MPO TRIO

Název projektu:

Nová technologie pro inteligentní plánování pohybu robotů v průmyslových procesech

(FV 20597)

Software

[O] Řídicí systém prototypu

(Implementační dokumentace)

Martin Švejda (ZČU), Arnold Jáger (ZČU), Jan Reitinger (ZČU), Martin Goubej (ZČU), Ondřej Severa (ZČU)

původní termín: do 28. 2. 2021

skutečný termín (prodloužení řešení projektu[1]): 25. 11. 2021

MINISTERSTVO
PRŮMYSLU A OBCHODU

Identifikátor účastníka projektu:

ZČU: Západočeská univerzita v Plzni, LT: LaserTherm spol. s r.o.

[1] Řešení projektu prodlouženo do konce roku 2021 (dodatky 1) a 2) součástí přílohy roční zprávy)
1) D O D A T E K č. 1/2021 ke Smlouvě č. FV20597 o poskytnutí účelové podpory na řešení projektu formou dotace z výdajů státního rozpočtu na výzkum, vývoj a inovace
2) DODATEK č. 2 ke Smlouvě o účasti na řešení projektu ze dne 29.11.2016.

Obsah

Popis úkolu z přihlášky projektu

Zpráva bude obsahovat ucelený popis kompletního řídicího systému vyvinutého záznamového zařízení, tzn. integrované dříve vyvinuté celky (zpracování dat, interface atd.). Součástí zprávy bude nejen popis HW vybavení (řídicí počítač, zařízení pro sběr senzorických dat), ale zároveň i příslušné SW komponenty (knihovny funkcí/funkčních bloků) tvořící jádro řídicího systému prototypu.

Úvod

Předložená technická zpráva dále rozšiřuje výstupy [2, 7] a zahrnuje především implementační poznámky a popis funkcí SW vyvinutého trekovacího zařízení, a to z pohledu použitých algoritmů a metod. Dále jsou popsány výsledné HW komponenty trekovacího zařízení, které byly finálně použity k realizaci prototypu, a to především z hlediska senzoriky, řídicího počítače a jejich vzájemného propojení.
Řídící systém prototypu byl navržen na základě výstupů dílčích výzkumných a vývojových aktivit probíhající v letech 2017 - 2021, které byly dále zpřesněny a doimplementovány. Hlavní komponenty řídicího systému lze popsat následovně:

1. **HW vybavení** - senzorika, řídicí průmyslové PC, rozšiřující vstupně/výstupní periferie, komunikační rozhraní - založeno na aktivitách popsaných v [1, 2]
2. **SW vybavení** - v podobě metod a algoritmů implementovaných v řídicím systému trekovacího zařízení včetně interfacu s průmyslovým robotem, operátorské ovládání/vizualizace - založeno na [2, 3, 4, 5, 6].

Technická zpráva představuje detailní technické řešení řídicího systému trekovacího zařízení, jehož uživatelská dokumentace je předložena v [8].

HW vybavení

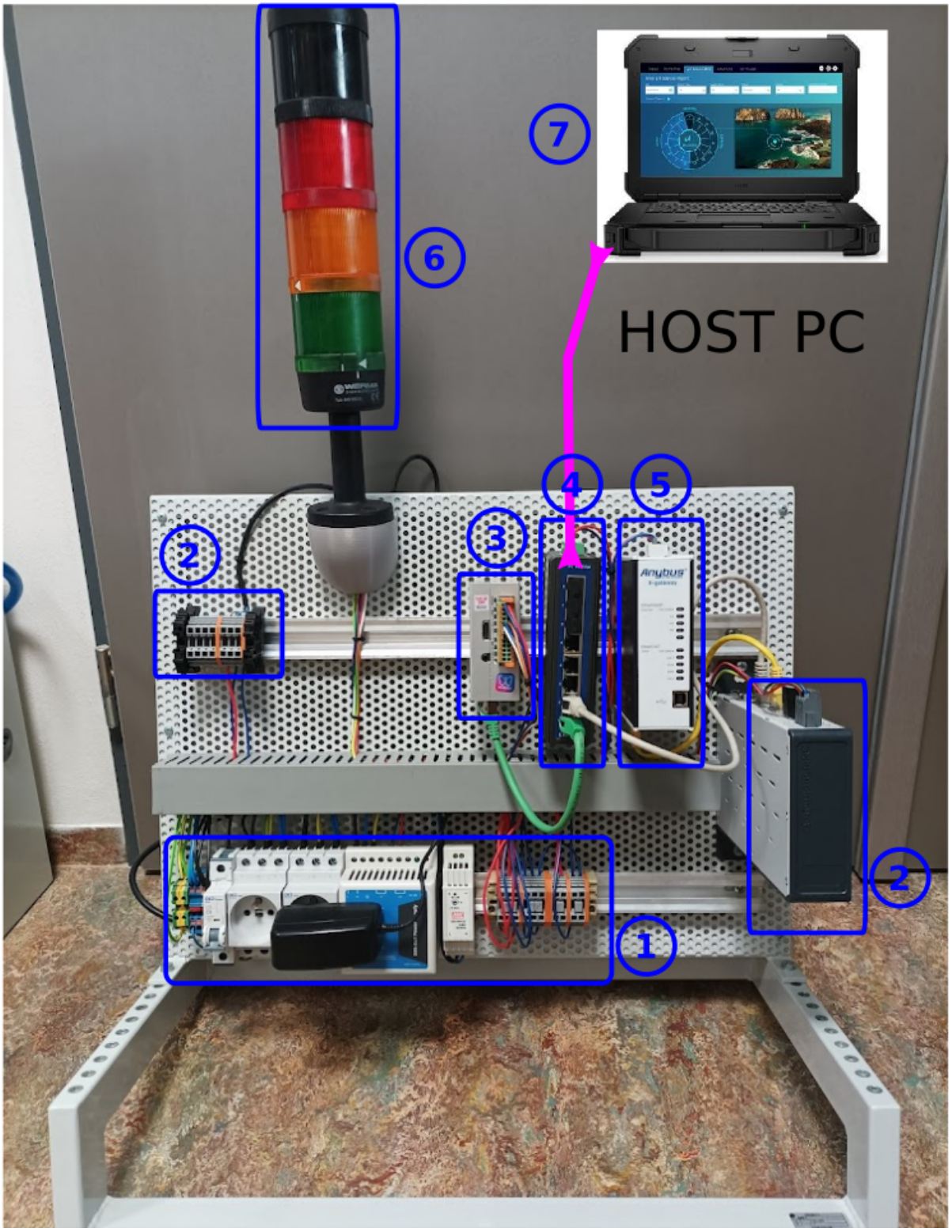
HW vybavení trekovacího zařízení se sestává z kontroleru, viz Obrázek 1, a samotného trekovacího zařízení, viz Obrázek 2, určeného jak k montáži na pracovní nástroj operátora (v našem případě stříkací pistoli), tak k montáži na samotný robot. Takové uspořádání vede na klíčovou výhodu z hlediska možnosti použití ruční stříkací pistole jak pro samotné učení pohybu operátorem, tak na automatický lakovací proces s robotem.

Kontroler trekovacího zařízení je složen z následujících komponent:

1. **Napájecí rozvod, konektorová část** - Napájení a jištění 230V AC, napájení 24V DC (řídicí počítač, průmyslový switch, EtherCAT-EthernetIP gateway, Monarco PLC), napájení 7,5V DC, napájení 5V DC (napájení servopohonu pro ovládání spouště stříkací pistole)
2. **Řídicí počítač (TARGET)** je realizován v podobě průmyslového PC od výrobce B&R, model Automation PC 2200[2] s operačním systémem Linux Debian (Buster).
3. **Monarco HAT[3] (TARGET)** je HW nadstavba kompaktního počítače **Raspberry Pi**, která umožňuje realizovat funkce standardního průmyslového PLC. Zde je využívána jako vzdálené digitální vstupy/výstupy a PWM výstupy pro řízení serva ovládání spouště stříkací pistole, řízení signálního sloupku.
4. **Průmyslový switch Advantech EKI-2528** - standardní switch pro rozvod interní TCP/IP sítě pro datovou komunikaci mezi komponentami.
5. **Gateway Anybus X-gateway** - převod mezi komunikačním protokolem EtherCAT (komunikace na straně řídicího algoritmu trekovacího zařízení) a EthernetIP (standardní komunikační rozhraní s roboty Fanuc).

[2] https://www.br-automation.com/en/products/industrial-pcs/automation-pc-2200/
[3] https://www.monarco.io/

6. **Signální sloupek** - indikace stavů trekovacího zařízení v procesu učení pohybu
7. **HOST počítač -** standardní PC/notebook s instalovaným prostředím Matlab, na kterém běží vzdálená aplikace datového postprocesoru. Na tomto stroji může být zároveň zobrazen i operátorský panel (HMI).



