Fakulta matematiky, fyziky a informatiky UK

Návrh

Projekt Serial transformers

Vypracovali : Ivana Bekešová, Andrej Nagy, Zdenko Németh, Mykola Shulhin

Obsah

1. Špecifikácia vonkajších interfejsov

Táto kapitola popisuje všetky vonkajšie rozhrania informačného systému, ktoré sú zodpovedné za jeho komunikáciu s ostatnými aplikáciami, súbormi, zariadeniami.

1.1. Hardvérové rozhranie

Táto kapitola popisuje použitý hardvér jeho špecifikáciu a účel, pre ktorý bol použitý.

1.1.1. Teensy 4.1 Microcontroller

Mikrokontrolér Teensy 4.1 slúži ako hlavná riadiaca jednotka systému, na ktorej beží softvér, ktorý spracúva prichádzajúce dáta a implementuje logiku transformácie a filtrovania paketov. V rámci informačného systému umožňuje transformovať, filtrovať pakety pre ovládanie rýchlosti krokových motorov teleskopu. Prepojenie s riadiacim PC je pomocou USB a s krokovými motormi teleskopu cez sériové rozhrania RS232 a RS485.

Špecifikácia:

- ARM Cortex-M7 at 600MHz
- 1024K RAM (512K is tightly coupled)
- 8 Mbyte Flash (64K reserved for recovery & EEPROM emulation)
- USB Host Port
- 2 chips Plus Program Memory
- 55 Total I/O Pins
- 3 CAN Bus (1 with CAN FD)
- 2 I2S Digital Audio
- 1 S/PDIF Digital Audio
- 1 SDIO (4 bit) native SD
- 3 SPI, all with 16 word FIFO
- 7 Bottom SMT Pad Signals
- 8 Serial ports
- 32 general purpose DMA channels
- 35 PWM pins
- 42 Breadboard Friendly I/O
- 18 analog inputs
- Cryptographic Acceleration
- Random Number Generator
- RTC for date/time
- Programmable FlexIO
- Pixel Processing Pipeline
- Peripheral cross triggering
- 10 / 100 Mbit DP83825 PHY (6 pins)
- microSD Card Socket
- Power On/Off management

1.1.2. RS232 na RS485 prevodník

Tento prevodník zabezpečuje transformáciu signálu medzi komunikačnými štandardmi RS232 (Teensy) a RS485 (krokové motory teleskopu).

1.2. Softvérové rozhranie

1.2.1. PC Softvér

Softvér slúži ako grafické používateľské rozhranie (GUI) pre výber modulu, jeho ovládanie, nastavenie parametrov a sledovanie stavu systému, komunikácie.
Štruktúra transformačného modulu pre teleskop:

- Prepínač režimov – manuálny/automatický/bypass
- Vstupná hodnota – manuálne nastavenie rýchlosti
- Vizualizácia transformácie – graficky znázornená transformácia s možnosťou nastavenia intervalu priemerovania
- Zobrazenie dát o paketoch – infromácie o počte so zmenenou rýchlosťou, pozdržaných, nezmenených, zadržaných
- Tlačidlá pre manipuláciu logov – zobrazenie/stiahnutie/zmazanie

1.1.1. Komunikácia s Autoguider kamerou

Komunikácia s Autoguider kamerou prebieha cez webový server. Komunikačné rozhranie:

- Kamera odosiela údaje cez protokol HTTP alebo HTTPS na špecifický port (napr. 8080).
- Kamera posiela údaje v JSON formáte prostredníctvom HTTP POST požiadavky na adresu webserveru.
- Kamera odosiela údaje v pravidelných časových intervaloch.
- Webserver prijíma tieto POST požiadavky a zapisuje údajo do interného systému alebo logu.
- Ak webserver zaznamená chybu pri prijímaní údajov, loguje tento incident s časovou pečiatkou pre ďalšie diagnostické účely.

