

# BPC-PRP

## Projekt: Únik z bludiště

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

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

**Abstract**—Cílem tohoto projektu je navrhnut a implementovat systém, který umožní robotovi samostatně opustit bludiště, aniž by se dotýkal stěn. Při řešení úlohy robot využívá plný rozsah svých senzorických schopností, mezi které patří LiDAR pro detekci vzdáleností a překážek, kamera s detekcí Aruco značek pro orientaci, čidla čáry pro sledování povrchu, inerciální jednotka (IMU) pro sledování náklonu a rotace, enkodéry kol pro měření pohybu a otáček, a tlačítka s LED diodami pro uživatelské vstupy a vizuální zpětnou vazbu. Důraz je kladen na přesnou lokalizaci a navigaci bez kolize, s využitím pokročilých algoritmů zpracování dat a řízení.

**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. Body se přidělují podle následující funkce:

$$p = \min(\max(kx + q, 0), 40)$$

kde proměnná  $x$  představuje upravený čas, který je ovlivněn penalizacemi a bonusy.

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.

Veškerý kód použitý během soutěže musí být uložen v online repozitáři [1], kde bude dostupný pro kontrolu a archivaci.

V rámci tohoto projektu byl vytvořen software postavený na architektuře ROS 2, který kombinuje zpracování dat z různých senzorů (LiDAR, kamera, IMU, enkodéry, čidla čáry) 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.

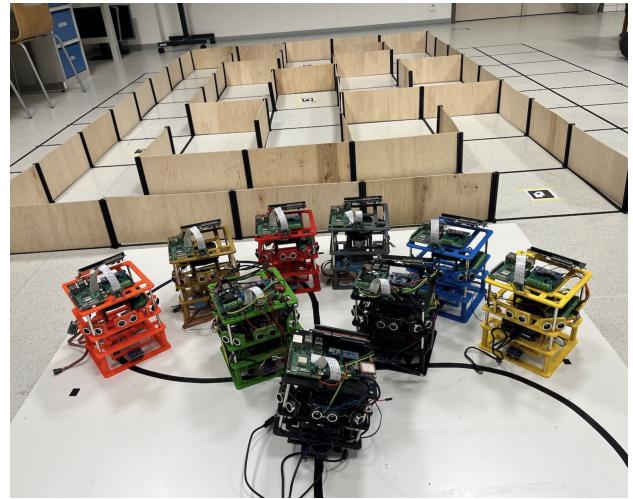


Fig. 1. Robots

### II. ARCHITEKTURA SYSTÉMU

Robotický systém je postaven na frameworku ROS 2 (*Robot Operating System 2*), který poskytuje flexibilní a modulární infrastrukturu pro vývoj distribuovaných robotických aplikací. ROS 2 umožňuje komunikaci mezi jednotlivými komponentami pomocí modelu *publisher-subscriber* a zajistuje 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 tiků na jednu otočku enkodéru.
- Hraníční 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).

Díky tomuto modulu je celý projekt lépe čitelný, přehlednejší a méně náhylný k chybám při opakovaném použití stejných hodnot nebo výpočtů.

Tato **modulární architektura** přináší několik klíčových výhod:

- **Škálovatelnost** – nové komponenty lze snadno přidávat bez zásahů do existujícího kódu.
- **Znovupoužitelnost** – algoritmy jsou nezávislé na konkrétním hardware.
- **Snadná údržba** – oddělené moduly umožňují rychlé ladění a testování.
- **Paralelní vývoj** – více vývojářů může pracovat současně na různých částech systému.
- **Robustnost** – chyba v jednom uzlu neohrožuje celý systém.

### III. ALGORITMUS

*1) Senzory:* Pro navigaci v bludišti se opíráme o data ze tří typů senzorů: lidaru, kamery a inerciální měřicí jednotky (IMU).

Lidar snímá data o vzdálenostech překážek v celém 360° rozsahu kolem robota. Pro samotné řízení pohybu vpřed a pro základní rozhodování v bludišti (např. detekce stěn a volného prostoru) využíváme primárně data z předem definovaných

seztorů: přímo před robotem, nalevo od něj a napravo od něj (viz Obrázek 2).

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.

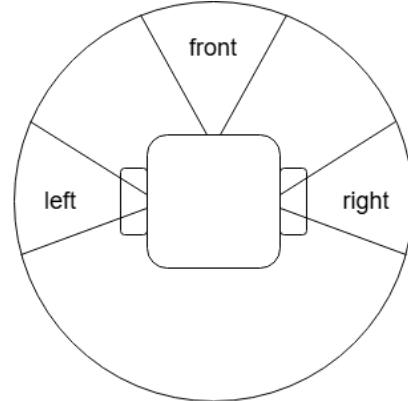


Fig. 2. Schéma vyhodnocovaných oblastí dat ze senzoru lidar.