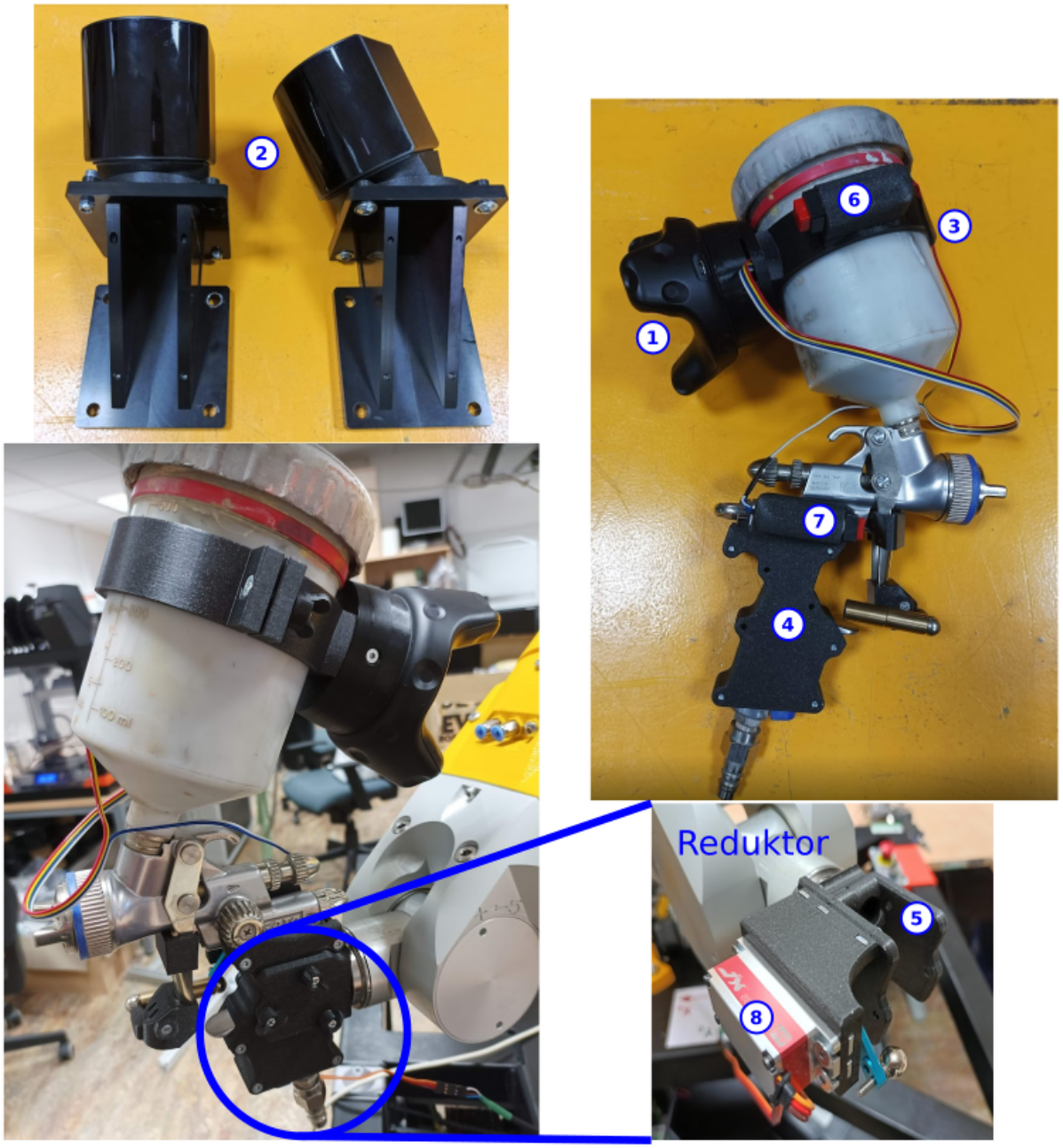
Obrázek 1: Finální realizace kontroleru prototypu trekovacího zařízení

Trekovací zařízení je složeno z následujících komponent:

1. **HTC Vive Tracker**[4] - systém využívaný v herním průmyslu pro virtuální realitu, systém poskytuje možnosti integrace do řídicích systémů dalších stran[5]. Komunikační protokol se systémem HTC Vive Tracker je dále popsán v SW modul interfacu s HTC Vive Trackerem. HTC Vive tracker představuje polohový senzor trekovacího zařízení[6].
2. **Bázové (základní stanice)** - Základní stanice SteamVR Base Station 2.0[7] snímající pohyb HTC Vive Trackeru. Možnost osadit 1-4 základní stanice.
3. **Konzole k ukotvení HTC Vive Trackeru** - slouží k uchycení trackeru k lakovací pistoli v různé poloze, umožňuje sejmutí samotného trackeru.
4. **Konzole uchycení lakovací pistole** - slouží k uchycení lakovací pistole k reduktoru na přírubě robotu. Zajišťuje snadné sejmutí a znovunasazení pistole mezi procesem učení a automatické replikace pohybu.
5. **Reduktor** - slouží jako redukční přechod mezi přírubou robotu a konzolí lakovací pistole.
6. **Funkční tlačítko** - slouží k ovládání procesu učení (inicializace, start/stop, reset)
7. **Snímač stisku spouště lakovací pistole** - pro záznam aktivace stříkání v procesu učení operátorem
8. **Ovládací servo spouště** - ovládání spouště stříkací pistole při automatické replikaci pohybu včetně magnetického systému propojení spouště stříkací pistole s výstupní přírubou serva umístěného na reduktoru (snadné odpojení/připojení serva).

Detailní informace k HW vybavení včetně výkresové dokumentace a výsledků testování prototypu lze nalézt v [9].

[4] https://www.vive.com/us/accessory/vive-tracker/
[5] https://developer.vive.com/us/hardware/tracker3/
[6] Jako senzor trekovacího zařízení byl využit pouze HTC Vive Tracker, jednotka IMU BNO055 nakonec použita nebyla, neboť již nepřinesla žádné zlepšení.
[7] https://www.vive.com/us/accessory/base-station2/

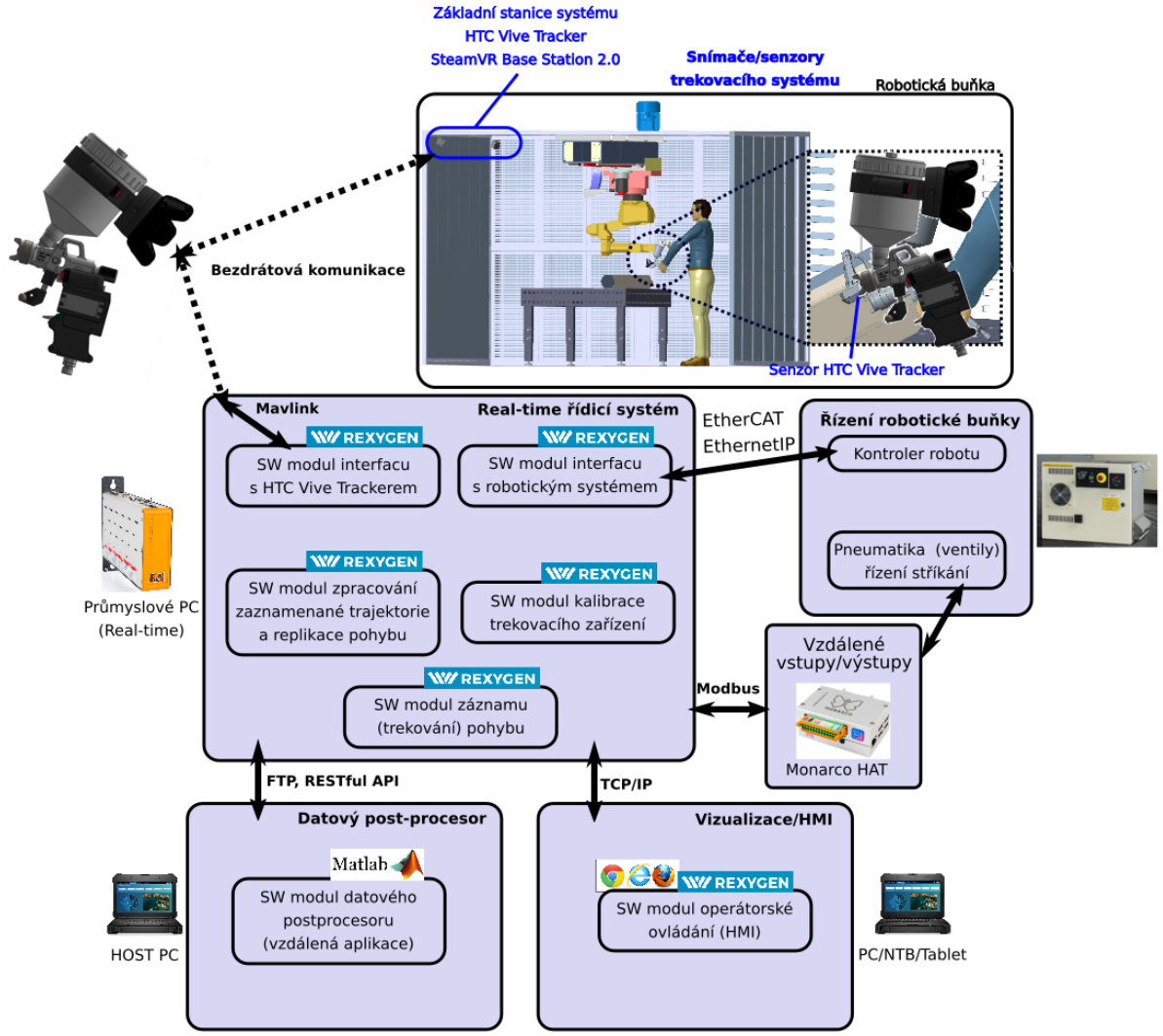


Obrázek 2: Trekovací zařízení připevněné na (ruční) stříkací pistoli

SW vybavení

Kompletní řídicí systém je koncipován v podobě tří konfigurací, viz Obrázek 3:

1. **Konfigurace TRACKER:** Sestava dílčích SW modulů, které jsou implementovány v řídicím systému reálného času REXYGEN a realizují hlavní (vlastní) řídicí systém trekovacího zařízení. Konfigurace běží na TARGET počítači (průmyslovém řídicím počítači, viz HW vybavení).
2. **Konfigurace MonarcoInterface:** Sestava dílčích SW modulů, které jsou implementovány v řídicím systému reálného času REXYGEN a realizují pomocný řídicí systém ovládání vzdálených vstupů/výstupů. Konfigurace běží na TARGET počítači (MonarcoHAT), viz HW vybavení.
3. **Konfigurace MATLAB:** SW vzdálené aplikace, která je implementována v prostředí Matlab a realizuje **datový postprocesor** pro zpracování nasnímaných dat. Konfigurace běžící na HOST počítači, viz HW vybavení.



Obrázek 3: Řídicí systém trekovacího zařízení včetně komunikačního rozhraní

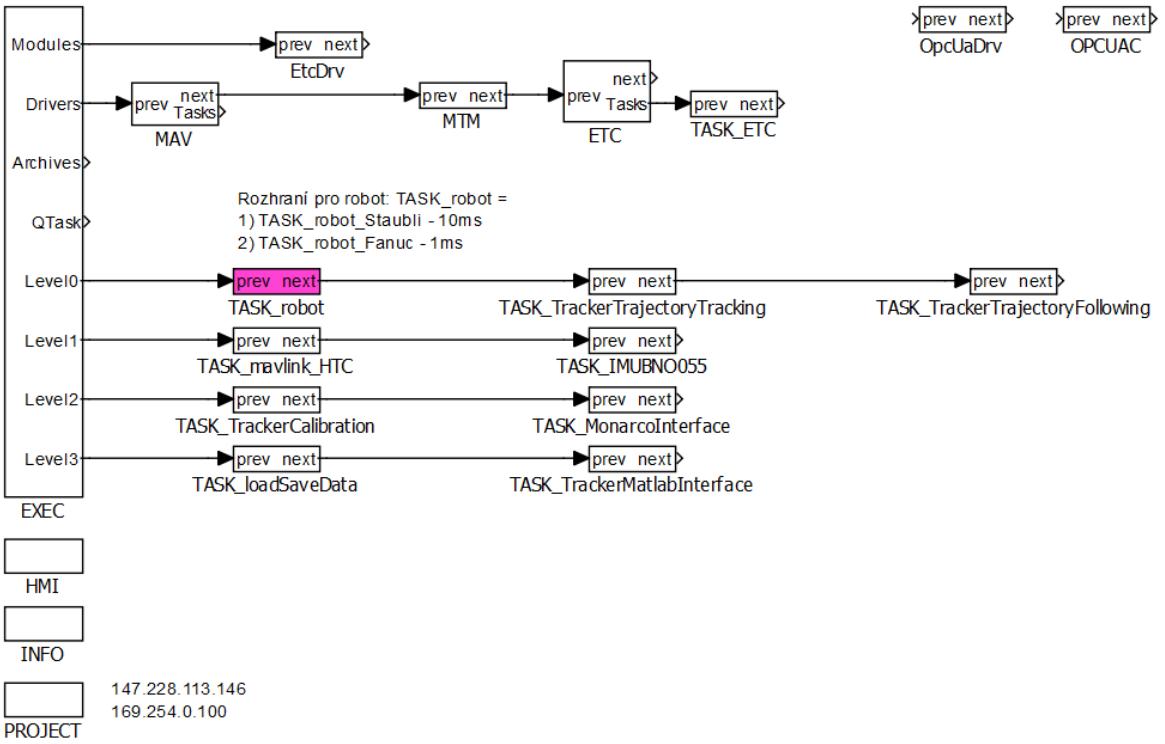
Konfigurace TRACKER

SW moduly konfigurace TRACKER implementované v řídicím systému REXYGEN jsou následující:

1. **SW modul interfacu s robotickým systémem** - finální interface pro komunikaci s cílovým robotem. V rámci projektu bylo vyvinuto nejen specifické komunikační rozhraní pro povelování robotu z nadřazeného řídicího systému s robotem Fanuc, ale dále i s roboty UR a Stäubli - založeno na aktivitách [3, 5].
2. **SW modul interfacu s HTC Vive Trackerem** - Založen na knihovně *libsurvice*[8] umožňující komunikaci s trackerem na developerské úrovni.
3. **SW modul kalibrace trekovacího zařízení** - finální podoba algoritmu, který inicializuje kalibrační funkce prostřednictvím vzdálené aplikace a kalibrační výsledky zpětně interpretuje v řídicím systému trekovacího zařízení - založeno na aktivitách [2].
4. **SW modul záznamu (trekování) pohybu pracovního nástroje vedeného operátorem** - finální podoba algoritmizace procesu trekování (vlastní záznam pohybu) včetně procesu referencování trekovacího zařízení - založeno na aktivitách [2].
5. **SW modul zpracování zaznamenané trajektorie a replikace pohybu** - finální podoba algoritmu, který inicializuje zpracování nasnímaných dat prostřednictvím vzdálené aplikace a zpracovaná data zpětně interpretuje v řídicím systému trackeru za účelem replikace výsledného pohybu robotem - založeno na [3, 4, 5, 6].
6. **SW modul operátorské ovládání (HMI)** - operátorský panel včetně vizualizace stavu řídicího systému trekovacího zařízení a robotu - založeno na [6].

SW moduly jsou implementovány v podobě řídicích TASKů v prostředí řídicího systému reálného času REXYGEN, viz dokumentace[9]. Hlavní řídicí TASK *exec* sdružující všechny dílčí TASKy je znázorněn na Obrázku 4.

[8] https://github.com/cntools/libsurvive/blob/master/README.md
[9] https://www.rexygen.com/doc/index.html



Obrázek 4: Hlavní TASK řídicího systému trekovacího zařízení (TRACKER)

Dílčí tasky příslušející SW modulům:

1. **SW modul interfacu s robotickým systémem:**
   a. **TASK_robot:** Implementace interfacu s robotickým systémem, finálně realizovány a testovány dvě rozhraní komunikace pro roboty:
      i. Stäubli: Komunikace (povelování) přes OPC UA[10]
      ii. Fanuc: Komunikace (povelování) přes EtherCAT[11] => gateway => EthernetIP[12]

      Slouží primárně k integraci systému trekovacího zařízení do robotických systému průmyslových dodavatelů.
   b. **TASK_ETC:** Součástí interfacu s robotem Fanuc (část na straně řídicího systému trekovacího zařízení), komunikace dat pro gateway EtherCAT-EthernetIP, interpretace nasnímaných a zpracovaných dat pro použitou funkci (option) DPM (Dynamic Path Modifier) v řídicím systému robotu Fanuc.
2. **SW modul interfacu s HTC Vive Trackerem**
   a. **TASK_mavlink_HTC:** Komunikační rozhraní přes protokol MAVlink[13], který prostřednictvím knihovny *libsurvive* zajišťuje přenos snímaných dat HTC Vive Trackerem do řídicího systému.