2. Dátový model perzistentných údajov pre transformačný modul

V dátovom modeli budú reprezentované údaje uchovávané počas celej prevádzky systému, ktoré sú dôležité pre konfiguráciu, logovanie a transformáciu komunikácie.

2.1. Konfiguračné údaje

Konfigurácia je uložená v štruktúrovanom súbore, ktorý obsahuje nasledujúce nastavenia:

- **Mód (mode):** prepína medzi automatickým a manuálnym módom.
- **Transformačný modul:** identifikátor aktívneho transformačného modulu (teleskopický alebo univerzálny).
- **Konštanty zrýchlenia/spomalenia (acceleration/deceleration constants):** nastavuje pevnú konštantu zrýchlenia pre os RA.
- **Bypass mód:** binárna hodnota určujúca, či systém funguje v móde bypass (1 = bypass, 0 = transformácia).
- **Logovanie činnosti:** úroveň logovania (napr. "plné logovanie," "iba chybové hlásenia").

  Logovanie paketov: parameter, ktorý určuje, či sa majú logovať prichádzajúce a odchádzajúce pakety. "Áno" pre zapnutie logovania alebo "nie" pre jeho vypnutie.

2.2. Údaje o zaznamenaných paketoch

Každý záznam o pakete obsahuje:

- **ID záznamu (record_id):** unikátne ID záznamu.
- **Časová pečiatka (timestamp):** čas príchodu paketu do systému (a čas po transformácii, ak sa transformuje).
- **Originálny paket (original_packet):** surové dáta paketu, ako prišli zo vstupu.
- **Transformovaný paket (transformed_packet):** výsledný paket po aplikovaní transformačných pravidiel (ak je aktívna transformácia).

2.3. Údaje o používateľských a automatických akciách/udalostiach

Každá akcia je logovaná s nasledujúcimi atribútmi:

- **Čas akcie (action_timestamp):** čas, kedy bola akcia vykonaná.
- **Typ akcie (action_type):** popis akcie (napr. zmena módu, reset systému, nová konštanta z kamery, nová nakalibrovaná hodnota pre určitú deklináciu a pod.).
- **Parametre akcie (action_parameters):** prípadné dodatočné informácie o akcii, ako napr. hodnota nastavenej konštanty.

2.4. Automaticky kalibrované konštanty

- Tabuľka s poslednou najlepšou platnou verziou nájdených automatických konštánt

3. Formát súborov pre transformačný modul pre teleskop

Pre komunikáciu a ukladanie dát systém využíva nasledujúce formáty súborov

3.1. Konfiguračný súbor

- **Formát:** TXT.
- **Obsah:** Obsahuje konfiguračné údaje, ktoré systém načíta pri štarte, vrátane nastavení pre transformáciu, bypass mód, a konfigurácie transformačných modulov.

3.2. Debug-Log súbor

- **Formát:** TXT.
- **Obsah:** Tento log uchováva informácie o vykonaných operáciách systému, ako je nastavenie rýchlosti motorov, aplikované pravidlá pre transformáciu, a akékoľvek chyby alebo výnimky počas behu systému**.**
- **Štruktúra logu:**
  - **Časová pečiatka, Úroveň závažnosti, Správa, Kód chyby**

3.3. Paket-Log súbor

- **Formát:** TXT.
- **Obsah:** Logy obsahujú zaznamenané údaje o prichádzajúcich a odchádzajúcich paketov, časové pečiatky, typ paketu, záznam o vykonaných transformáciách. Binárne pakety sú logované v ASCII podobe, t.j. napr. bajt s hodnotou 0x14 sa zobrazí ako textový reťazec „14“
- **Štruktúra logu:**
  - **Časová pečiatka, Čas zdržania, Originálny paket, Transformovaný paket**

4. Komunikačné protokoly

Pre komunikáciu v master-slave architektúre systému je potrebný špecifický protokol, ktorý zahŕňa príkazy pre riadenie teleskopu a dynamickú úpravu rýchlosti motorov.

