BPC-PRP PROJECT - MAZE ESCAPE
Šedá myš

Petr Danielka
ID: 240326
VUT FEKT Brno, Česká republika
240326@vutbr.cz

Lukáš Korzonek
ID: 240372
VUT FEKT Brno, Česká republika
240372@vutbr.cz

***Abstract*—Tento dokument slouží jako seznámení s naším řešením úniku z bludiště. V úvodní kapitole se seznámíme se zadáním a stanovenými pravidly. Ve druhé si řekneme o základním fungování robota a struktuře řešeného programu. V poslední kapitole se budeme zabývat konkrétními částmi kódu.**

***Index Terms*—Robotika, projekt, BPC-PRP, regulace, pid, bludiště, maze, escape, VUT, BUT, Brno**

I. Úvod

Projekt byl vypracován v rámci závěrečné zkoušky předmětu BPC-PRP. Cílem projektu bylo vytvořit program, který docílí vyjetí robota z bludiště. Robot je vybaven ultrazvukovými senzory, optickými snímači (snímače čar), LiDARem, kamerou a IMU. Bludiště je složeno z "buněk" 40x40 cm. Jsou v něm umístěny ArUco markery, které slouží jako nápovědy k rozhodování, v jakém směru se vydat na následující křižovatce. V bludišti se rovněž nachází poklad a 3 minotauři, jejichž nalezení vede k časovému bonusu/penalizaci.

- Zdi - červené čáry
- Únikova trasa - zelená čára
- Start - zelená buňka
- Poklad - modrá buňka
- Minotauři - červené buňky

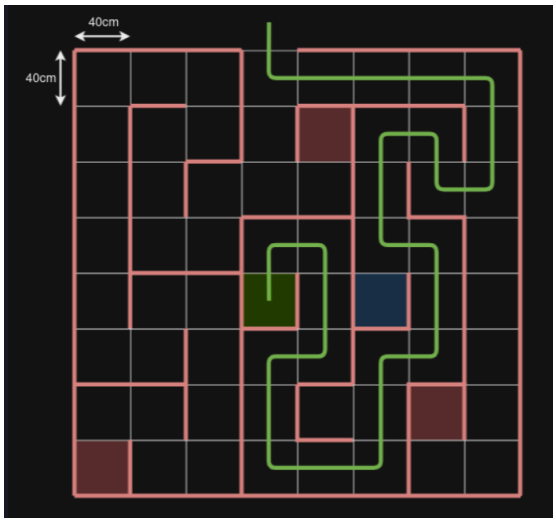

Fig. 1. Příklad bludiště

II. Fungování robota a struktura programu

K robotovi se program připojí pomocí rozhraní ROS2. Komunikace s robotem probíhá pomocí tzv. topiků. Každá komponenta robota má svůj vlastní komunikační kanál (topic), po kterém posílá nebo přijímá data ze sítě.

Program se skládá z jednoduchého mainu, kde se pouze vytvoří instance MazeNode, která řeší veškerou logiku robota. Dále je zde již zmíněná noda (podrobné vysvětlení jednotlivých částí nody je v následující kapitole). A dále zde jsou pomocné algoritmy (regulátor - pid.hpp, detekce ArUco markerů - aruco_detector.hpp).

III. Maze node

Maze node je jediná node využitá v programu. Obsahuje kompletní logiku řízení robota. Node je vytvořena v main a do chodu ji uvede executor.

A. Stavový automat

Hlavní částí je timer, který slouží jako main funkce node. Nachází se v ní stavový automat, který ovládá robota v závislosti na jeho aktuální situaci.

- Stav -1 - Počáteční stav (klidový)
- Stav 0 - kalibrace
- Stav 1 - pohyb kupředu
- Stav 2 - nastavení směru otáčení
- Stav 3 - otáčení

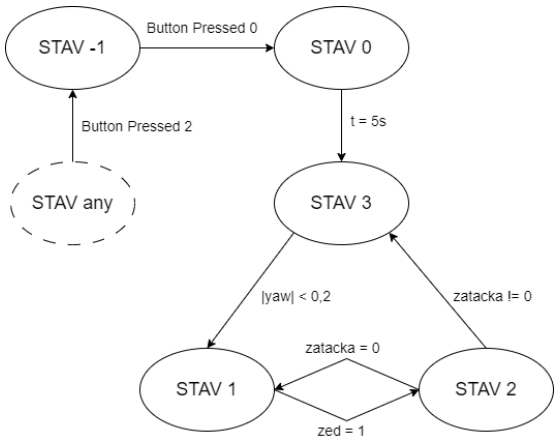

Fig. 2. Zjednodušený stavový diagram

1) Stav 0: Do stavu 0 (kalibrace) se dostaneme stiskem tlačítka 0. V tomto stavu se zkalibruje IMU a nastaví se yaw (ve stupních) podle úhlu, který vypočítá funkce calculateRotation (funkce vrací úhel, který robot svírá s bočními zdmi). Dále skočí do stavu 3.

2) Stav 1: Tento stav řeší samotnou cestu vpřed. tento stav je dále rozdělen pomocí podmínek do dalších situací, a to: robot detekuje na obou stranách zeď, robot detekuje zeď pouze na pravé straně, robot detekuje zeď pouze na levé straně, nebo jinak - robot nedetekuje zeď ani vlevo ani vpravo.

Podle situace následně nastavuje rychlost jednotlivých kol. V případě, že detekuje obě strany, je pro každou stranu vlastní regulátor, který řídí rychlost daného kola. Robot se snaží udržet stejnou vzdálenost na obou stranách.

Detekuje-li pouze jednu stěnu, je použit pouze jeden regulátor (pro druhé kolo s opačným znaménkem). Robot se snaží udržet v konstantní vzdálenosti od zdi (v polovině buňky - 20 cm).

