

# BPC-PRP

## Projekt: Únik z bludiště

1<sup>st</sup> Sevada Saakian  
FEKT  
Vysoké Učení Technické v Brně  
Brno, Česká Republika  
240879@vut.cz

2<sup>nd</sup> Nikita Kolyuka  
FEKT  
Vysoké Učení Technické v Brně  
Brno, Česká Republika  
xkolyu00@vutbr.cz

**Abstract**—Předkládaný projekt se zabývá návrhem a implementací systému umožňujícího autonomní navigaci mobilního robota v prostředí bludiště s cílem jeho opuštění bez kolize se stěnami. K řešení této úlohy robot využívá komplexní senzorický systém, zahrnující LiDAR pro detekci vzdáleností a překážek, kamerový systém s funkcí detekce Aruco značek pro účely lokalizace a orientace, inerciální měřicí jednotku (IMU) pro monitorování náklonu a rotace, enkodéry kol pro přesné odometrické měření a tlačítka s LED indikátory pro uživatelské rozhraní a vizuální signalizaci. Klíčovým aspektem projektu je dosažení přesné lokalizace a bezkolizní navigace robota prostřednictvím aplikace pokročilých algoritmů pro zpracování senzorických dat a řízení pohybu.

**Index Terms**—Robotics, project, BPC-PRP, ROS2, OpenCV, LiDAR

### I. ÚVOD

Cílem soutěžní úlohy je únik robota z bludiště o rozměrech  $8 \times 8$  buněk, přičemž každá buňka má velikost  $40 \times 40$  cm. Robot má za úkol najít cestu ven bez kontaktu se stěnami, přičemž jeho výkon je hodnocen na základě času potřebného k úniku.

Bludiště obsahuje tři tzv. *Minotaury*, z nichž každý přidává penalizaci +30 s k celkovému času. Naopak, nalezení *pokladu* snižuje čas o –30 s. Každý dotek stěny je penalizován přičtením +10 s. Pokud robot narazí do stěny a dojde k jejímu poškození, tým je diskvalifikován (DNQ).

Struktura bludiště je navržena bez smyček – existuje právě jedna možná úniková cesta. V orientaci robotovi pomáhají značky *ArUco*, které vizuálně označují směr úniku nebo trasu vedoucí k pokladu.

Každý tým má k dispozici tři pokusy. Jeden pokus je časově omezen na maximálně 5 min, přičemž mezi jednotlivými pokusy musí být dodržena pauza 45 min.

V rámci tohoto projektu byl vytvořen software postavený na architektuře *ROS 2*, který kombinuje zpracování dat z různých senzorů (LiDAR, kamera, IMU, enkodéry) s rozhodovací logikou implementovanou v hlavní řídicí smyčce. Značky *ArUco* jsou detekovány pomocí knihovny *OpenCV* a celá navigace je navržena tak, aby probíhala bez kontaktu se stěnami, s důrazem na modularitu a přesnost.

### II. ARCHITEKTURA SYSTÉMU

Robotický systém je postaven na frameworku *ROS 2* (*Robot Operating System 2*), který poskytuje flexibilní a modulární



Fig. 1. Robots

infrastrukturu pro vývoj distribuovaných robotických aplikací. *ROS 2* umožňuje komunikaci mezi jednotlivými komponentami pomocí modelu *publisher-subscriber* a zajišťuje efektivní výměnu dat v reálném čase.

Pro zajištění přehledné a udržovatelné architektury je systém rozdělen do samostatných *ROS 2 node-ů*, přičemž každý zajišťuje specifickou funkcionalitu:

- **Lidar Node** – zajišťuje sběr a předzpracování dat ze senzoru LiDAR.
- **Camera Node** – obstarává komunikaci s kamerou robota, zajišťuje snímání a přenos obrazu.
- **IO Node** – řídí digitální vstupy a výstupy, včetně čtení stavů tlačítek a ovládání LED diod.
- **IMU Node** – zpracovává data z inerciální měřicí jednotky (IMU) pro odhad pohybu a orientace.
- **Line Loop Node** – hlavní řídicí smyčka robota, která koordinuje data z ostatních uzlů a určuje chování robota.

Pro detekci značek *ArUco* je využita knihovna *OpenCV*, která poskytuje robustní nástroje pro zpracování obrazu. Detekce je integrována do kamerového uzlu a umožňuje odhad polohy značek v prostředí v reálném čase.