[10] https://opcfoundation.org/about/opc-technologies/opc-ua/, https://www.rexygen.com/doc/ENGLISH/MANUALS/RexOpcUa/RexOpcUa_ENG.html

[11] https://www.beckhoff.com/en-en/products/i-o/ethercat/?pk_campaign=AdWords-AdWordsSearch-EtherCatEN&pk_kwd=ethercat&gclid=Cj0KCQiA5OuNBhCRARIsACgaiqU5FnKq0UqeGbEomXam_-71qUZJh3TAMJSV7nH3-Ob8flekB3e2dRsaAjWNEALw_wcB

[12] https://cs.wikipedia.org/wiki/Ethernet/IP

[13] https://mavlink.io/en/

3. **SW modul kalibrace trekovacího zařízení:**
    a. **TASK_TrackerCalibration:** Implementuje proces předzpracování a záznamu dat během kalibrace trekovacího zařízení s robotickým systémem. Dále zajišťuje administraci celého procesu kalibrace (přejezdy robotu do definovaných pozic, uložení polohových a zaznamenaných dat, atd.). Zprostředkovává spouštění kalibračního algoritmu v datovém postprocesoru.
4. **SW modul záznamu (trekování) pohybu pracovního nástroje vedeného operátorem:**
    a. **TASK_TrackerTrajectoryTracking:** Zajišťuje proces záznamu nasnímaných dat pro pozdější zpracování. Zároveň je zde vyřešen proces referencování trekovacího zařízení před každým zahájením záznamu pohybu.
5. **SW modul zpracování zaznamenané trajektorie a replikace pohybu:**
    a. **TASK_TrackerTrajectoryFollowing:** Zajišťuje spouštění/replikaci zpracované pohybové trajektorie. Zprostředkovává spouštění algoritmu generování trajektorie v datovém postprocesoru.
6. **SW modul operátorské ovládání (HMI)**
    a. Implementovaný v prostředí InkScape s rozšířením pro tvorbu vizualizací a napojení na signály v řídicím systému REXYGEN (HMI Designer[14])
7. **Ostatní podpůrné TASKy a samostatné funkční bloky:**
    Dále podrobněji nedokumentované pomocné tasky, které slouží především k záznamu datových bodů a komunikaci s periferními zařízeními (ovládání serva, digitální vstupy/výstupy).
    a. **TASK_IMUBNO055:** Komunikační rozhraní s inerciální jednotkou, nakonec nevyužito.
    b. **TASK_MonarcoInterface:** Komunikační s Monarco HAT - realizace PWM řízení serva a vzdálené digitální vstupy/výstupy.
    c. **TASK_loadSaveData:** Implementace ukládání zaznamenaných dat do paměti řídicího počítače.
    d. **TASK_TrackerMatlabInterface:** Komunikační rozhraní využívající protokol RESTful API k zápisu a čtení signálů z datového postprocesoru (vzdálená aplikace implementovaná v Matlabu)
    e. **EtcDrv, MAV, MTM, ETC, OPcUaDrv, OPCUAC:** Funkční bloky driverů použitých komunikačních protokolů: EtherCAT (robot Fanuc), MAvlink (HTC Vive Tracker), Modbus (MonarcoHAT), OPC UA (robot Stäubli).

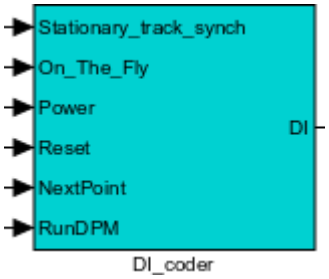
SW modul interfacu s robotem Fanuc

Ve zprávách [3, 5] byla ke komunikaci s robotem Fanuc používána funkce kontroleru *HMI DEVICE COMMUNICATION* s předpřipraveným rozhraním Modbus TCP. Komunikace se simulovaným robotem probíhala bez problémů a podle představ. Během testování na reálném zařízení se však zjistilo, že tato volba není pro účely realtime komunikace vhodná, neboť docházelo ke zpožděním a tím pádem i odchylkám v pohybu robotu. Proto byla komunikace implementována pomocí protokolu EtherCAT na straně trekovacího zařízení

[14] https://www.rexygen.com/doc/ENGLISH/MANUALS/RexHMI/RexHMI_ENG.html

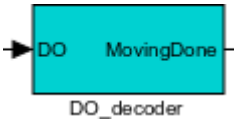
(REXYGEN), protokolu EthernetIP na straně robotu a převodníku EtherCAT-EthernetIP. Tato komunikace již byla dostatečně stabilní a s požadovanou latencí.
Data jsou komunikována jako série 16-bitových celých čísel. V popisu dále jsou uvedeny binární a reálné signály. Tyto signály jsou vnitřně na obou stranách komunikace převáděny na celá čísla. Způsob převodu a optimalizace je vždy uveden u daného typu signálu. Výsledná koncepce SW interfacu s roboty Fanuc byla tedy realizována v následující podobě:

1. Komunikace na straně řídicího systému trekovacího zařízení (REXYGEN)
    - **Binární vstupy** do robotu Fanuc pro řízení chodu programu (režimy kalibrace, trekování, replikace pohybu). Tyto vstupy jsou blokem *DI_coder* (na Obrázku 5) převáděny na jednu celočíselnou hodnotu, která je následně komunikována po EtherCAT. Je možné použít až 16 binárních vstupů. Momentálně jsou definovány následující:
        - Stationary_track_synch - hodnota ON ukončí Stationary Track (sledování offsetu i na konci pohybu)
        - On_The_Fly - hodnota ON přeruší činnost DPM, např. kvůli dosažení max. odchylky
        - Power - spuštění běhu programu v kontroleru
        - Reset - znovuspuštění programu
        - NextPoint - robot přejede do další polohy (v kalibračním módu) nebo do počátečního bodu pro replikaci pohybu (ve sledovacím módu)
        - RunDPM - spuštění replikace pohybu



Obrázek 5: Funkční blok pro převod binárních vstupů na celé číslo

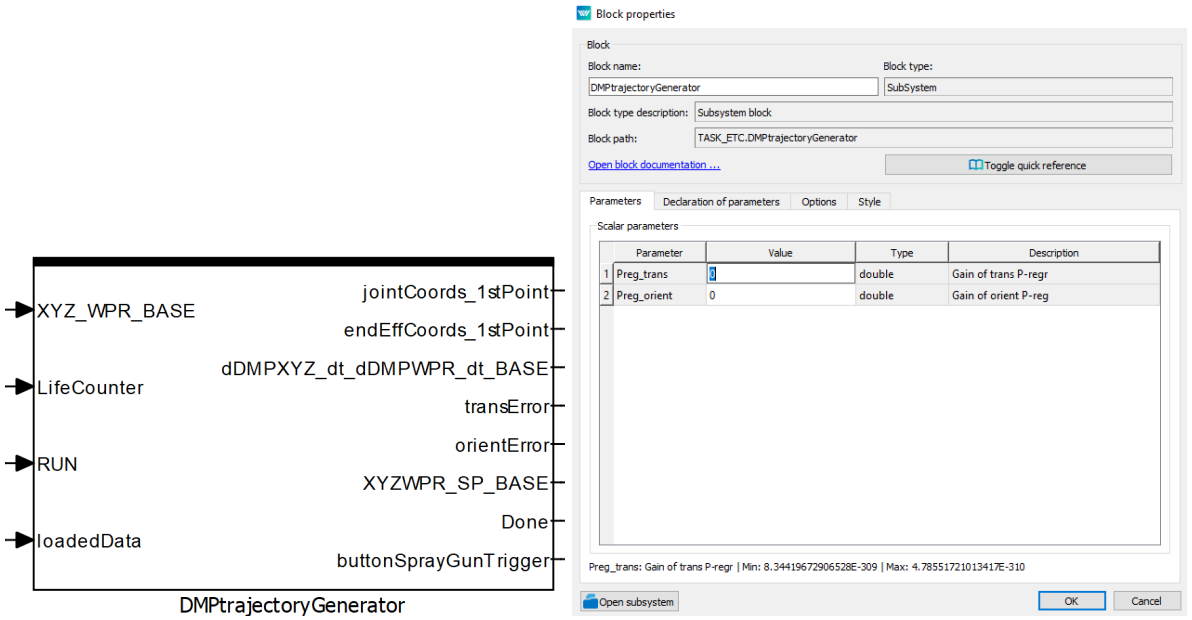
- **Binární výstupy** robotu (kontroleru) přicházejí jako jedno celé číslo a jsou dekódovány do binární podoby blokem *DO_decoder* (viz Obrázek 6). Momentálně je používán jeden binární výstup, nicméně v případě potřeby je možné doplnit další. Lze definovat až 16 výstupů. Aktuálně je používán tento:
    - MovingDone - robot dojel do požadovaného bodu při kalibraci a nebo výchozího bodu pro replikaci pohybu



Obrázek 6: Funkční blok pro převod celého čísla na binární výstupy

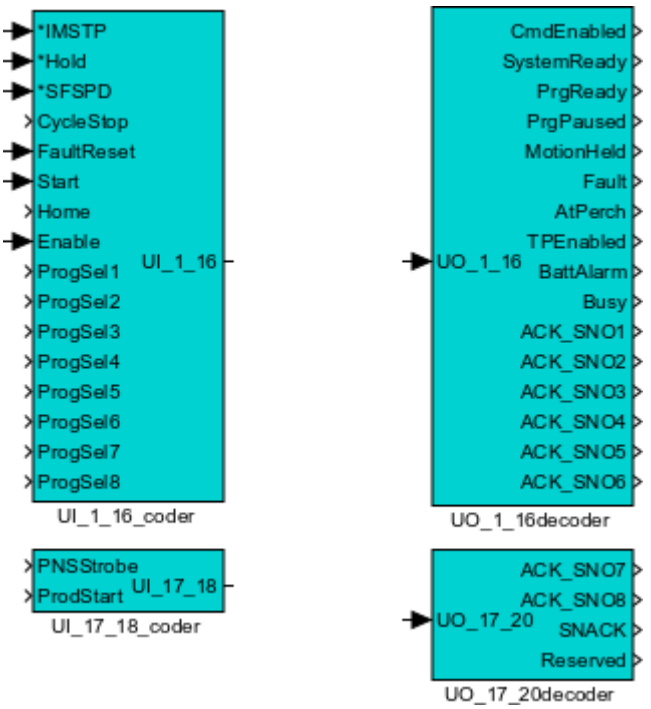
- **Celočíselné vstupy** jsou definovány takto:
    - Control - zapínání a vypínání Ethercat komunikace
    - Robot_mode - přepíná mezi jednotlivými módy programu (kalibrační, trekovací, replikační)
- Z robotu Fanuc je komunikováno několik **celočíselných výstupů**, které mají následující význam:
    - Status - stav komunikace
    - Counter - life counter, které je v každém cyklu inkrementován o 1