*2) Stavový automat:* Řídicí algoritmus robota je implementován jako stavový automat se třemi hlavními stavami: Kalibrace (CALIBRATION), Otáčení (TURNING) a Sledování koridoru (CORRIDOR\_FOLLOWING). Přechody mezi stavami jsou řízeny na základě dat ze senzorů a vnitřní logiky. Schéma tohoto stavového automatu je znázorněno na Obrázku 3.

Při spuštění programu robot nejprve vstupuje do stavu CALIBRATION. Během této fáze probíhá nezbytná inicializace a kalibrace klíčových senzorů, zejména IMU a lidaru, aby byla zajištěna jejich přesná funkce. Po úspěšném dokončení kalibrace přechází systém automaticky do stavu CORRIDOR\_FOLLOWING.

Ve stavu CORRIDOR\_FOLLOWING se robot pohybuje autonomně vpřed koridorem v bludišti. Udržuje přitom bezpečnou vzdálenost od bočních stěn pomocí dat z lidaru. Součástí tohoto stavu je také rozhodovací logika, která na základě analýzy dat ze senzorů (především lidaru pro detekci geometrie prostoru a případně kamery pro čtení navigačních značek) identifikuje moment, kdy je nutné provést otočení – například na křížovatce. Jakmile algoritmus rozhodne o potřebě změny směru, systém přechází do stavu TURNING.

Ve stavu TURNING robot provádí otáčení na místě do požadovaného směru. Řídicí systém kontinuálně monitoruje aktuální úhel natočení robota pomocí IMU a porovnává jej s cílovým úhlem (který mohl být určen například instrukcí z QR kódu nebo je předdefinován jako standardní otočení o 90° či 180°). Jakmile je dosaženo cílového úhlu v rámci definované tolerance, otáčení je ukončeno a systém se vrádí

zpět do stavu CORRIDOR\_FOLLOWING pro pokračování jízdy v nově zvoleném směru.

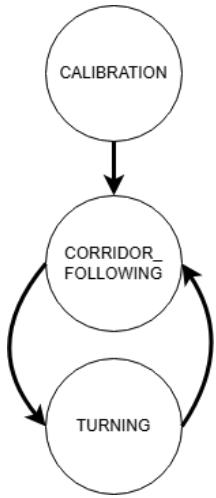


Fig. 3. Schéma stavového automatu řídícího algoritmu.

3) *Rozhodování o směru jízdy:* Základní rozhodování algoritmu o potřebě a směru otáčení v rámci bludiště je založeno především na analýze dat ze tří klíčových sektorů lidaru: vpředu, vlevo a vpravo.

Pro každý z těchto tří sektorů systém vyhodnocuje minimální naměřenou vzdálenost k nejbližší překážce. Tato vzdálenost je porovnána s definovanou prahovou hodnotou  $d_{thresh}$ . Pokud je naměřená vzdálenost  $d < d_{thresh}$ , sektor je klasifikován jako blokovaný (obsahuje "stěnu", S). V opačném případě ( $d \geq d_{thresh}$ ) je sektor považován za volný ("volno", V). Tímto způsobem získáváme tři binární příznaky popisující stav bezprostředního okolí robota pro sektory Přední (P), Levý (L) a Pravý (R).

Kombinací stavů těchto tří binárních příznaků (P: S/V, L: S/V, R: S/V) existuje celkem  $2^3 = 8$  možných unikátních situací, ve kterých se robot může nacházet vůči okolním překážkám.

Tyto kombinace jsou pro účely rozhodování rozděleny do dvou hlavních skupin v závislosti na stavu předního sektoru:

- **Skupina 1: Přední sektor je volný (P:V).** V těchto čtyřech případech (označených 1a–1d) může robot potenciálně pokračovat přímo vpřed, nebo se rozhodnout pro otočení na základě stavu bočních sektorů, případně dle externích instrukcí z QR kódů. Tyto případy jsou znázorněny na Obrázku 4.
- **Skupina 2: Přední sektor je blokovaný (P:S).** V těchto čtyřech případech (označených 2a–2d) je robot nucen změnit směr a musí se rozhodnout, zda se otočí doleva, doprava, nebo případně o 180°. Tyto případy jsou znázorněny na Obrázku 5.

Detailní popis rozhodovací logiky a akce robota pro každou z osmi možných kombinací (označených dle příslušnosti ke skupině a případu na obrázcích) je následující:

- **Stav 1a: [P:V, L:V, R:V]**  
**Akce:** Rozhoduj na základě dat z QR kódu.