Vedle těchto uzlů obsahuje projekt složku *algorithms*,

kde jsou umístěny pomocné výpočetní moduly:

- aruco\_detector.hpp – logika detekce Aruco tagů pomocí OpenCV.
- lidar\_filtr.hpp – filtrování a předzpracování dat z LiDARu.
- pid.hpp – univerzální implementace PID regulátoru pro řízení a stabilizaci.
- planar\_imu\_integrator.hpp – integrace a fúze dat z IMU pro výpočet orientace.

#### Pomocný modul helper.hpp

Pro sjednocení často používaných konstant a pomocných funkcí byl vytvořen hlavičkový soubor helper.hpp. Tento modul obsahuje:

- Fyzikální parametry robota jako například poloměr kola, rozvor nebo počet tisků na jednu otočku enkodéru.
- Hraniční hodnoty pro čidla (např. vzdálenost od zdi, maximální hodnota černé barvy u čidel čáry).
- Omezení pro motory, včetně maximálního a minimálního PWM signálu.
- Výčtové typy pro identifikaci Aruco značek.
- Definice ROS 2 témat v jmenném prostoru Topic (např. /imu, /camera, /line\_sensors).
- Pomocné funkce jako výpočet průměru (mean), převody mezi stupni a radiány (deg2rad, rad2deg) a funkce pro určení znaménka (sgn).

### III. ALGORITMUS

1) *Senzory:* Navigace robota v bludišti je založena na datech ze tří klíčových senzorů: LiDARu, kamery a inerciální měřicí jednotky (IMU).

LiDAR poskytuje data o vzdálenostech překážek v plném horizontálním rozsahu 360° kolem robota. Pro řízení pohybu a základní rozhodování, jako je detekce stěn a volného prostoru, jsou primárně využívána data z několika předem definovaných úhlových sektorů: přímo před robotem, nalevo a napravo od něj (viz Obrázek 2). Kromě těchto základních sektorů jsou dále využívány dva rozšířené sektory, označované jako **wide left** a **wide right**, které se vyznačují širším úhlovým záběrem.

Data získávaná ze základních sektorů (přední, levý, pravý) jsou zpracovávána výpočtem střední hodnoty všech naměřených vzdáleností v rámci každého sektoru. Tyto střední hodnoty slouží především pro vzájemné porovnání vzdáleností a pro následné rozhodovací procesy v navigačním algoritmu.

V případě sektorů **wide left** a **wide right** je zvolen odlišný přístup ke zpracování dat: v každém z těchto širokých sektorů je vyhledána minimální naměřená vzdálenost. Tento postup má za cíl approximovat kolmou vzdálenost ke stěně, čímž se snižuje závislost měření na aktuálním natočení robota vůči stěnám koridoru. Takto určené minimální vzdálenosti jsou následně využívány především pro řízení směru jízdy pomocí PID regulátoru, jelikož poskytují robustnější vstupní hodnotu méně ovlivněnou relativní orientací robota.

Kameru využíváme pro detekci a čtení QR kódů umístěných na specifických místech v bludišti, typicky na křížovatkách. Tyto kódy obsahují zakódované informace, které robotovi udávají požadovaný směr další jízdy.

Inerciální měřicí jednotku (IMU) využíváme pro přesné řízení otáčení robota na místě. Během otáčení měří IMU aktuální úhel natočení robota vůči výchozí orientaci a řídicí systém zastaví motory pohonů kol, jakmile robot dosáhne požadovaného cílového úhlu pro další segment cesty.

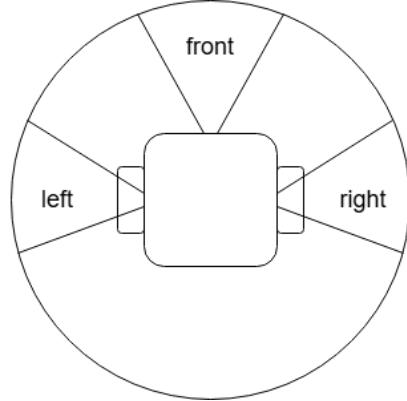


Fig. 2. Schéma vyhodnocovaných oblastí dat ze senzoru lidar.

2) *Stavový automat:* Řídicí algoritmus robota ve třídě LineLoop je implementován jako stavový automat. Tento automat se skládá ze čtyř hlavních stavů: Kalibrace IMU (CALIBRATION), Sledování koridoru (CORRIDOR\_FOLLOWING), Provádění otočení (TURNING) a Vyhodnocení po otočení (AFTER\_TURNING). Přechody mezi těmito stavami jsou dynamicky řízeny na základě dat ze senzorů (především LiDARu a IMU) a logiky rozhodování o dalším pohybu, včetně pokynů z Aruco značek. Schéma tohoto stavového automatu je znázorněno na Obrázku 3.