    - Override - aktuální rychlost robotu vyjádřena v procentech
- **Reálné vstupy k polohování os robotu** (pro účely přejezdů během kalibrace) jsou určeny pro komunikování pozice robotu, do které se má přesunout. Pozice je dána souřadnicemi X, Y, Z, W, P, R. Každá hodnota je komunikována pomocí dvou 16-bitových celých čísel. První číslo udává hodnotu před desetinnou čárkou, druhé číslo reprezentuje neceločíselnou část zaokrouhlenou na tisíciny. Tyto hodnoty jsou komunikovány dokud nedojde k zapnutí DPM. Poté jsou místo pozic posílány požadované rychlosti k replikaci výsledného pohybu..
    - POS_X_u - souřadnice X
    - POS_Y_u - souřadnice Y
    - POS_Z_u - souřadnice Z
    - POS_W_u - souřadnice W
    - POS_P_u - souřadnice P
    - POS_R_u - souřadnice R
- **Reálné vstupy k replikaci výsledného pohybu** jsou generovány implementovaným funkčním blokem *DMPtrajectoryGenerator*, viz Obrázek 7, který generoval data pro funkci Fanuc DPM, vyhodnocoval chybu sledování replikovaného pohybu a zároveň umožňoval zavést polohovou korekční zpětnou vazbu (korekce nepřesností v časování, zaokrouhlování hodnot, atd.). Posílaná data jsou požadované rychlosti ve směrech X, Y, Z, W, P, R s přesností na jedno desetinné místo. Reálné hodnoty jsou převáděny na celé číslo prostým vynásobením deseti a zaokrouhlením. Tato data jsou komunikována pouze pokud je aktivní DPM. Jinak jsou místo nich komunikovány požadované pozice pro polohování os robotu.
    - VEL_X - rychlost ve směru X
    - VEL_Y - rychlost ve směru Y
    - VEL_Z - rychlost ve směru Z
    - VEL_W - rychlost ve směru W
    - VEL_P - rychlost ve směru P
    - VEL_R - rychlost ve směru R



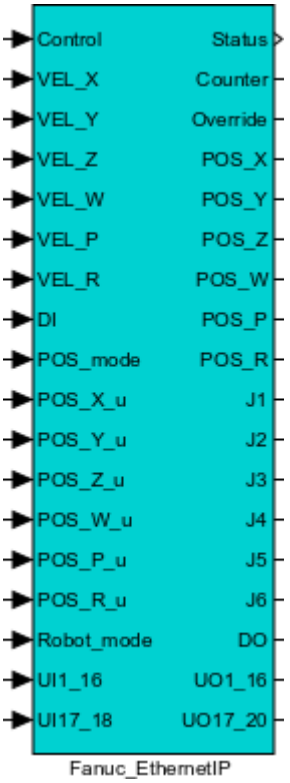
Obrázek 7: Funkční blok generování dat pro funkci Fanuc DPM

- **Reálné výstupy** slouží hlavně k uzavření zpětné vazby a je s jejich pomocí komunikována aktuální poloha robotického ramene. Pozice je udávána jak v souřadném systému X, Y, Z, W, P, R, tak v natočení jednotlivých kloubů J1-J6. Pozice X, Y a Z jsou udávány s přesností na jedno desetinné místo a jsou posílány pomocí vždy jednoho celého čísla. Ostatní pozice jsou uváděny pomocí dvou celých čísel s rozdělením podle desetinné čárky.
    - POS_X
    - POS_Y
    - POS_Z
    - POS_W
    - POS_P
    - POS_R
    - J1
    - J2
    - J3
    - J4
    - J5
    - J6
- Analogicky je do bloků implementováno komunikování signálů určených k **diagnostice** stavu robotu a jeho **vzdálenému ovládání**. K ovládání a diagnostice je využito rozhraní UOP (User Operator Panel), které není potřeba nijak modifikovat. Je potřeba pouze namapovat dané signály na používanou komunikaci. Bloky pro UOP jsou na Obrázku 8.



Obrázek 8: Bloky pro vzdálené řízení a monitorování běhu programu Fanuc

Blok určený ke komunikaci s Fanuc pomocí EtherCAT a EthernetIP je na Obrázku 9.



Obrázek 9: Funkční blok pro komunikaci s Fanuc

2. Komunikace na straně kontroleru robotu Fanuc
    - **Binární vstupy a výstupy** do robotu jsou analogické k popisu v bodu 1 výše. Je potřeba zkontrolovat jejich konfiguraci. Pro komunikaci jsou přednastaveny digitální vstupy DI[129-144] a digitální výstupy DO[129-144]. Tyto signály by měly být nastaveny na rack 89 (EthernetIP), slot 2 (pokud není nakonfigurováno jinak), start 321. Binární signály na této straně komunikace není potřeba nijak dekódovat.
    - Pro komunikování **celočíselných a reálných signálů** slouží takzvané sdružené vstupy a výstupy (group inputs and outputs). Ty je opět potřeba nakonfigurovat stejně jako binární signály. Vstupy jsou používány v rozsahu GI[10-22] a výstupy GO[10-32]. Podle požadovaného typu signálu se komunikovaná hodnota dále upravuje:
        - **Celočíselné** *kladné* hodnoty není potřeba nijak upravovat a používají se v programu buď přímo nebo se přiřazují skalárním proměnným R[-]. Všechny celočíselné signály (vstpy i výstpy) zmíněné v bodu 1 výše nabývají pouze kladných hodnot.
        - Pokud by bylo nutné komunikovat **celočíselnou** *zápornou* hodnotu, je situace složitější. Signály GI a GO jsou napevno nastaveny jako 16-bitové neznaménkové (UINT16). Je tedy potřeba vždy tyto signály přiřadit do skalárního registru R[-] a provést konverzi z rozsahu UINT16 do INT16 nebo opačně. Pro konverzi GI[-] -> R[-], tj. UINT16 -> INT16 je využíván vztah, který kladné hodnoty nechává nezměněny a záporné správně nastaví. Vztah v programovacím jazyce Fanuc může být zapsán jako:

            R[-] = (GI[-] - (GI[-] DIV 32768) * 65536).

          Pro konverzi R[-] -> GO[-], tj. INT16 -> UNIT16 zvládá Fanuc hodnoty změnit prostým přiřazením:

            GO[-] = R[-].
        - Výše zmíněný postup konverze je důležitý pro všechny přenášené **reálné signály** (*kromě* těch určených pro DPM), neboť jak bylo zmíněno v bodu 1, tyto signály jsou přenášeny jako celočíselné a do oboru reálných čísel jsou konvertovány násobením a dělením 10^x, kde x je požadovaný počet desetinných míst. Například zápis pro konverzi požadované pozice v ose X (v bodu 1 reálný vstup POS_X_u) bude vypadat v programu Fanuc takto:

            R[1] =((GI[1] - (GI[1] DIV 32768) * 65536) + (GI[2] - (GI[2] DIV 32768) * 65536) / 1000)

          za předpokladu, že na vstup GI[1] je přenášena celočíselná část hodnoty a na GI[2] její desetinný zbytek.
        - Pro **reálné vstupy** využívané v **DPM** je konverze prováděna škálováním, které je nastavené přímo v DPM (viz [3]).

SW modul interfacu s HTC Vive Trackerem

HTC Vive Tracker využívá uzavřený komunikační protokol, který využívá napojení na knihovnu SteamVR[15], která je součástí služby Steam. Použití této knihovny v robotických aplikací, které nevyžadují připojení HMD (brýlí pro virtuální realitu) je velmi komplikované. Naštěstí existuje open source knihovna *libsurvive*[16], která umožňuje přímé spojení klientských aplikací s HTC Vive trackerem a dalšími periferiemi pro potřeby virtuální reality. Libsurvive je sada nástrojů a knihoven umožňující sledování 6 DoF s pomocí základnových stanic lighthouse a pohybujících se komponent jako je Tracker, Controller, apod., Knihovna je kompletně open source a může běžet na jakémkoliv zařízení. V současné době podporuje zařízení generace SteamVR 1.0 i SteamVR 2.0 a měla by podporovat všechny komerčně dostupné sledované objekty.