(Otevřená křížovatka nebo prostor; směr určí externí instrukce.)

- **Stav 1b: [P:V, L:V, R:S]**  
**Akce:** Rozhoduj na základě dat z QR kódu.  
(Možnost jet rovně nebo doleva; směr určí externí instrukce.)
- **Stav 1c: [P:V, L:S, R:V]**  
**Akce:** Rozhoduj na základě dat z QR kódu.  
(Možnost jet rovně nebo doprava; směr určí externí instrukce.)
- **Stav 1d: [P:V, L:S, R:S]**  
**Akce:** Pokračuj rovně.  
(Koridor; jet rovně na základě PID.)
- **Stav 2a: [P:S, L:V, R:V]**  
**Akce:** Rozhoduj na základě dat z QR kódu.  
(Stěna vpředu (T-křížovatka/roh), volno vlevo i vpravo; směr určí externí instrukce.)
- **Stav 2b: [P:S, L:V, R:S]**  
**Akce:** Otoč doleva o 90°.  
(Jediná volná cesta vede doleva.)
- **Stav 2c: [P:S, L:S, R:V]**  
**Akce:** Otoč doprava o 90°.  
(Jediná volná cesta vede doprava.)
- **Stav 2d: [P:S, L:S, R:S]**  
**Akce:** Otoč o 180° (otoč se zpět).  
(Slepá ulička.)

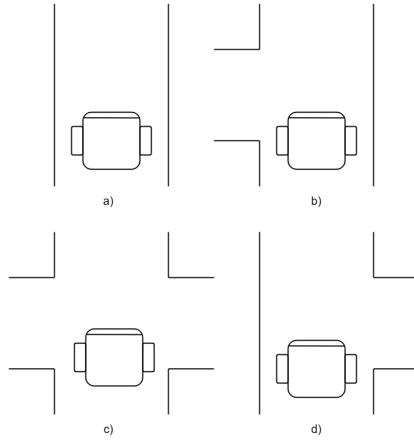


Fig. 4. Ilustrace případů rozhodování s volným předním sektorem (Skupina 1: stavы 1a–1d).

4) *Udržování pozice v koridoru:* Při jízdě robota koridorem (ve stavu CORRIDOR\_FOLLOWING) je pro udržení stabilní a bezpečné trajektorie využíván PID regulátor. Hlavním úkolem tohoto regulátoru je aktivně korigovat pohyb robota tak, aby se snažil dodržovat jízdu přibližně uprostřed mezi detekovanými bočními stěnami. Jako regulační odchylka (vstupní signál pro PID) se typicky používá rozdíl vzdáleností  $d_{left} - d_{right}$  naměřených lidarovým senzorem k levé a pravé stěně v definovaných bočních sektorech. Na základě této odchylky vypočítává PID regulátor korekční akci, která ovlivňuje řízení robota, rozdíl rychlosť levého a pravého kola.

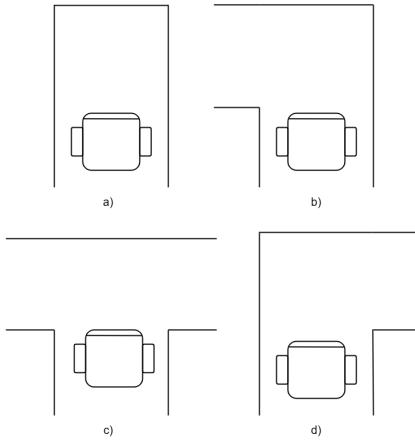


Fig. 5. Ilustrace případů rozhodování s blokovaným předním sektorem (Skupina 2: stavy 2a–2d).

Ve specifických situacích, kdy není možné nebo vhodné udržovat středovou pozici – například při průjezdu některými typy křížovatek nebo při finálním přiblížení k čelní stěně před zahájením otáčení – může algoritmus dočasně přepnout na jinou strategii řízení. Často se používá režim sledování jediné stěny (tzv. "wall following"). V tomto režimu je PID regulátor konfigurován tak, aby udržoval konstantní vzdálenost  $d_{target}$  od jedné vybrané stěny (např. levé). Regulační odchylka je pak  $d_{left} - d_{target}$  a cílem je udržet tuto odchylku nulovou, což vede k jízdě paralelně s vybranou stěnou.