Při inicializaci, nebo pokud je to vyžadováno, robot vstupuje do stavu CALIBRATION. V tomto stavu řídicí smyčka maze\_loop čeká na dokončení kalibrace inerciální měřicí jednotky (IMU). Jakmile je IMU zkalirovaná, systém automaticky přechází do stavu CORRIDOR\_FOLLOWING.

Ve stavu CORRIDOR\_FOLLOWING robot autonomně naviguje chodbami. K udržení správné trajektorie využívá PID regulaci na základě dat z LiDARu: buď se centruje mezi dvěma stěnami, nebo sleduje jednu stěnu, pokud je druhá strana chodby otevřená. Rozhodovací logika v tomto stavu analyzuje geometrii prostoru detekovanou LiDAREm – například vzdálenost k překážce vpředu, otevřenosť cest vlevo či vpravo. Pokud je detekována překážka vpředu a není možné pokračovat rovně, robot iniciuje otočení (např. o 180° ve slepé uličce, pokud jsou obě strany uzavřené, nebo o 90° pokud je volná cesta do strany). Na křížovatkách může být rozhodnutí o směru ovlivněno detekovanou Aruco značkou. Při rozhodnutí o změně směru systém přechází do stavu TURNING.

Stav TURNING řídí samotný proces otáčení robota. Tento proces je v kódu často rozdělen do dříve uvedených fází pomocí proměnné. Může zahrnovat například fazu TurnStage::FORWARD\_TO\_EDGE, kdy se robot nejprve mírně posune vpřed, aby se dostal na vhodnější pozici pro otočení, a následně fazu TurnStage::ROTATE pro samotné otáčení na místě. Cílové natočení je stanoveno na základě

předchozího rozhodnutí (např. otočení o  $90^\circ$ ,  $180^\circ$ , nebo směr určený Aruco značkou). Robot využívá data z IMU k monitorování aktuálního natočení a otáčí se, dokud nedosáhne s definovanou tolerancí. Po úspěšném dokončení rotační fáze systém přechází do stavu AFTER\_TURNING.

Stav AFTER\_TURNING slouží k bezprostřednímu vyhodnocení situace po dokončení otočení. Robot v tomto stavu znovu analyzuje okolí pomocí LiDARu. Pokud je před ním volná uzavřená chodba, přechází do stavu CORRIDOR\_FOLLOWING pro pokračování jízdy. V opačném případě – například pokud je stále příliš blízko stěny, nachází se v otevřeném prostoru vyžadujícím specifické sledování stěny, nebo čelí nové překážce – může pokračovat v pohybu s upravenou logikou přímo ze stavu AFTER\_TURNING, nebo dokonce iniciovat další otočení přechodem zpět do stavu TURNING, pokud to aktuální situace vyžaduje.

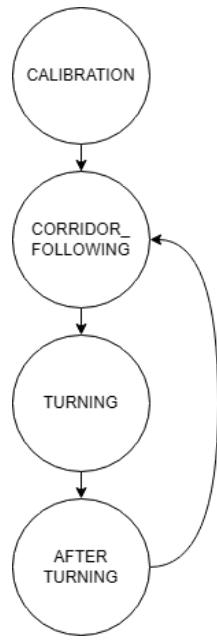


Fig. 3. Schéma stavového automatu řídícího algoritmu.

3) *Rozhodování o směru jízdy:* Základní rozhodování algoritmu o potřebě a směru otáčení v rámci bludiště je založeno především na analýze dat z LiDARu odpovídajících třem klíčovým směrům: vpřed, vlevo a vpravo.

Pro každý z těchto tří směrů systém pracuje s reprezentativní hodnotou vzdálenosti  $d$  k nejbližší překážce. Tato hodnota  $d$  (jejíž konkrétní způsob určení z dat LiDARu – např. průměrná či minimální vzdálenost v daném sektoru – je popsán v sekci ??) je porovnána s definovanou prahovou hodnotou  $d_{thresh}$ . Pokud je naměřená vzdálenost  $d < d_{thresh}$ , daný směr je klasifikován jako blokován (obsahuje překážku, znače S). V opačném případě ( $d \geq d_{thresh}$ ) je směr považován za volný (V). Tímto způsobem jsou odvozeny tři binární příznaky popisující stav bezprostředního okolí robota: Přední (P), Levý (L) a Pravý (R). V implementaci třídy LineLoop těmito klasifikacemi funkčně odpovídají vstupní data jako je porovnání hodnoty lidar\_results\_.front

s konstantou FRONT\_WALL\_DISTANCE a přímo poskytované booleovské příznaky isLeftOpen/isLeftClosed a isRightOpen/isRightClosed.