Tato knihovna je implementována v jazyce C++ a zároveň obsahuje přímé propojení s jazykem Python. Pro potřeby sledování objektů v tasku řídicího systému REXYGEN bylo potřeba tyto technologie propojit. V rámci řešení projektu vznikl modul napsaný v jazyce Python, který je závislý na knihovně *libsurvive* a komunikuje s ostatními komponentami pomocí otevřeného protokolu MAVLink[17]. K tomu je využívána open source knihovna *pymavlink*[18]. Aby bylo možné zpracovávat periodicky data z trackeru, reagovat na příchozí zprávy ze sítě MAVLink a generovat příslušné informace o poloze a rotaci sledovaného objektu bylo potřeba mít v aplikaci k dispozici několik vláken, která zpracovávají data a komunikaci zároveň. K tomu se využívá knihovny *RxPy*[19]. Jedná se o knihovnu, která podporuje reaktivní programování.

SW modul je složen z hlavního tasku, který běží ideálně s periodou 250ms. Jelikož se jedná o spouštění úlohy v tzv. režimu soft-realtime, tak pravidelnost úlohy není zaručena. Nicméně u polohových zpráv se využívá principu časových značek, kde příjemce zpráv ví, jak stará data přijal.

SW modul se spouští jako služba pomocí systémového démona (*systemd*). Jakmile dojde ke spuštění služby, modul začne komunikovat na konfigurovatelném UDP portu a začne poslouchat příchozí zprávy ve formátu MAVLink a vysílat zprávu typu HeartBeat (HB - základní zpráva protokolu MAVLink). Jakmile obdrží zprávu typu HB, provede inicializaci knihovny *libsurvive* a začne periodicky posílat získaná polohová data z HTC Trackeru.

Komunikační interface je mapován na standardní zprávy protokolu MAVLink a to následovně:

VISION_POSITION_ESTIMATE[20], kde *x,y,z* je poloha prvního detekovaného trackeru a první čtyři hodnoty v poli *covariance* představují hodnoty kvaternionu rotace trackeru.

---

[15] https://store.steampowered.com/app/250820/SteamVR/
[16] https://github.com/cntools/libsurvive
[17] https://mavlink.io/en/
[18] https://github.com/ArduPilot/pymavlink
[19] https://github.com/ReactiveX/RxPY
[20] https://mavlink.io/en/messages/common.html#VISION_POSITION_ESTIMATE

MANUAL_CONTROL[21] v proměnné *buttons* se zasílá informace o stisknutí tlačítka SYSTEM, TRIGGER, TRACKPAD (využívána dále např. k detekci spouště stříkací pistole operátorem)

U aplikace lze dále pomocí zpráv typu COMMAND_LONG zapnout logování do systémového logu, zasílání debuglogů v textové podobě a změnu periody čtení polohových dat.

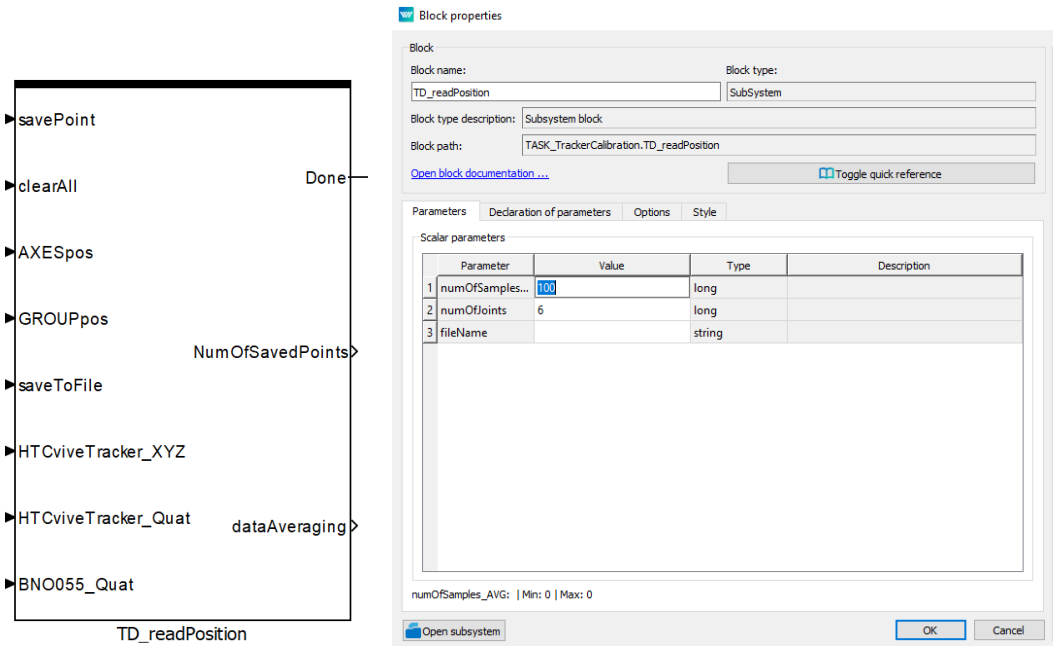
Data ze služby zpracovává task v řídicím systému TASK_mavlink_HTC, který využívá driver MAVLinkDrv pro komunikaci protokolem MAVLink, který je součástí systému REXYGEN.

Jelikož služba běží na stejném řídicím počítači v samostatném procesu, je možné ji nezávisle na běhu řídicího systému restartovat pomocí standardních příkazů pro služby déma *systemd.*

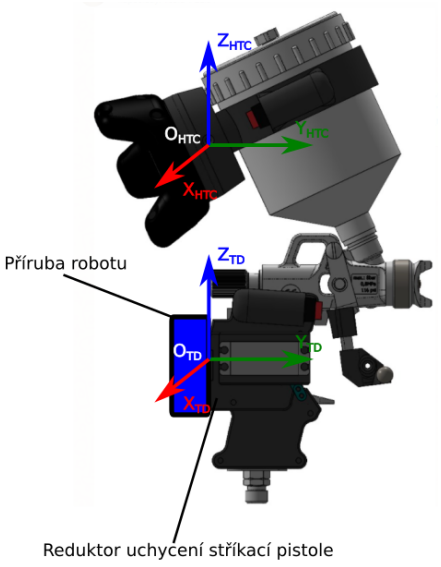
SW modul kalibrace trekovacího zařízení

Kalibrace trekovacího zařízení je klíčová úloha sloužící k identifikace polohy souřadného systému (s.s.) trekovacího zařízení $F_{TD} = O_{TD} - x_{TD}, y_{TD}, z_{TD}$, který je umístěn souhlasně se s.s. příruby robotu a s.s. HTC Vive Trackeru $F_{HTC} = O_{HTC} - x_{HTC}, y_{HTC}, z_{HTC}$, viz Obrázek 11, jehož poloha je měřena bázovými stanicemi, viz HW vybavení. Výsledná nalezená transformace mezi s.s. F_{TD} a F_{HTC} pak umožňuje replikovat zaznamenaná a zpracovaná data z procesu učení použitým robotem identicky, neboť měřená polohová data jsou vždy přepočtena (po procesu referencování, viz SW modul záznamu (trekování) pohybu pracovního nástroje vedeného operátorem) z původního s.s. trackeru do s.s. příruby robotu (= s.s. trekovacího zařízení), jejíž poloha je řízena robotem. Jinými slovy, pokud známe (měříme) pohyb s.s. trackeru F_{HTC} a zároveň známe jeho umístění vzhledem k s.s. příruby robotu F_{TD}, lze vypočítat odpovídající pohyb příruby robotu, který replikuje pohyb trackeru.
Proces kalibrace je realizován postupným najetím do definovaných polohových bodů, záznamem výstupu trackeru, zprůměrováním dat a výpočtem výsledné transformace v datovém postprocesoru. Algoritmy výpočtů kalibrací jsou detailně popsány v [1], kde byl původní OTUS tracker nahrazen HTC Vive Trackerem.
Záznam kalibračních poloh a měřených polohových dat včetně jejich průměrování byl zajištěn nově implementovaným funkčním blokem *TD_readPositon*, viz Obrázek 10.
Chod kalibračního algoritmu byl řízen standardním stavovým automatem (blok *ATMT* v systému REXYGEN).