4.1. Master-Slave Komunikácia

- **Formát telegramu:** každý telegram obsahuje hlavičku, adresu, príkaz, a checksum.

- **Hlavička (header):** identifikuje začiatok telegramu.
- **Adresa (address):** špecifikuje, či ide o príkaz pre os RA alebo DEC.
- **Príkaz (command):** binárna hodnota určujúca typ akcie (napr. zvýšenie/zníženie rýchlosti).
- **Dátová sekcia (data):** informácie o požadovanej rýchlosti, oneskorení, prípadne ďalších parametroch pre transformáciu.
- **Checksum:** zabezpečuje správnosť telegramu.

4.2. API pre Autoguider

Externý softvér komunikuje s transformačným zariadením cez špeciálne definované API. Toto API obsahuje príkazy pre:

- **Dynamickú zmenu konštanty zrýchlenia/spomalenia:**
  - **API príkaz:** SET_ACCEL_CONST
  - **Popis:** Tento príkaz umožňuje zmenu konštanty zrýchlenia alebo spomalenia zariadenia. Hodnota konštanty je zadávaná ako parameter v príkaze a odosiela sa prostredníctvom HTTP požiadavky na API zariadenia.
  - **Formát**: http://IP_ADRESA/camera_api?passwd=HESLO&accel_const=HODNOTA
    - passwd=HESLO – Heslo pre autentifikáciu prístupu k API.
    - accel_const=HODNOTA – Nová hodnota konštanty zrýchlenia/spomalenia, ktorá bude nastavená na zariadení.

- **Reset zariadenia:**
  - **API príkaz:** RESET
  - **Popis:** Tento príkaz slúži na resetovanie zariadenia, čím sa zariadenie vráti do pôvodného stavu.
  - **Formát:** http://IP_ADRESA/camera_api?passwd=HESLO&reset=1
    - passwd=HESLO – Heslo pre autentifikáciu prístupu k API.
    - reset=1 - indikuje, že zariadenie má byť resetované. Hodnota 1 znamená, že reset bude vykonaný.

5. Dátový model perzistentných údajov pre univerzálny transformačný modul

5.1. Konfiguračné údaje pre univerzálny modul

Univerzálny modul vyžaduje dodatočné konfiguračné údaje pre pravidlá transformácie:

- **Pravidlá transformácie:** Pole pravidiel, kde každé pravidlo obsahuje:
  - **ID pravidla:** Unikátne identifikátor pravidla.
  - **Vzor (pattern):** Regulárny výraz, ktorý opisuje, aké pakety sa majú transformovať.
  - **Náhrada (replacement):** Text alebo vzor, ktorým sa nahradí zodpovedajúca časť paketu.
  - **Aktivácia pravidla:** Binárna hodnota určujúca, či je pravidlo aktívne (1 = aktívne, 0 = neaktívne).
- **Poradie pravidiel:** Definuje prioritu aplikácie pravidiel, ak ich je viacero.

5.2. Údaje o zaznamenaných paketoch pre univerzálny modul

Rovnaké ako pre teleskopický modul:

- **Originálny paket:** Surové dáta paketu.
- **Transformovaný paket:** Paket po aplikácii pravidiel univerzálneho modulu.
- **Aplikované pravidlá:** Zoznam ID pravidiel, ktoré boli aplikované na konkrétny paket.

5.3. Logovanie pre univerzálny modul

Rozšírenie logovania pre univerzálny modul:

- **ID aplikovaného pravidla:** Identifikátor pravidla, ktoré bolo použité na transformáciu.

6. Formát súborov pre univerzálny transformačný modul

6.1. Konfiguračný súbor pre univerzálny modul

- **Formát**: TXT.
- **Obsah**: Obsahuje konfiguračné údaje, ktoré systém načíta pri štarte, vrátane nastavení pre transformáciu, bypass mód a konfigurácie pravidiel transformácie.