Kombinací stavů těchto tří binárních příznaků (P: S/V, L: S/V, R: S/V) existuje celkem  $2^3 = 8$  možných základních konfiguračních situací, ve kterých se robot může nacházet vůči okolním překážkám. Tyto situace jsou pro ilustraci znázorněny na obrázcích 4 a 5.

Toto základní vyhodnocení stavu okolí (P, L, R) poskytuje důležitý vstup pro navigační logiku. Nicméně, pro robustní a plynulé řízení v komplexním prostředí bludiště se algoritmus nemůže spoléhat výhradně na tuto zjednodušenou okamžitou klasifikaci. Robot se totiž může nacházet v řadě specifických a dynamických situací (například těsně po dokončení otáčecího manévrů, při průjezdu atypicky tvarovanou částí bludiště nebo při nutnosti precizního manévrování podél stěny), kde je potřeba detailnější řízení a zohlednění širšího kontextu. Právě z těchto důvodů implementace ve třídě LineLoop využívá pokročilejší mechanismy stavového automatu. Mezi ně patří dedikovaný stav AFTER\_TURNING, určený pro reevaluaci situace a stabilizaci pohybu po otočení, a interní fáze (definovány pomocí turn\_stage\_) ve stavu TURNING, jako je například TurnStage::FORWARD\_TO\_EDGE pro optimalizaci pozice před samotnou rotací. Tyto prvky umožňují algoritmu lépe reagovat na dynamiku pohybu a konkrétní podmínky encountered v bludišti nad rámec jednoduché osmistavové klasifikace.

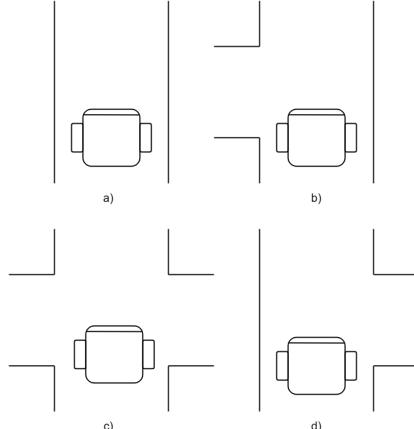


Fig. 4. Ilustrace případů rozhodování s volným předním sektorem (Skupina 1: stav 1a–1d).

4) *Udržování pozice v koridoru:* Pro udržení stabilní a bezpečné trajektorie robota při průjezdu koridorem (typicky ve stavech CORRIDOR\_FOLLOWING a AFTER\_TURNING) je nasazen PID regulátor. Jeho primárním úkolem je vypočítat korekční úhlovou rychlosť robota. Tato úhlová rychlosť je následně, v kombinaci s požadovanou dopřednou lineární rychlosťí, pomocí inverzní kinematiky (kinematics\_.inverse()) transformována na konkrétní rychlosti pro levé a pravé kolo motoru.

a) *Centrování v koridoru:* Pokud LiDAR indikuje přítomnost stěn na obou stranách robota, algoritmus se

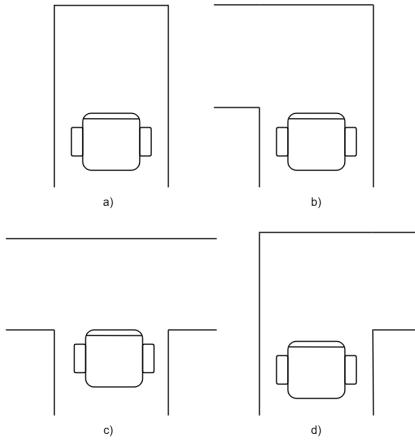


Fig. 5. Ilustrace případů rozhodování s blokovaným předním sektorem (Skupina 2: stavy 2a–2d).

snaží udržovat robota přibližně uprostřed chodby. Regulační odchylka pro PID regulátor je určena jako rozdíl naměřených vzdáleností. Na základě této odchylky, nastavených PID konstant a aktuálního časového kroku, metoda vypočítá požadovanou korekční úhlovou rychlosť. Spolu se základní lineární rychlostí tvoří vstup pro modul inverzní kinematiky. Pomocí inverzní kinematiky dostavame nutnou hodnotu rychlosti pro motory. Bez tohoto modulu je velmi složité spravně nastavit parametr zesilení u PID.

b) *Sledování stěny (Wall Following):* V situacích, kdy je jedna strana koridoru detekována jako otevřená, algoritmus může přejít do režimu sledování protilehlé stěny. Tato strategie přímého pohybu se sledováním stěny se uplatní, je-li jedna strana koridoru otevřená, a zároveň buď není detekovanou Aruco značkou určeno okamžité odbočení do volného prostoru, nebo se robot nachází blízko čelní překážky.