[21] https://mavlink.io/en/messages/common.html#MANUAL_CONTROL



Obrázek 10: Funkční blok pro záznam kalibračních dat

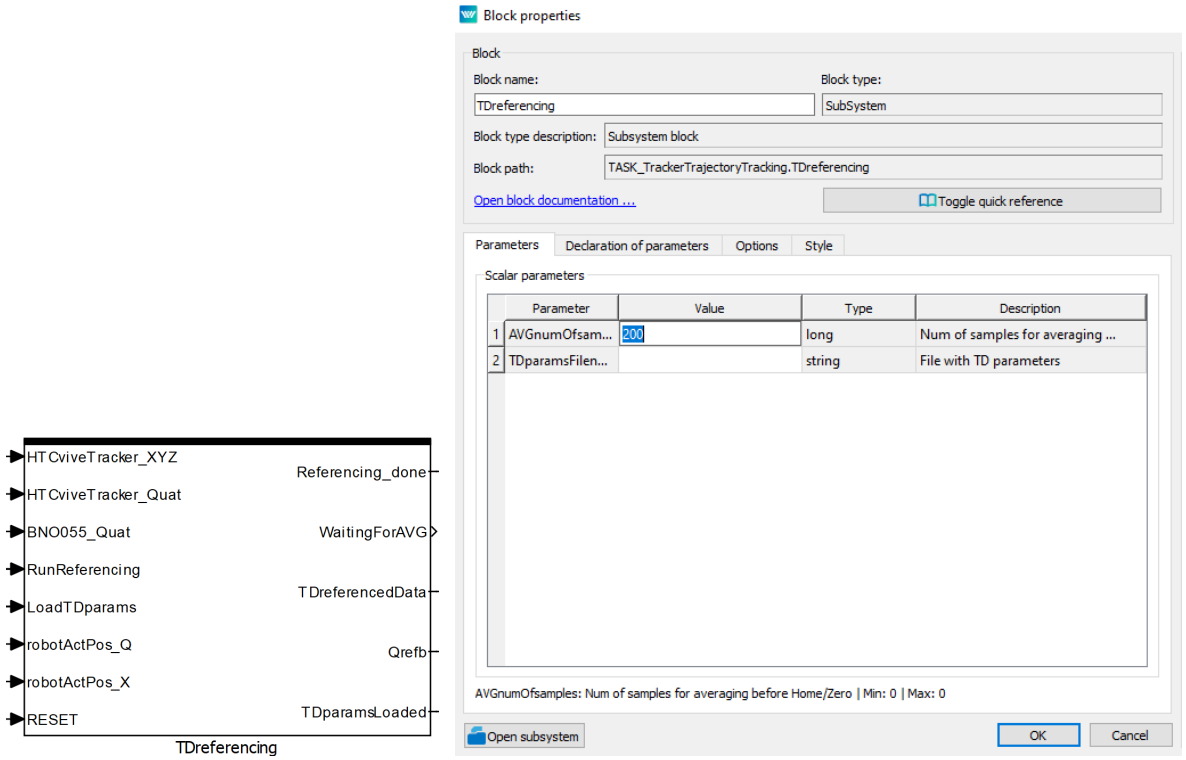


Obrázek 11: Souřadné systémy na trekovacím zařízení

SW modul záznamu (trekování) pohybu pracovního nástroje vedeného operátorem

Kromě formální funkce ukládání snímaných polohových dat řeší tento SW modul proces referencování trekovacího zařízení před zahájením snímání pohybu. Trekovací zařízení je v tomto případě umístěno dočasně na přírubu robotu a po zahájení referencování je aktuální měřená poloha trackerem korigována dle známé polohy robotu. Tento proces slouží k odstranění nahodilých chyb HTC Vive Trackeru jako například drift, ztráta/přeskok aktuální polohy, chyba v interní filtraci dat (fúze inerciálních dat z interní IMU a dat z optického lokalizačního systému - dáno principem funkce HTC Vive Trackeru).

Referencování trakovacího zařízení bylo zajištěno nově implementovaným funkčním blokem *TDreferencing*, viz Obrázek 12.
Chod trekování (referencování, záznam, uložení dat, atd.) byl řízen standardním stavovým automatem (blok *ATMT* v systému REXYGEN).



Obrázek 12: Funkční blok pro referencování trekovacího zařízení

SW modul zpracování zaznamenané trajektorie a replikace pohybu

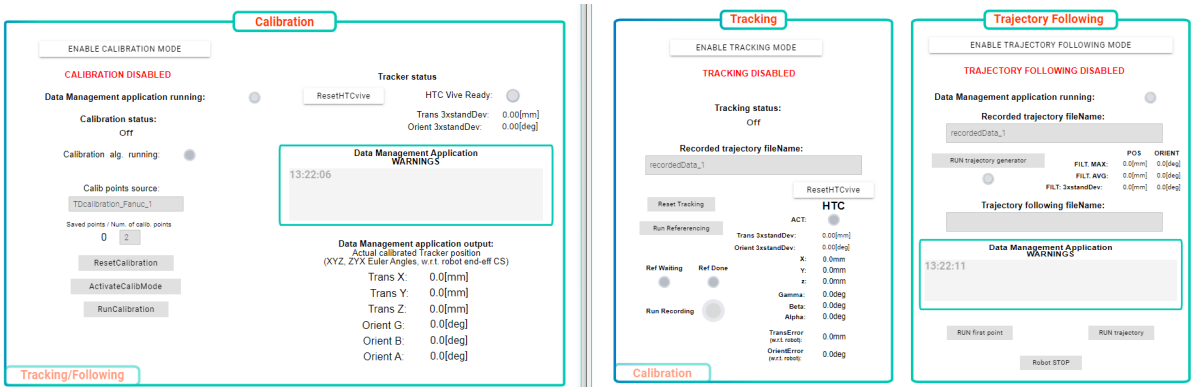
SW modul navazuje na TASK_TrackerMatlabInterface a realizuje pouze následující rozhraní:

- Povelování robotu (najetí do počátku záznamu, průjezd trajektorií, atd.), viz SW modul interfacu s robotem Fanuc
- Komunikaci signálů pro ovládání a monitoring vzdálené aplikace zpracování dat - datového postprocesoru, viz SW modul datového postprocesoru (Matlab)

SW modul operátorské ovládání (HMI)

SW modul představuje vlastní vizuální rozhraní celé aplikace trekovacího zařízení. Klíčovou výhodou navrženého řešení je možnost zobrazit vizualizaci v libovolném webovém prohlížeči na zařízení připojeném do stejné sítě jako řídicí počítač trekovacího zařízení. SW modul HMI je rozdělen do následujících obrazovek, viz Obrázek 13:

- Úvodní obrazovka - rozcestník
- Obrazovka k ovládání kalibrace trekovacího zařízení
- Obrazovka k ovládání snímání a replikace pohybu

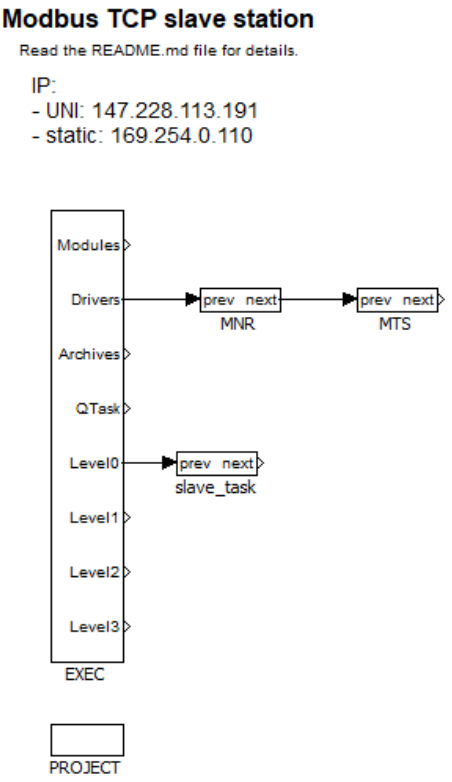


Obrázek 13: Okno ovládání kalibrace trekovacího zařízení (vlevo) a snímání a replikace pohybu

Podrobný popis uživatelského rozhraní včetně uživatelské dokumentace lze nalézt v [8].

Konfigurace MonarcoInteface

Jediným SW modulem konfigurace MonarcoInteface implementované v řídicím systému REXYGEN je **SW modul vzdálených vstupů/výstupů**, který slouží k ovládání logických signálů (záznam spouště, funkční tlačítko, signální sloupek) či signálů k ovládání serva spouště stříkací pistole.
Hlavní řídicí TASK *exec* sdružující všechny dílčí TASKy je znázorněn na Obrázku 14.



Obrázek 14: Hlavní TASK řídicího systému trekovacího zařízení (TRACKER)

Dílčí tasky příslušející SW modulům:

1. **SW modul vzdálených vstupů/výstupů**:
    a. **slave_task:** Implementace řízení vstupů/výstupů, dekódování logických signálů (např. pro signální sloupek), realizace PWM řízení serva.
2. **Samostatné funkční bloky:** Funkční bloky driverů použitých komunikačních protokolů: Modbus

Konfigurace MATLAB

Konfigurace MATLAB implementovaná v prostředí Matlab realizuje **SW modul datového postprocesoru (vzdálená aplikace) -** aplikace běžící na samostatném HOST počítači s instalovaným Matlabem - založeno na [2].

