 1 kanál 5V - https://techfun.sk/produkt/rele-modul-1-kanal-5v/
- Siréna 12V LS-103 - https://techfun.sk/produkt/sirena-12v-ls-103/



Komunikačné protokoly:

Websocket

WS beží na porte :81. Všetkým pripojeným klientom posiela aktuálne informácie vo formáte JSON.

JSON obsahuje:

```
{"azimuth":[azimuth zaoktuhleny na 2 desatine
miesta],"elevation":(elevacia zaoktuhlena na 2 desatine miesta)}
```

V pripade potreby odoslat nejaku spravu pripojenym klientom moze JSON vyzerat aj nasledovne:

```
{"azimuth":[azimuth zaoktuhleny na 2 desatine
miesta],"elevation":(elevacia zaoktuhlena na 2 desatine
miesta),"error":"[message text]"}
```

Alebo:

```
{"azimuth":[azimuth zaoktuhleny na 2 desatine
miesta],"elevation":(elevacia zaoktuhlena na 2 desatine
miesta),"warning":"[message text]"}
```

Riadiaca jednotka a displej

Riadiaca jednotka posiela nasledovné správy:

BUTTON1\n - prepína medzi obrazovkou, ktorá zobrazuje aktuálnu polohu teleskopu a obrazovkou, kde sa nastavuje azimut
BUTTON2\n - prepína medzi stupňami, minútami a sekundami azimutu
BUTTON3\n - zvyšuje sa vybrané číslo
BUTTON4\n - znižuje sa vybrané číslo
SEND\n - získa z displeja nastavený azimut, aby sa mohol poslať inerciálnej jednotke
{azimut stupne}{azimut minúty}{azimut sekundy}\n{elevácia stupne}{elevácia minúty}{elevácia sekundy}\n - zobrazí poslanú polohu na displeji

Displej posiela nasledovné správy:

{azimut stupne}{azimut minúty}{azimut sekundy}\n - pošle nastavený azimut na základe dopytu z riadiacej jednotky

Modul s inerciálnou a riadiaca jednotka – protokol LoRa

Modul s inerciálnou posiela pomocou LoRa protokolu cez rádio v paketoch správy v tvare:

```
{svoju adresu}{dĺžka správy}{azimuth: %d, elevation: %d}
```

- správa obsahuje údaj o aktuálnom azimute a elevácii tubusu, adresu a dĺžku správy posiela kvôli bezpečnosti

```
{svoju adresu}{dĺžka správy}{"ACK:RESTART_INERTIAL_UNIT"}
```

- správa obsahuje informáciu pre riadiacu jednotku, že modul s inerciálnou prijal príkaz na reset a vykonal ho, adresu a dĺžku správy posiela kvôli bezpečnosti

Riadiaca jednotka posiela správy v tvare:

```
{svoju adresu}{dĺžka správy}{RESTART_INERTIAL_UNIT: %d}
```

- posiela informáciu o tom že sa má inerciálna resetovať a azimuth hodnotu na ktorú sa má nakalibrovať senzor, prípadne sa má len resetovať ak je azimut -1, adresu a dĺžku správy posiela kvôli bezpečnosti

```
{svoju adresu}{dĺžka správy}{|SET_CALIBRATION_MATRIX:
%int,int,int,int,int,int,int,int,int,int,int,int,int,int,int,int}
```

- posiela informáciu o novej transformačnej matici ktorú si má inerciálna naparsovať zo stringu v tvare 16tich integerov oddeleným čiarkami, adresu a dĺžku správy posiela kvôli bezpečnosti