Pro řízení využívá hodnoty ze širších bočních sektorů LiDARu a cílová požadovaná vzdálenost od stěny (WALL\_DISTANCE). Výstupem je opět korekční úhlová rychlosť, která je dále, analogicky k případu centrování v koridoru, zpracována modulem inverzní kinematiky pro řízení motorů.

#### IV. VÝZVY A ÚSPĚCHY

Jednou z největších výzev během vývoje byla implementace a správné naladění PID regulátoru pro spolehlivé řízení pohybu robota v koridoru. V počátečních fázích jsme se potýkali s nestabilitou řídicího systému, která se projevovala buď silnými oscilacemi robota kolem středové osy koridoru, nebo v horších případech nekontrolovaným zatáčením, jež vedlo k divergenci od požadované trajektorie a nárazům do stěn.

Pro dosažení stabilního chování bylo nutné provést důkladné ladění řídicího algoritmu. Experimentálně jsme optimalizovali nejen základní konstanty PID regulátoru ( $K_p, K_i, K_d$ ), ale také parametry související se zpracováním dat z LiDARu a časováním řídicí smyčky. Za tímto účelem byly do systému zpracování dat z LiDARu přidány nové, širší sektory, označované jako wide\_left a wide\_right. Tyto změny

si vyžádaly i úpravu celkové architektury řídicího programu. Původně program fungoval převážně asynchronně, kdy hlavní řídicí smyčka a operace robota probíhaly relativně nezávisle na aktuálním příchodu senzorických dat. Nově je však hlavní výkonný cyklus algoritmu (metoda maze\_loop) přímo vázán na příjem nových dat z LiDARu. Tímto přechodem na model řízený událostmi (konkrétně dostupností dat z LiDARu) byl eliminován významný problém, kdy robot mohl operovat na základě zastaralých informací o prostředí, což mohlo vést k chybným rozhodnutím, například k minutí bodu nutného pro odbočení.

Úspěšná stabilizace jízdy vpřed však odhalila novou problematickou oblast, která se manifestovala při manévrech vyžadujících přesné přiblížení k čelní stěně a následné otočení na místě. PID regulátor, optimalizovaný pro udržování centrální pozice v relativně přímém koridoru, měl tendenci nežádoucím způsobem reagovat na dynamicky se měnící vzdálenosti od stěn při přiblížování k čelní překážce nebo během přípravy na otočení. Pro komplexní řešení tohoto problému byl navržen a implementován rozšířený stavový automat, jak bylo detailněji popsáno v předchozích kapitolách. Klíčovým prvkem tohoto rozšíření bylo zavedení strategie řízení pomocí PID regulátoru založené na sledování jediné stěny (tzv. wall following).

Tato strategie využívající sledování jediné stěny se navíc ukázala jako robustní a velmi užitečná i pro zajištění bezpečného a předvídatelného průjezdu některými typy křížovatek, kde centrování mezi nejasně definovanými či chybějícími bočními "stěnami" může selhat. Úspěšná implementace a vhodné dynamické přepínání mezi těmito dvěma režimy řízení (centrování v koridoru a sledování jedné stěny) v rámci navrženého stavového automatu představuje klíčový úspěch tohoto projektu. Výsledkem je funkční systém schopný spolehlivě autonomně navigovat robota v testovacím bludišti, řízený mimo jiné i instrukcemi z detekovaných Aruco značek.

#### V. ZÁVĚR

V rámci tohoto projektu byl úspěšně vyvinut systém pro autonomní navigaci mobilního robota, jehož cílem je únik z bludiště. Díky integraci dat z LiDARu, kamery s detekcí Aruco značek a IMU v architektuře ROS 2, a implementaci robustního stavového automatu s PID regulací pro řízení pohybu, bylo dosaženo spolehlivé bezkolizní navigace. Výsledný systém tak prokazuje schopnost robota efektivně nalézt cestu ven z komplexního testovacího prostředí.