SW modul datového postprocesoru představuje klíčovou komponentu pro systém řízení trekovacího zařízením a slouží ke zpracování zaznamenaných polohových dat. Modul je implementován v prostředí Matlab a komunikuje s řídicím systémem ve dvou úrovních:

- Řídicí signály (binární, celočíselné, reálné, textové řetězce, atd.) se komunikují prostřednictvím protokolu RESTful API.
- Výměna rozměrných datových polí (záznamy bodů, trajektorií, výstupy zpracování) jsou komunikovány prostřednictvím protokolu FTP (výměna datových souborů .CSV)

Konfigurace a spuštění datového postprocesoru

Konfigurace datového postprocesoru se provádí konfiguračním skriptem config_trekZarizeni.m, ve kterém jsou uloženy všechny parametry:

1. Nastavení TARGET (konfigurace TRACKER) a HOST (konfigurace MATLAB) zařízení: IP adresy, hesla, cesty, zdrojové soubory dat, atd.
2. Model robotu: Kinematické parametry použitého robotického zařízení (Fanuc, Staubli, atd.), omezení na polohy, rychlosti, zrychlení kloubů, atd.
3. Parametry filtrace dat: Nastavení filtrů zpracovávající nasnímaná data
4. Parametry reinterpolace pohybu: Např. vzorkovací periody funkce DPM, časy rozjezdu/dojezdu, faktor rychlosti, atd.

Poznamenejme, že některé parametry z config_trekZarizeni.m lze přepisovat z operátorského rozhraní (HMI).
Více podrobností lze vyčíst z komentářů přímo ve zdrojovém kódu *config_trekZarizeni.m*.

Spuštění datového postprocesoru se provádí spuštěním skriptu *setup_REX.m*, který běží v uzavřeném cyklu s peridou 500ms a v každém kroku provádí následující operace:

1. Čtení/zápis vstupů z konfigurace TRACKER: Ve skriptu jsou konfigurovány tzv. “connection links” na dílčí vstupy/výstupy funkčních bloků v konfiguraci TRACKER běžící v reálném čase na průmyslovém počítači.
2. Spouštění zpracování dat z procesu kalibrace trekovacího zařízení - funkce *FCN_trekZarizeniKalibrace.m*.
3. Spouštění zpracování dat z procesu trekování pohybu - funkce *FCN_trekZarizeniTrekovani.m*.

4. Komunikaci uživatelských stavů: Signály do operátorského rozhraní - např. chybové logy, stavy dílčích procesů (funkcí), informace o běhu datového procesoru, atd.

Kalibrace trekovacího zařízení (funkce FCN_trekZarizeniKalibrace.m)

Zajišťuje zpracování zaznamenaných dat z procesu kalibrace trekovacího zařízení, viz SW modul kalibrace trekovacího zařízení. Algoritmus kalibrace je založen na kalibračním modelu trekovacího zařízení, který je vytvořen odděleně pro kalibraci translace a orientace HTC Vive trackeru na přírubě uvažovaného robotu.

Vstupy algoritmu:

- Nasnímaná data z kalibračního procesu: Měřená poloha (translace a orientace) HTC Vive Trackerem
- Skutečná poloha příruby robotu (translace a orientace)
- Data uložena ve vstupním CSV souboru

Výstupy algoritmu:

- Transformace posunutí s.s. HTC Vive Trackeru vůči přírubě robotu - transformace mezi s.s. F_{TD} a $F_{HTC,}$ viz SW modul kalibrace trekovacího zařízení
- Data uložena ve výstupním CSV souboru

Algoritmus:

- Detailně popsány v [1], kde byl původní OTUS tracker nahrazen HTC Vive Trackerem. Více podrobností lze vyčíst z komentářů přímo ve zdrojovém kódu *FCN_trekZarizeniKalibrace.m*.

Zpracování nasnímaných dat z trekování (funkce FCN_trekZarizeniTrekovani)

Zajišťuje zpracování zaznamenaných dat z procesu trekování pohybu, viz SW modul záznamu (trekování) pohybu pracovního nástroje vedeného operátorem, SW modul zpracování zaznamenané trajektorie a replikace pohybu. Algoritmus zpracování pohybu je rozdělen do následujících částí:

1. **Filtrace dat:** Probíhá nad polohovými daty nekauzálním filtrem (Zero-phase digital filtering) typu dolní propust[22]. Tato technika filtrace umožňuje pracovat s amplitudovým spektrem zpracovávaného signálu při minimálním možném (nulovém) zkreslení fáze dílčích harmonických složek. Tento postup je žádoucí z hlediska odstranění nežádoucích složek záznamu pohybu působících typicky na vysokých frekvencích (šum měření, třes ruky operátora při záznamu pohybu, mechanické vibrace trekovacího systému) a jejich oddělení od užitečné složky signálu obsahujícího trajektorii ukládaného pohybu. Nulové fázové zpoždění zavedené filtrem je výhodné pro potlačení nežádoucího zkreslení zaznamenané trajektorie. Pro realizaci byla využita technika nekauzální IIR filtrace známá jako “forward-backward filtering” s optimalizací počáteční a koncové podmínky interního stavu filtru Výsledný proces filtrace byl zvolen na základě kompromisu mezi jeho implementační složitostí, rychlostí, robustností a snadné konfiguraci.

[22] https://www.mathworks.com/help/signal/ref/filtfilt.html

2. **Výpočet inverzní kinematické úlohy:** Výpočet poloh/rychlostí/zrychlení aktuátorů pro zvolený typ robotu, který generuje požadovaný pohyb. Slouží především k:
    a. Řízení samotného robotu při replikaci pohybu (např. s roboty Staubli, kdy je řízení realizováno přímo povelováním dílčích kloubů robotu)
    b. Ověření realizovatelnosti generovaného pohybu:
        i. Existence řešení inverzní kinematiky robotu (např. generovaný pohybu se nenachází v singulárních polohách či mimo pracovní prostor robotu daný jeho kinematickým uspořádáním).
        ii. Dodržení min. a max. poloh aktuátorů (generovaný pohyb je v pracovním prostoru robotu daném konstrukčním omezením pohybu kloubů).
        iii. Dodržení limitů na maximální rychlost a zrychlení kloubů - dáno omezeným výkonem aktuátorů robotů.
3. **Generování dat pro řídicí systém robotu pro replikaci pohybu:** Podpořeny výsledně dva typy robotů:
    a. Staubli: Robot Staubli TX40 pro testovací účely - generován datový soubor CSV přímo pro vstup do bloku *RM_Feed* v systému REXYGEN + propojení s robotem přes HW rozšíření Stäubli uniVAL Drive + komunikace EtherCAT, viz [10].
    b. Fanuc: Robot LR Mate 200iD, M800iA60 pro cílovou aplikaci. Z více možností komunikací, viz [3], nakonec zvoleno rozhraní prostřednictvím funkce Fanuc DPM, viz SW modul interfacu s robotem Fanuc. Generován datový soubor CSV přímo pro vstup do bloku DPMtrajectoryGenerator.

Vstupy algoritmu:

- Nasnímaná data z procesu trekování: Referencovaná měřená poloha trekovacího zařízení (translace a orientace) reprezentovaná přírubou robotu (s.s. F_{TD}), viz SW modul kalibrace trekovacího zařízení
- Data uložena ve vstupním CSV souboru

Výstupy algoritmu:

- Generovaný datový soubor pro blok *RM_Feed* (Staubli)
- Generovaný datový soubor pro blok *DPMtrajectoryGenerator* (Fanuc)
- Data uložena ve výstupním CSV souboru

Algoritmus:

- Více podrobností lze vyčíst z komentářů přímo ve zdrojovém kódu *FCN_trekZarizeniTrekovani.m*.

Zdroje dat:

Řídicí systém prototypu je dostupný v podobě zdrojových souborů (REXYGEN, MATLAB) na níže uvedeném odkazu:

https://drive.google.com/drive/folders/1GJPq6L8Psvfkx5y452lTaMLI7CRNMgZ9?usp=sharing

Reference

[1] J. Reitinger, M. Švejda: *Návrh senzorického systému a zpracování dat (Kalibrace a testování systému OTUS tracker)*, ZČU, 2018.

[2] J. Reitinger, V. Šetka, M. Švejda, O. Severa: *Komplexní řídicí SW robotické buňky v pilotní aplikaci (Implementační a uživatelská dokumentace)*, ZČU, 2020.

[3] J. Reitinger, V. Šetka, M. Švejda: *Software interfacu s průmyslovým robotem (Implementační a uživatelská dokumentace)*, ZČU, 2020.

[4] M. Švejda, J. Reitinger, M. Goubej: *Návrh senzorického systému a zpracování dat*, ZČU, 2020

[5] *Návrh interfacu s průmyslovým robotem (Protokol o testování SW)*, ZČU, LT, 2020.

[6] M. Švejda: *Operátorské navádění a vizualizace prototypu*, ZČU, 2020.

[7] *Komplexní řídicí SW robotické buňky v pilotní aplikaci (Protokol o testování SW)*, *ZČU, LT, 2020.*

[8] M. Švejda, O. Severa: *Software zpracování dat senzorického systému, řídicí systém (Uživatelská dokumentace)*, ZČU, 2021.

[9] M. Švejda, A. Jáger: Kompletní dokumentace prototypu (Výkresová a implementační dokumentace, protokol z testování), ZČU, 2021.

[10] J. Reitinger, V. Šetka: *Aktivita 3.3: Vývoj interfacu s průmyslovým robotem*, ZČU, 2019.