6.2. Debug-Log súbor pre univerzálny modul

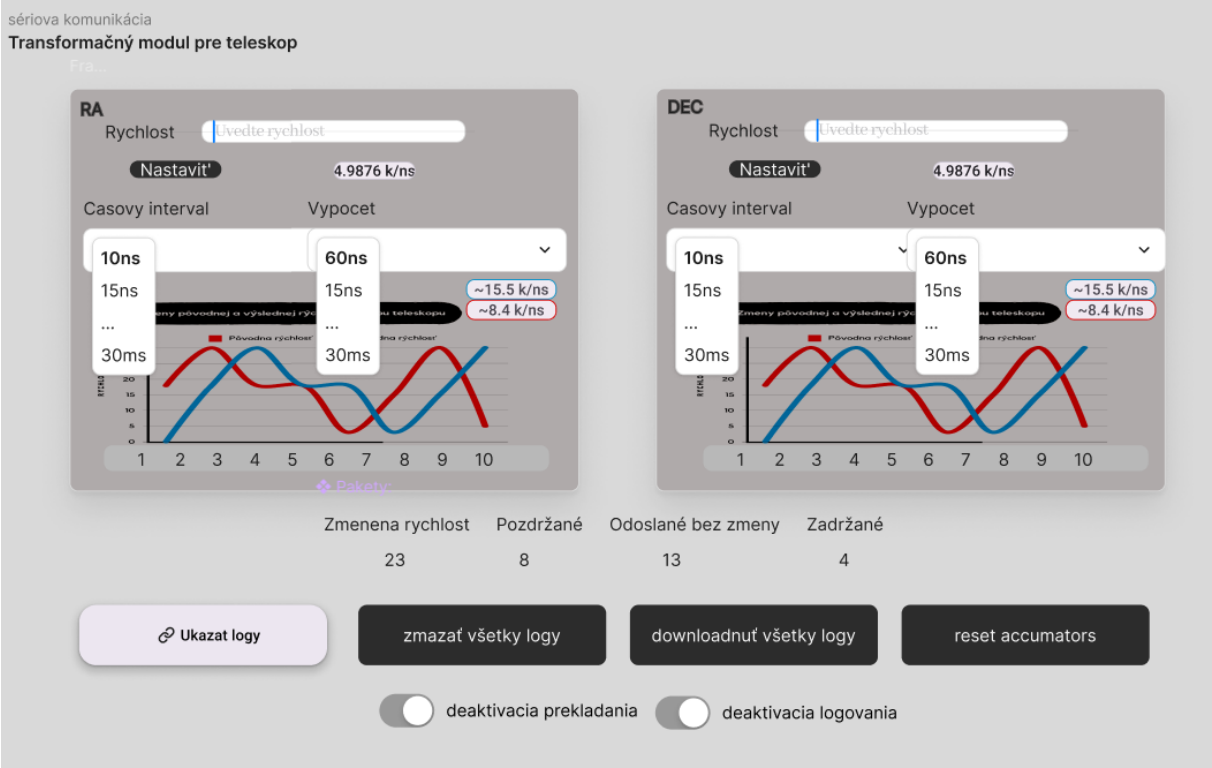
- **Formát**: TXT.
- **Obsah**: Tento log uchováva informácie o vykonaných operáciách systému, ako je aplikovanie nových pravidiel regulárnych výrazov, zmeny konfigurácie, a akékoľvek chyby alebo výnimky počas behu systému.
- **Štruktúra logu**:
  - **Časová pečiatka,Úroveň závažnosti,Správa,Kód chyby**