#### IV. VÝZVY A ÚSPĚCHY

Jednou z největších výzev během vývoje byla implementace a správné naladění PID regulátoru pro spolehlivé řízení jízdy robota v koridoru. V počátečních fázích jsme se potýkali s nestabilitou řídicího systému. Ta se projevovala buď silnými oscilacemi robota kolem středové osy koridoru, nebo v horších případech nekontrolovaným zatáčením, které 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. Mezi ně patřila geometrie vyhodnocovaných sektorů (jejich úhlová šířka a relativní pozice vůči robotu) a prahová hodnota  $d_{thresh}$  pro detekci přítomnosti stěny. Po pečlivém nastavení těchto parametrů se podařilo dosáhnout stabilní a plynulé jízdy robota sředem koridoru.

Úspěšná stabilizace jízdy vpřed však odhalila nový problém, který nastával při manévrech vyžadujících přesné přiblížení k čelní stěně a následné otočení na místě. PID regulátor, optimalizovaný pro udržování centrální pozice v koridoru, měl tendenci nežádoucím způsobem reagovat na blížící se čelní stěnu nebo na poslední detekované boční stěny.

Abychom tento problém vyřešili, navrhli jsme a implementovali přepínání mezi dvěma režimy řízení. Pro fázi přiblížení ke stěně před otočením byl zaveden režim sledování jediné stěny ("wall following"). V tomto režimu se robot orientuje a

udržuje konstantní vzdálenost pouze od jedné, typicky boční, stěny. To umožňuje kontrolovanější a spolehlivější přiblížení k čelní překážce bez nežádoucích korekcí od PID regulátoru pro centrování.

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 "stěnami" může selhat. Úspěšná implementace a vhodné 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 podle zadaných instrukcí.

#### V. ZÁVĚR

V tomto projektu jsme navrhli a implementovali systém pro autonomní únik robota z bludiště, založený na modulární architektuře využívající framework ROS 2. Díky rozdělení systému do samostatných uzlů pro každý senzor a komponentu se nám podařilo vytvořit přehlednou, udržitelnou a snadno rozšířitelnou softwarovou strukturu. Detekce Aruco tagů pomocí OpenCV, využití dat z LiDARu a IMU, stejně jako oddělení jednotlivých výpočetních algoritmů do samostatných modulů přispěly k celkové robustnosti a flexibilitě systému.

Rídící algoritmus byl postaven na stavovém automatu, který zajišťoval přechody mezi režimy kalibrace, pohybu koridorem a otáčení. Navigace probíhala primárně na základě analýzy vzdáleností měřených lidar senzorem a orientace zjištované pomocí IMU. Kamera byla využita pro detekci ArUco značek, které poskytovaly dodatečné instrukce při rozhodování o směru pohybu.

Během vývoje jsme čelili několika výzvám, zejména při ladění parametrů pro bezpečné a přesné vyhýbání se stěnám, a také při minimalizaci chyb způsobených akumulací odchylek z IMU při otáčení. I přes tyto komplikace se nám podařilo vytvořit plně funkční řešení, které dokáže detektovat cestu a řídit robota bez kolizí.

#### Možné vylepšení

Jedním z potenciálních vylepšení systému bylo využití modelu přímky ve tvaru  $y = ax + b$  pro odhad úhlové odchylky robota vůči stěnám na základě více měření z lidar senzoru. Tento přístup by umožnil určit směr natočení robota relativně ke stěnám a přesněji upravit jeho směr během pohybu nebo otáčení. Takový algoritmus by mohl být použit buď jako doplněk k datům z IMU, nebo jako jejich náhrada, čímž by se snížil vliv kumulativní chyby v orientaci. Výsledkem by byla přesnější a stabilnější navigace v bludišti, zejména při častých změnách směru.

Celkově projekt ukázal, že dobré navržená architektura a integrace více senzorů umožňují realizovat i složitější úlohy v oblasti autonomní robotiky efektivně a spolehlivě.

## PODĚKOVÁNÍ

Rádi bychom poděkovali všem, kteří nás na naší cestě tímto projektem podporovali.

Zvláštní poděkování patří našim rodičům za jejich trpělivost, podporu a motivaci během celého studia.

Dále děkujeme našim přátelům, kteří nám byli oporou, poskytovali zpětnou vazbu a pomáhali nám překonávat obtížné chvíle.

V neposlední řadě děkujeme našim učitelům a školitelům, kteří nám poskytli cenné rady, odborné vedení a podpořili nás při řešení technických problémů. Bez jejich pomoci by tento projekt nevznikl v této podobě.

## REFERENCES

- [1] Nikita Sevada. *BPC-PRP: Robot Maze Escape Project*.  
<https://github.com/FlexShashlik/BPC-PRP/tree/main>.  
Accessed: 2025-05-12. 2025.