V případě, že nedetekuje žádnou zeď, nastaví se maximální povolená rychlost na obě kola.

Následně ve všech případech se k vypočtené rychlosti pro jednotlivá kola odečte/přičte jednička, v závislosti na natočení od yaw.

Při detekci zdi se přechází do stavu 2.

3) Stav 2: V tomto stavu se pouze nastaví yaw na + nebo - 90 (v jistých případech i o 180, nebo se neupravuje) podle detekované zatáčky. Následně se přejde do stavu 3.

4) Stav 3: V tomto stavu se robot začne otáčet konkrétním směrem podle yaw. Rychlost otáčení je řízena regulátorem + je zaručena minimální rychlost otáčení. Robot se bude otáčet do doby, dokud nebude hodnota yaw téměř nulová (robot stojí ve směru jízdy). Poté znovu proběhne vypočtení úhlu svírajícího s okolními zdmi (toto je přidáno, jelikož po delší trase a mnoha zatáčkách se yaw příliš vychylovalo).

B. Funkce jednotlivých snímačů

1) Ultrazvuk: Byl pokus o využití ultrazvukových snímačů. První z nich je detekce zdi před robotem (primárně se k tomuto využívá LiDAR, v kódu to ovšem zůstalo jako pojistka).

Druhý způsob využití bylo zjištění, kdy se robot nachází uprostřed křižovatky pomocí detekce stěn bočními snímači. Nakonec jsme od tohoto využití upustili.

2) Tlačítka: Využita byla všechna 3 tlačítka. Tlačítko 0 spouští kalibraci (stav 0). Tlačítko 1 uvádí robota do chodu (toto tlačítko jsme nakonec nevyužívali, automaticky jsme z kalibrace přecházeli do dalšího stavu). Poslední tlačítko 2 je resetovací tlačítko, které uvede robota do klidového stavu -1.

3) IMU: Z této jednotky jsme využili pouze jednu hodnotu, a to úhlové zrychlení v ose z, ze které se následně v hlavním cyklu vypočítává aktuální hodnota yaw.

4) Kamera: Kamera je využita pro snímání ArUco markerů, které udávají informaci o směru jízdy na následující křižovatce. Ke zpracování je využita knihovna OpenCV. Součástí je algoritmus aruco_detector.hpp. Tento algoritmus se stará o samotnou detekci markerů a vrací ID, podle kterého se určuje směr.

Informace o následující křižovatce se ukládá do proměnné. Upřednostňuje se cesta k pokladu. Do proměnné se ukládají jak informace k východu, tak k pokladu. Ve chvíli, kdy je v proměnné již nějaká informace uložená, přepsat ji může jen informace k pokladu. Po projetí zatáčky se proměnná nastaví opět na defaultní.

5) LiDAR: Tento blok využívá pomocnou funkci getLidarRange, která vypočítá změřenou vzdálenost v určitém směru. Měříme hodnoty kolmo před robotem, na jeho stranách a za ním.

Podle těchto hodnot detekujeme zeď před robotem nebo křižovatku a určujeme směr při zatáčení. (Je zde také detekce slepé uličky - u pokladu - kdy se nastaví otočení o 180 stupňů)

Je-li detekováno více možných cest, nastaví se proměnná zatacka=0. Podmínka ve stavu 2 následně zajistí správné nastavení zatáčky podle informací z naskenovaného ArUco markeru. (Při křižovatkách, kde je možná cesta rovně, je detekce přední zdi posunuta o jednu buňku dál)

Pokud ovšem LiDAR detekuje pouze jedinou možnou cestu, nastaví se proměnná na 1 nebo 2.

Je zde rovněž přidáno rozsvěcování LEDek podle detekce blízkých stěn.

C. Pomocné funkce

1) double getLidarRange(const double midAngle, int numOfPoints, const int angle): Funkce slouží ke zpracování dat z LiDARu. Jako argumenty bere úhel ve stupních, počet bodů, který se má odečíst, případně úhel měření, podle kterého se počet bodů nastaví automaticky. Funkce počítá medián ze všech zadaných bodů, čímž eliminuje občasné výpadky měřených hodnot. S větším počtem bodů se zvýší šance, že výsledná hodnota bude definovaná, ale chyba může být větší, pokud chybí více hodnot. V programu je využívána ke zjišťování pozice a přítomnosti zdí.

2) double calculateRotation(): Tato funkce využívá trigonometrie ke zjištění úhlu, který robot se zdmi svírá. Funkce získává data z LiDARu výše popsanou funkcí. Rotace je vypočtena z rozdílu vzdáleností, které mají 2 paprsky LiDARu s pevně daným úhlem, jenž spolu svírají. Výpočet probíhá pro každou stranu zvlášť a vždy je proveden dvakrát, kolmý boční paprsek se propočítá s jedním pootočeným ve směru a s jedním proti směru hodinových ručiček. Tím se zvýší šance na získání použitelné hodnoty, ale nesnižuje přesnost výpočtu. Funkce je ošetřena proti měření v rozích nebo v otevřených prostranstvích bludiště, kde může zachytit vzdálenější stěny a znehodnotit tak výsledky.

IV. Závěr

Naše mise byla úspěšná a z bludiště se nám podařilo vyjet i se sebráním pokladu. Robot se celou dobu byl schopen držet převážně ve středu koridoru.

Je zde několik věcí, které by se daly vylepšit nebo optimalizovat. Regulátory ovládající rychlost kol nemají omezený akční zásah, díky tomu sice robot rychle reaguje, hrozí však, že požadovanou hodnotu přejede nebo se rozkmitá. Řešením by mohla být regulace úhlu natočení robota v závislosti na vzdálenosti od stěny namísto přímé regulace vzdálenosti.