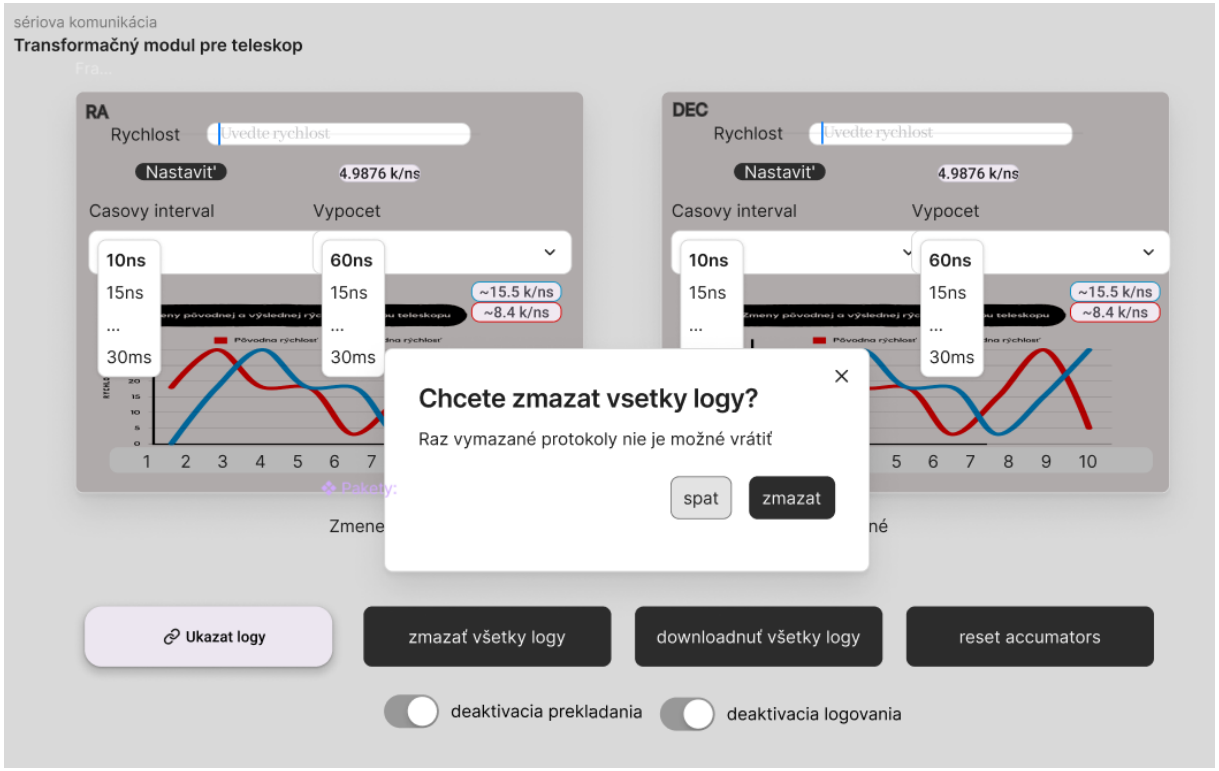
6.3. Paket-Log súbor pre univerzálny modul

- **Formát**: TXT.
- **Obsah**: Logy obsahujú zaznamenané údaje o prichádzajúcich a odchádzajúcich paktoch, časové pečiatky, typ paketu, záznam o vykonaných transformáciách.
- **Štruktúra logu**:
- **Časová pečiatka,Čas zdržania,Originálny paket,Transformovaný paket, Aplikované regex pravidlo** .

7. Návrh používateľského rozhrania

7.1. Pre transformačný modul pre teleskop

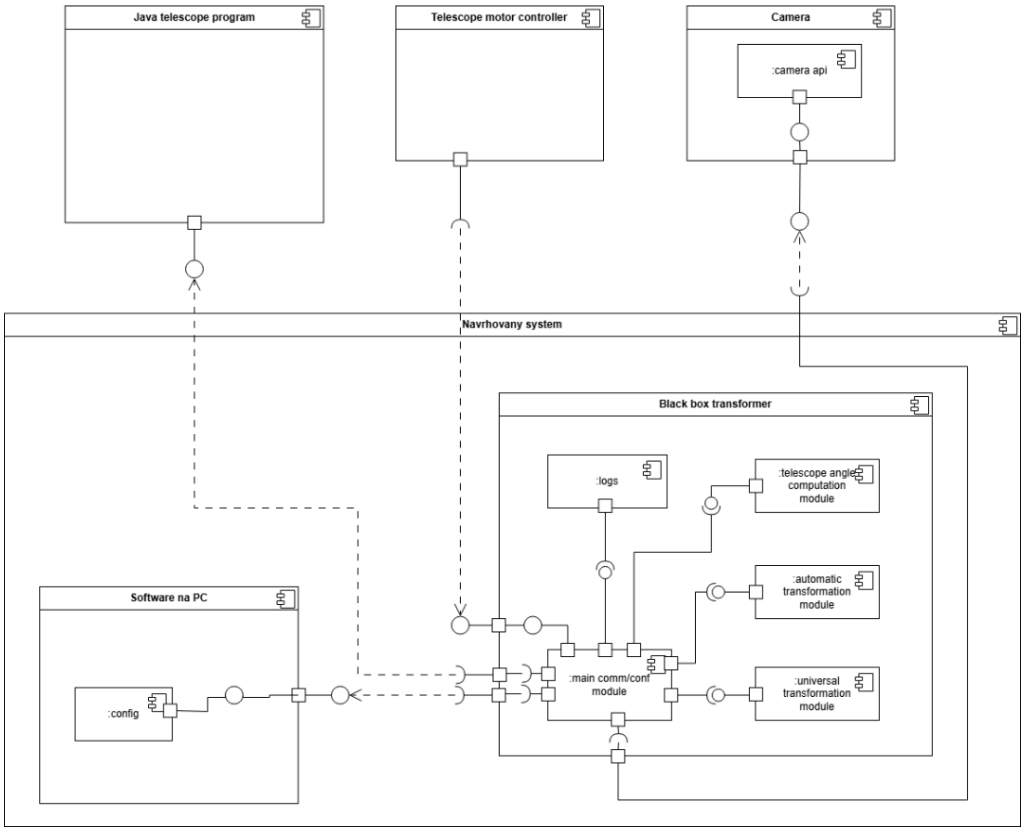




7.2. Pre univerzálny transformačný modul

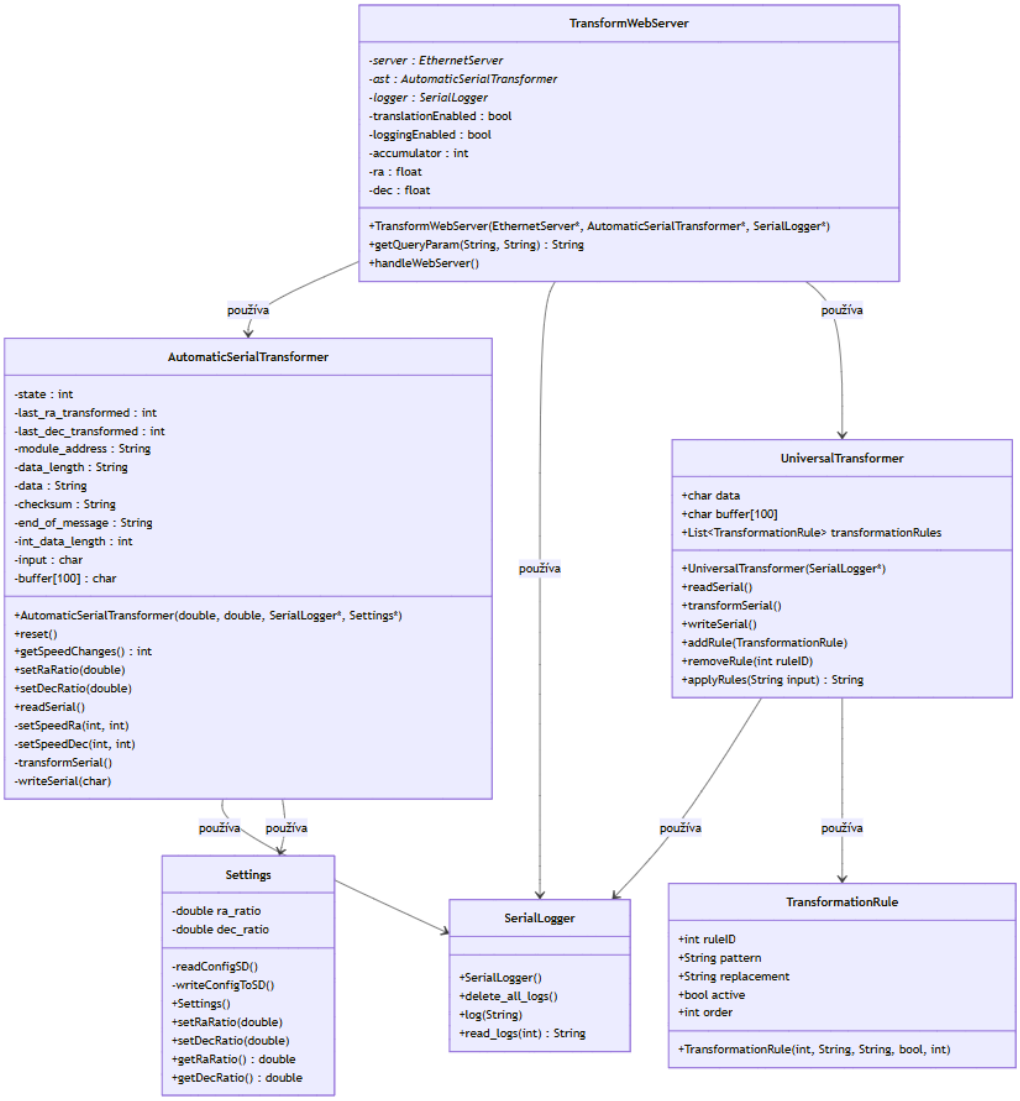
8. Návrh implementácie

8.1. Diagram komponentov



- Java telescope program - externý softvérový modul, ktorý komunikuje so systémom.
- Telescope motor controller - riadi pohyb teleskopu na základe prijatých príkazov.
- Camera - obsahuje modul, ktorý umožňuje komunikáciu s navrhovaným systémom. Používa sa na snímanie obrazu z teleskopu.
- Software na PC - slúži na nastavovanie a správu.
- Black box transformer - obsahuje niekoľko modulov, ktoré vykonávajú rôzne výpočty a transformácie: telescope angle computation module, telescope angle computation module, universal transformation module, logs, main comm/conf module

8.2. Triedny diagram



- TransformWebServer – Poskytuje webové rozhranie na komunikáciu so systémom. Využíva: UniversalTransformer na univerzálne transformácie dát, SerialLogger na logovanie.
- AutomaticSerialTransformer – Spracováva a transformuje sériové dáta. Využíva: SerialLogger na záznam udalostí, Settings na čítanie a ukladanie konfigurácie.
- UniversalTransformer – Poskytuje univerzálnu transformáciu dát, pričom: Využíva SerialLogger na logovanie, Uchováva a spracováva zoznam pravidiel TransformationRule
- TransformationRule – Definuje jednotlivé pravidlá, ktoré UniversalTransformer aplikuje na dáta

- Settings – Uchováva konfiguračné údaje a dokáže ich načítať alebo uložiť na SD kartu.
- SerialLogger – Zabezpečuje logovanie (zápis do súboru, čítanie logov, mazanie logov) a využíva sa všade tam, kde je potrebné zaznamenávať udalosti (AutomaticSerialTransformer, UniversalTransformer, TransformWebServer).

8.3. Rozdelenie na časti (moduly) a ich interfejsy

-
```
class TransformWebServer {
public:
    // Konštruktor
    TransformWebServer(EthernetServer* ptr_server,
AutomaticSerialTransformer* ptr_ast, SerialLogger* ptr_logger) {
}
    // Extrahuje parameter z HTTP requestu
    String getQueryParam(String request, String param) { }
    // Spracováva prichádzajúce HTTP požiadavky
    void handleWebServer() { }
};
```

-
```
class UniversalTransformer {
public:
   // Konštruktor
   UniversalTransformer() { }
   // Číta dáta zo sériového portu
   void readSerial() { }
   // Vykoná transformáciu dát
   void transformSerial() { }
   // Zapíše dáta na sériový port
   void writeSerial() { }
};
```

```
class Settings {
public:
  // Konštruktor - načíta konfiguráciu zo SD karty
  Settings() { }
  // Nastaví a uloží RA pomer
  void setRaRatio(double ra) { }
  // Nastaví a uloží DEC pomer
  void setDecRatio(double dec) { }
  // Vráti aktuálny RA pomer
  double getRaRatio() { }
  // Vráti aktuálny DEC pomer
  double getDecRatio() { }
};
```

```
class SerialLogger {
public:
  // Konštruktor
  SerialLogger() { }
  // Vymaže všetky logy
  void delete_all_logs() { }
  // Zapíše správu do logu
  void log(String message) { }
  // Vráti obsah logu
  String read_logs(int amount) { }
};
```

```
class AutomaticSerialTransformer {

public:

   // Konštruktor

   AutomaticSerialTransformer(double ra, double dec, SerialLogger*
ptr_logger, Settings* ptr_settings) { }

   // Resetuje interné premenné

   void reset() { }

   // Vráti počet zmien rýchlosti

   int getSpeedChanges() { }

   // Nastaví RA pomer a aktualizuje logy

   void setRaRatio(double ra) { }

   // Nastaví DEC pomer a aktualizuje logy

   void setDecRatio(double dec) { }

   // Číta dáta zo sériového portu

   void readSerial() { }

};
```

8.4. Využité technológie

• **Arduino/C++ framework** – pre programovanie mikrokontrolérov, kde sa využíva jazyk C++ a knižnice ako Arduino.h, ktoré umožňujú prácu so sériovou komunikáciou, časom, a pod.

• **Teensy 4.1** – ako hlavný mikrokontrolér, ktorý poskytuje dostatočný výkon a periférie.

• **Serial komunikácia** – na odosielanie a prijímanie dát medzi modulmi.

• **Ethernet a QNEthernet knižnica** – pre implementáciu webového servera, ktorý spracováva HTTP požiadavky a umožňuje interakciu so systémom cez webové rozhranie.

• **SPI a SD knižnica** – na prácu so SD kartou, ktorá slúži na ukladanie konfiguračných údajov a logov.

9. Plán implementácie

1. Fáza – Základné moduly a infraštruktúra
   • Najprv sa implementujú základné triedy, ako napríklad SerialLogger a Settings, ktoré sú kľúčové pre logovanie a správu konfigurácie.
2. Fáza – Implementácia transformačných modulov
   • V ďalšej fáze sa vyvíjajú transformačné moduly – AutomaticSerialTransformer a UniversalTransformer.
3. Fáza – Vývoj webového rozhrania
   • Po overení funkčnosti základných a transformačných modulov sa implementuje TransformWebServer, ktorý poskytuje komunikáciu cez HTTP/HTTPS a zabezpečuje integráciu s externými zariadeniami, ako je autoguider či PC softvér.
4. Fáza – Integrácia s externými komponentmi
   • Nasleduje integrácia s externými systémami, ako sú Java telescope program, Telescope motor controller a kamera. Cieľom je zabezpečiť bezproblémovú komunikáciu medzi všetkými modulmi a overiť, že rozhrania medzi komponentmi fungujú korektne.
5. Fáza – Systémové testovanie a ladenie
   • V záverečnej fáze prebehne celková integrácia systému. Systém bude testovaný v reálnych scenároch, čo umožní identifikovať a odstrániť prípadné chyby, optimalizovať výkon a zabezpečiť finálnu spoľahlivosť celého riešenia.